Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Веб-сервіс для автоматизації інформаційних процесів спільноти мешканців одного чи кількох територіально поєднаних будинків "Житловий комплекс". Front-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-1

Янов Д.І.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Кравець Н.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Янову Данилу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-сервіс для автоматизації інформаційних процесів спільноти мешканців одного чи кількох територіально поєднаних будинків "Житловий комплекс". Front-end

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 01.06.2024

3. Вихідні дані до роботи Розробити FrontEnd застосунку для веб-сервісу житлових комплексів, який дозволить полегшити отримання необхідних сервісів та комунікацію між мешканцями

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	10.04.2024	*виконано*
3	Проектування ПЗ	13.04.2024	*виконано*
4	Розробка ПЗ	17.04.2024	*виконано*
5	Тестування ПЗ	09.05.2024	*виконано*
6	Оформлення пояснювальної записки	13.05.2024	*виконано*
7	Підготовка презентації та доповіді	22.05.2024	*виконано*
8	Попередній захист	04.06.2024	
9	Нормоконтроль, рецензування	02.06.2024	
10	Здача роботи у електронний архів	02.06.2024	
11	Допуск до захисту у зав. кафедри	04.06.2024	

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Янов Д.І.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Кравень Н.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 71 с., 17 рис., 12 джерел.

АДМІНІСТРУВАННЯ, ВЕБ СЕРВІС, ЖИТЛОВІ КОМПЛЕКСИ, КОМУНАЛЬНІ ПОСЛУГИ, МОВА ПРОГРАМУВАННЯ TypeScript, ФРЕЙМВОРК REACT, MOBX, SIGNALR

Об’єкт розробки – FrontEnd частина веб-сервісу для житлових комплексів.

Мета розробки – створення веб інтерфейсу для роботи з веб-сервісом для покращення функціонування житлових комплексів.

Метод рішення – середовище розробки JetBrains WebStorm, фреймворк React, мова програмування TypeScript, платформа Azure.

У результаті розробки створено FrontEnd додаток для веб-сервісу, який дозволяє полегшити отримання комунальних послуг, обмін повідомленнями між мешканцями, перегляд оголошень та обговорення нагальних потреб в житловому комплексі.

ADMINISTRATION, WEB SERVICE, RESIDENTIAL COPLEXES, COMMUNAL SERVICES, TypeScript PROGRAMMING LANGUAGE, REACT FRAMEWORK, MOBX, SIGNALR

The object of Development – the design and development of the FrontEnd part of a web service for residential complexes.

The purpose of Development – create a web interface for interacting with a web service to improve the functioning of residential complexes.

Solution Method – JetBrains WebStorm development environment, React framework, TypeScript programming language, Azure platform.

As a result of development a FrontEnd application for a web service has been created, which facilitates access to utility services, messaging between

residents, viewing announcements, and discussing urgent needs within the residential complex.

Я, Янов Данило Ігорович, студент гр. ПЗПІ-20-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Веб-сервіс для автоматизації інформаційних процесів спільноти мешканців одного чи кількох територіально поєднаних будинків "Житловий комплекс". Front-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У наш час цифрові технології стають невід'ємною частиною міського життя, значно впливаючи на його якість та ефективність управління житловими комплексами. Зростання чисельності населення в містах та підвищення вимог до комфорту проживання викликають необхідність у впровадженні сучасних рішень, які дозволяють мешканцям швидко та зручно взаємодіяти з адміністрацією, отримувати необхідні послуги та контролювати свої витрати.

Front-End розробка відіграє ключову роль у створенні таких веб-систем, оскільки саме вона відповідає за користувацький інтерфейс, зручність навігації та загальний досвід взаємодії з системою [1]. Інтуїтивно зрозумілий і привабливий інтерфейс значно підвищує задоволеність користувачів, спрощує доступ до інформації та послуг, а також сприяє ефективному управлінню житловим комплексом. Адаптивність дизайну дозволяє використовувати систему на різних пристроях, що є особливо важливим з повсякденним використанням мобільних пристроїв.

Сьогоднішні житлові комплекси все більше потребують інтеграції цифрових рішень, які можуть автоматизувати рутинні процеси, забезпечити прозорість і контроль над витратами, а також полегшити комунікацію між мешканцями і адміністрацією. Front-End розробка забезпечує можливість створення таких систем, які можуть включати функції подання заявок на ремонт, оплати рахунків, отримання інформації про новини та події, а також моніторингу енергетичних і водних витрат. Окремої уваги заслуговує питання безпеки даних і доступності для всіх категорій користувачів, включаючи людей з обмеженими можливостями.

Таким чином метою роботи є розробка FrontEnd застосунку для веб-сервісу, галуззю використання якого будуть територіальні об'єднання одного чи декількох будинків. Для житлових комплексів такий веб-сервіс є

надзвичайно актуальним і важливим, оскільки він дозволить вирішити багато проблем, які безпосередньо впливають на якість життя мешканців та ефективність управління. Більше того в процесі розробки будуть доступні широкі можливості для впровадження інноваційних рішень, що в подальшому сприятимуть сталому розвитку міського середовища та підвищенню комфорту проживання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Аналіз предметної галузі веб-систем для житлових комплексів з точки зору Front-End розробки передбачає детальне розуміння специфічних потреб користувачів та технічних вимог до створення ефективного, зручного та привабливого інтерфейсу. Веб-системи для житлових комплексів зазвичай включають кілька ключових функціональних компонентів, таких як управління об'єктами нерухомості, комунікація з мешканцями, обробка заявок та скарг, оплата послуг, а також інформування про новини та події. Основна мета Front-End розробки полягає в тому, щоб зробити ці функції максимально доступними та зрозумілими для користувачів.

Розробка таких систем починається з ретельного аналізу цільової аудиторії, яка включає як мешканців, так і адміністрацію житлових комплексів. Для мешканців важливо забезпечити простий доступ до інформації про рахунки, заявки на ремонт, розклад заходів та інші сервіси. Це вимагає створення інтуїтивно зрозумілих інтерфейсів, які адаптуються до різних пристроїв, зокрема мобільних телефонів і планшетів, оскільки користувачі часто використовують саме ці платформи для доступу до веб-систем. Використання адаптивного дизайну є критично важливим, оскільки він дозволяє підтримувати зручність користування незалежно від розміру екрану.

Важливим також є дизайн інтерфейсу, який повинен бути не лише функціональним, але й естетично привабливим. Використання сучасних принципів UI/UX дизайну дозволяє створювати зручні і привабливі інтерфейси, що сприяє підвищенню задоволеності користувачів. Особлива увага приділяється кольоровій схемі, типографіці, розташуванню елементів

на сторінці та інтерактивним елементам, які мають бути інтуїтивно зрозумілими і легко доступними

Не менш важливою є подальша інтеграція з різними зовнішніми сервісами, такими як платіжні системи, системи управління нерухомістю та інші, також є важливою частиною Front-End розробки. Це вимагає створення додаткового функціоналу з використанням API, що дозволить безперебійну і безпечну передачу даних між різними компонентами системи.

1.2 Огляд конкурентів

На ринку систем для управління житловими комплексами існує кілька конкурентів, які пропонують схожі послуги. Розглянемо деякі з них.

Condo Control Central також має широкий функціонал, наприклад замовлення послуг, оплата комунальних послуг та можливість голосування (див. рис. 1.1), та активно використовується в сполучених штатах. З переваг Condo Control Central можна виділити якісну оптимізацію інтерфейсу під мобільни пристрої. На головній сторінці можна одразу побачити усі можливі активності на території житлового комплексу, а також список оголошень та найближчих подій на календарі. Використовуючи бокове меню можна швидко перейти до інших сторінок, наприклад оплата рахунків, замовлення, голосування, тощо.

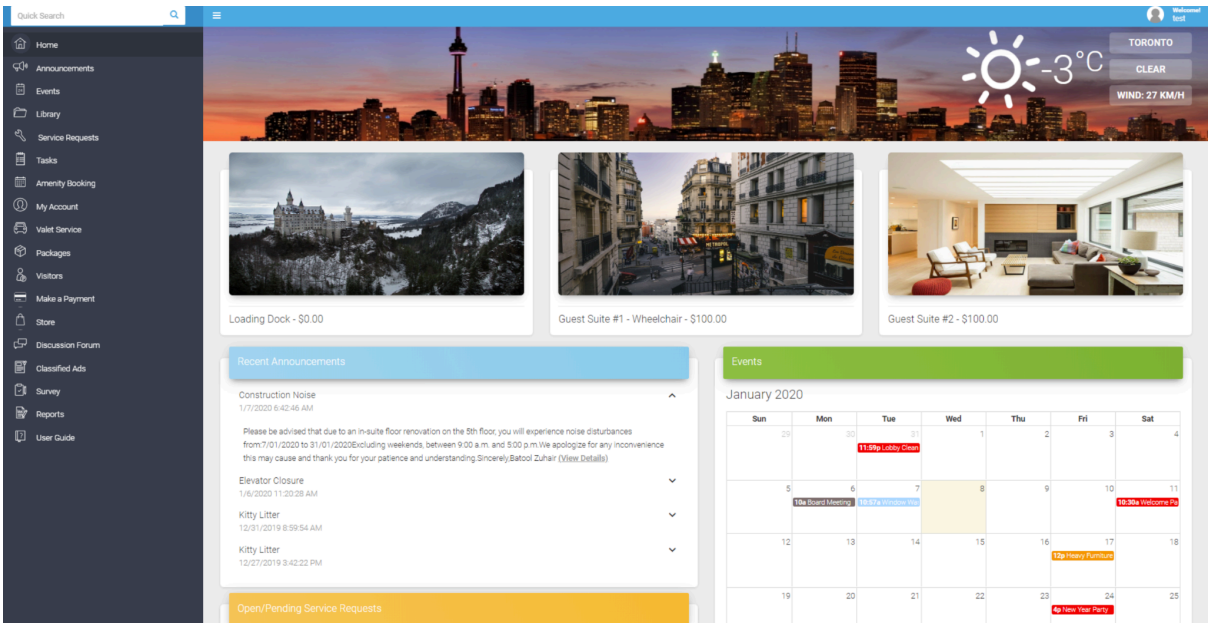


Рисунок 1.1 – Інтерфейс головної сторінки з акаунту мешканця в Condo Control Central (за даними [2])

BuildingLink також пропонує широкий функціонал, але вона може бути менш інтегрованою та менш гнучкою, ніж розроблювана система. При цьому інтерфейс BuildingLink, хоча і вміщує велику кількість корисної інформації, є вкрай застарілим на відміну від розроблюваної системи (див. рис. 1.2). Головна сторінка даної веб-системи містить досить невелику кількість інформації, серед якої найбільш корисною можна вважати список заходів запланованих на найближчий час. Зручними є функціонал створення та перегляду заяв, який також буде присутній в розроблюваній системі.

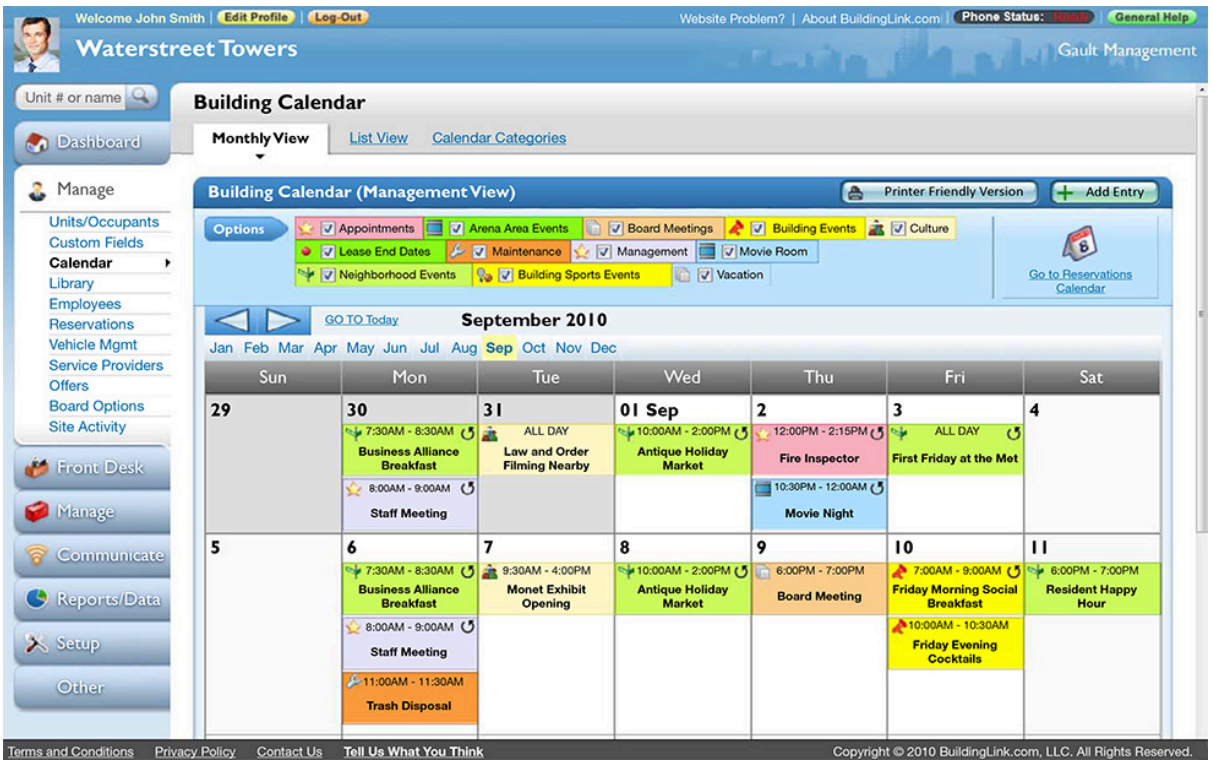


Рисунок 1.2 – Вигляд сторінки заходів в веб-сервісі BuildingLink (за даними [3])

Інший потенційний конкурент на ринку систем для управління нерухомістю це Yardi Voyager (див. рис. 1.3), який пропонує більш загальну платформу для управління нерухомістю, яка охоплює більший спектр нерухомості, а не лише житлові комплекси. Це може зробити її менш спеціалізованою для потреб управління ЖК та не мати такого глибокого розуміння цільової аудиторії.

На відміну від Yardi Voyager розроблювана система буде мати більш зручний та інтуїтивно зрозумілий інтерфейс користувача, який спрощує взаємодію з системою як для адміністраторів, так і для мешканців ЖК. Більше того в розроблюваній веб-системі планується додати більшу кількість функціоналу, наприклад внутрішні чати для мешканців, оголошення, можливість електронного голосування.

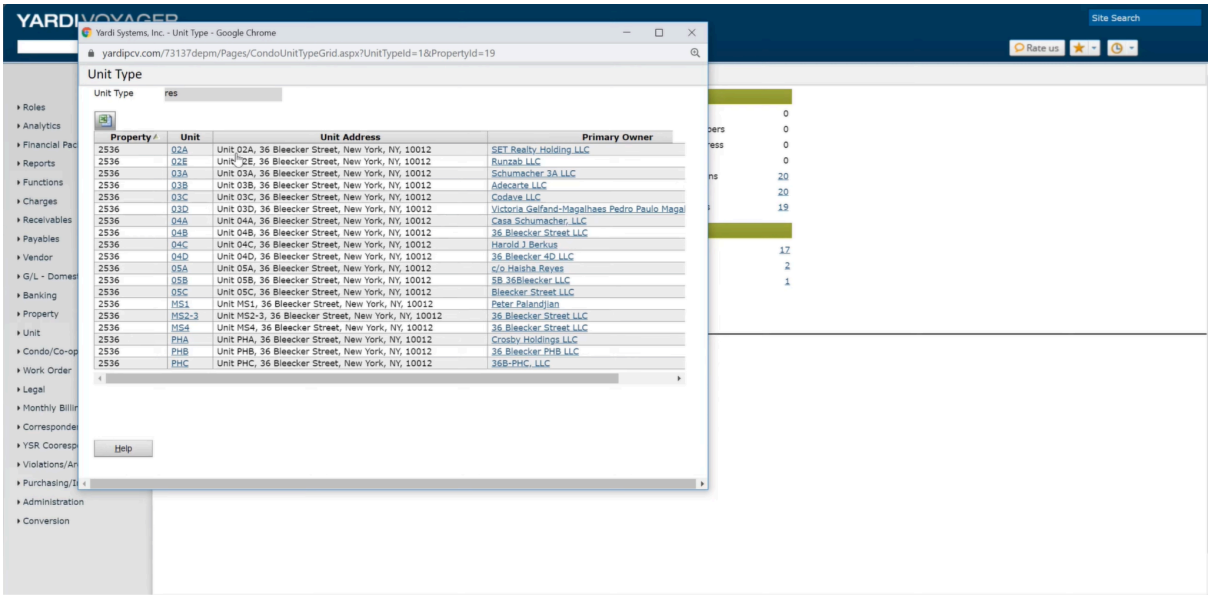


Рисунок 1.3 – Інтерфейс сторінки зі списком мешканців Yardi Voyager (за даними [4])

Розроблювана система вирізняється широким функционалом, гнучкістю, персоналізацією та можливістю інтеграції з великою кількістю житлових комплексів, що робить її привабливим вибором для управління нерухомості. В порівнянні з іншими системами, ми можемо надати більше можливостей та глибше розуміння потреб управління та мешканців.

Розроблювана система буде створюватися з урахуванням потреб сучасних житлових комплексів та їх мешканців. Вона пропонує широкий спектр функцій, які охоплюють всі аспекти управління ЖК, від замовлення послуг до аналітики та звітності. Веб-сервіс надає можливість зробити взаємодію з надавачами комунальних послуг ще зручнішою та ефективною.

1.3 Виявлення проблем та актуалізація рішень

Виявлення проблем та актуалізація рішень для житлових комплексів з точки зору Front-End розробки є ключовим етапом у створенні ефективних і зручних веб-систем. Однією з основних проблем, з якими

стикаються розробники, є забезпечення інтуїтивно зрозумілого інтерфейсу, який задовольняє потреби широкої аудиторії користувачів, включаючи мешканців різного віку та технічного рівня підготовки. Це вимагає ретельного аналізу користувацьких сценаріїв та впровадження принципів UI/UX дизайну, щоб зробити навігацію по системі максимально простою та логічною [5].

По-перше веб-сервіс повинна мати адаптивний дизайн, який автоматично підлаштовується під розміри екрану користувача, забезпечуючи зручне використання на будь-якому пристрої. Це особливо важливо, оскільки сучасні користувачі часто взаємодіють з веб-системами саме через мобільні пристрої.

Веб-сервіс також повинен швидко завантажуватись і реагувати на дії користувачів без значних затримок. Це включає мінімізацію розміру файлів, використання кешування, асинхронне завантаження даних та оптимізацію зображень і скриптів. Висока продуктивність забезпечує позитивний досвід користувачів і підвищує їхню задоволеність системою.

Важливим також є аспект забезпечення високого рівня безпеки даних. Це включає впровадження механізмів автентифікації та авторизації, шифрування даних, захист від XSS та CSRF атак та регулярне оновлення системи безпеки [6]. Надійний захист даних є критично важливим для запобігання несанкціонованому доступу та забезпечення конфіденційності інформації користувачів.

Таким чином, створення Front-End додатку для житлових комплексів охоплює розробку інтуїтивно зрозумілого та адаптивного інтерфейсу, оптимізацію продуктивності, інтеграцію з серверною частиною системи, забезпечення безпеки та доступності. Усі ці аспекти мають бути враховані для створення ефективної, надійної та зручної веб-системи, яка відповідає потребам користувачів і сприяє покращенню якості життя в житлових комплексах.

1.4 Постановка задачі

Основна мета розробки FrontEnd для веб-системи житлових комплексів полягає у створенні інтерактивної та адаптивної веб-системи, яка полегшить взаємодію мешканців з адміністрацією житлового комплексу, покращить комунікацію та спростить доступ до різноманітних послуг. Серед основних задач FrontEnd розробки можна виділити наступні:

— розробка інтерфейсу для модуля замовлення та відстеження послуг. Це включає створення форм реєстрації та відстеження заявок на побутові послуги, такі як ремонт, прибирання, сантехнічні роботи тощо. Інтерфейс повинен забезпечувати можливість створення нових заявок, відстеження статусу їх виконання, а також оцінки якості наданих послуг і збору відгуків від мешканців;

— верстка внутрішніх чатів для мешканців. Це включає розробку форми для створення групових чатів, де мешканці можуть обговорювати різні питання. Інтерфейс повинен підтримувати відображення нових повідомлень в реальному часі для обміну інформацією між мешканцями, включаючи попередження про планові ремонтні роботи або інші події;

— створення сторінки оголошень від адміністрації та мешканців. Це включає форму для створення оголошень адміністрацією, де можна публікувати інформацію про плановані роботи, графіки відключень, загальні збори тощо. Також необхідно розробити інтерфейс для мешканців, щоб вони могли розміщувати свої оголошення або сповіщення, а також переглядати існуючі оголошення, використовуючи фільтрацію та сортування;

— розробка інтерфейсу для оплати комунальних послуг. Це включає створення сторінки, де мешканці можуть переглядати та оплачувати рахунки за комунальні послуги через онлайн-платформу. Інтерфейс також повинен надавати функціонал відображення історії

споживання ресурсів, таких як електроенергія, вода, опалення, для кожного мешканця;

— розробка модуля голосування. Це включає створення інтерфейсу для проведення голосувань серед мешканців щодо важливих питань будинку або житлового комплексу. Інтерфейс повинен забезпечувати можливість анонімного голосування та відображення результатів для всіх зацікавлених сторін;

— верстка інтерфейсу для системи аналітики та звітності. Це включає створення форм для збору та аналізу даних про використання ресурсів, обсягів замовлень послуг, рівня задоволеності мешканців тощо. Інтерфейс повинен забезпечувати візуальне відображення звітів, сформованих на стороні BackEnd, для управляючих компаній та адміністрації житлових комплексів, що допоможе приймати обґрунтовані управлінські рішення;

— не менш важлива задача полягає в оптимізації продуктивності системи. Веб-сайт повинен швидко завантажуватись і реагувати на дії користувачів без затримок. Це включає мінімізацію розміру файлів, використання кешування, асинхронне завантаження даних та оптимізацію зображень і скриптів. Швидка та безперебійна робота системи є ключовим фактором для забезпечення позитивного досвіду користувачів.

Основою реалізації є використання React – популярної бібліотеки для створення інтерфейсів користувача, яка забезпечує високу продуктивність та гнучкість завдяки своїй компонентній архітектурі. Кожен модуль системи буде реалізовано у вигляді окремих компонентів, що сприятиме їх повторному використанню та спрощенню підтримки коду [7].

Для управління станом додатку буде використано MobX, оскільки цей інструмент дозволяє централізовано зберігати та управляти станом додатку, забезпечуючи єдине джерело даних та спрощуючи роботу між

компонентами [8, с. 45]. Це особливо важливо для модулів, що потребують спільного доступу до даних, таких як чат або система голосування.

Інтеграція з зовнішніми сервісами буде здійснюватися через RESTful API. Взаємодія з API буде реалізована за допомогою axios, що дозволяє легко робити запити до сервера та обробляти відповіді. Для забезпечення безпеки даних будуть використовуватися токени для аутентифікації та авторизації користувачів

Для забезпечення високого рівня безпеки буде впроваджено механізми аутентифікації користувачів за допомогою JWT. Це дозволить захистити дані від несанкціонованого доступу та забезпечити безпеку особистої інформації користувачів. Шифрування даних буде здійснюватися як на стороні клієнта, так і на стороні сервера.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Функціональні вимоги в програмній системі для управління житловими комплексами є ключовим елементом, який визначає, як система буде взаємодіяти з користувачами та які функції вона має надавати. Функціональні вимоги для Front-End розробки веб-системи житлових комплексів:

а) аутентифікація та авторизація;

1) реєстрація та вхід у систему для користувачів з різними ролями, такими як мешканець або адміністратор. Користувачі повинні мати можливість створювати облікові записи, входити до системи, використовуючи свої облікові дані, і відновлювати пароль у разі його втрати;

2) в залежності від ролей користувачів, інтерфейс повинен адаптуватися під різні ролі, забезпечуючи доступ до певних функцій лише для уповноважених користувачів, наприклад адміністратор повинен мати можливість створення оголошень, чатів, тощо;

б) модуль замовлення та відстеження послуг;

1) інтерфейс повинен дозволяти користувачам легко подавати заявки на послуги, переглядати статус їх виконання, підтримувати оновлення статусу в реальному часі;

2) реалізація системи сповіщень через електронну пошту або push-повідомлення;

в) внутрішні чати для мешканців;

1) інтерфейс для адміністратора повинен підтримувати створення групових чатів, додавання та видалення учасників, а для мешканців повинен бути розроблений функціонал відправки та отримання повідомлень без необхідності оновлення сторінки;

г) модуль оголошень;

1) інтерфейс повинен дозволяти адміністратору створювати, редагувати та видаляти оголошення, а мешканцям давати змогу фільтрувати та переглядати їх;

2) необхідно також підтримувати функціональність для залишення коментарів на оголошення від мешканців;

д) система комунальних показників та фінансова звітність;

1) мешканці повинні мати доступ до історії своїх платежів і можливість завантажувати квитанції;

е) модуль голосування;

1) інтерфейс повинен підтримувати створення опитувань, голосування та відображення результатів для всіх користувачів.

2.2 Нефункціональні вимоги

До нефункціональних вимог веб-сервісу для Front-End розробки можна віднести:

а) продуктивність;

1) забезпечення оптимізованої швидкості реакції системи на будь-які дії користувачів, щоб забезпечити плавний та миттєвий досвід взаємодії, відображення будь-якої з сторінок не має займати більше 200 мс;

2) максимізація швидкості завантаження та відгуку інтерфейсу, щоб уникнути довгого очікування користувачів;

б) безпека;

1) забезпечення надійного збереження та передачі особистих даних, використовуючи шифрування та інші методи захисту відповідно до стандарту GDPR;

2) захист від потенційних загроз, таких як XSS, CSRF, SQL-ін'єкції та інші види атак, для забезпечення безпеки та цілісності даних;

в) масштабованість;

1) забезпечення гнучкої архітектури, яка може масштабуватися без зниження продуктивності при збільшенні обсягу даних та користувачів за допомогою використання компонентної архітектури;

г) надійність;

1) мінімізація можливих ситуацій відмов та перерв у роботі системи для забезпечення безперебійного функціонування, доступність FrontEnd на рівні 99% часу;

д) підтримка;

1) Забезпечення ефективної підтримки користувачів, включаючи швидке вирішення проблем та відповіді на запити;

е) кросплатформеність;

1) забезпечення сумісності із різними веб-браузерами та операційними системами для максимальної доступності користувачів, наприклад Windows чи MacOS та Google Chrome чи Safari;

2.3 Вимоги до середовища розгортання

Вимоги до середовища розгортання веб-системи з використанням React передбачають створення масштабованої, безпечної та високопродуктивної інфраструктури. FrontEnd частина веб-сервісу повинна бути розгорнута на Azure, що забезпечує легке управління та масштабування додатку. Використання Azure Storage для зберігання статичних файлів, таких як зображення та документи, дозволить знизити навантаження на основний сервер, покращити продуктивність та зменшити

час завантаження цих файлів для користувачів. Захист даних забезпечується за допомогою Azure Key Vault для зберігання конфіденційної інформації, також використання Azure допомагає забезпечити захист від основних видів атак, включаючи CSRF та XSS. Крім того, слід налаштувати моніторинг і логування через Azure Monitor та Application Insights, що дозволить відстежувати продуктивність додатку та вчасно реагувати на будь-які проблеми. Для забезпечення безперебійної роботи та швидкого завантаження контенту використовується Azure Content Delivery Network (CDN). Всі ці компоненти разом створюють надійне та ефективне середовище для розгортання та підтримки веб-системи на базі React і Azure.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗ

Метою проектування FrontEnd частини веб-сервісу є розкрити складність системи, зокрема визначити важливість різних компонентів та їх взаємозв'язки. Це допомагає виявити можливі проблеми або вузькі місця у функціонуванні системи та спрямувати увагу розробника на ключові аспекти для подальшого вдосконалення. З точки зору FrontEnd розробки веб-сервісу для житлових комплексів, на меті стоїть детально описати, як реалізуються функціональні вимоги, а також враховуються нефункціональні аспекти, такі як продуктивність, безпека та масштабованість.

Починати функціональне моделювання необхідно з визначення основних модулів системи, кожен з яких відповідає за певну функціональність. Одним з ключових модулів, від якого залежить робота сервісу, є модуль аутентифікації та авторизації. Він включає реєстрацію та вхід до системи для користувачів з різними ролями, такими як мешканець або адміністратор. Контроль доступу до функціональності системи здійснюється на основі ролей користувачів, що забезпечує безпеку та захист даних від несанкціонованого доступу.

Іншим важливим модулем майбутнього веб-сервісу є модуль замовлення та відстеження послуг, який дозволяє користувачам створювати, переглядати та відстежувати статус заявок на різні побутові послуги. Реалізація цього модуля включає інтерактивні форми для подання заявок та систему нотифікацій, яка інформує користувачів про зміни статусу їхніх заявок. Цей модуль підвищує зручність та ефективність обслуговування мешканців житлових комплексів.

В свою чергу внутрішні чати для мешканців забезпечують можливість створення та управління чатами для обговорення питань між мешканцями певного комплексу, будинку чи під'їзду. Цей функціонал реалізується через окремий інтерфейс для створення чатів та систему повідомлень, що дозволяє мешканцям обмінюватися інформацією у реальному часі.

Модуль оголошень включає публікацію та перегляд оголошень від адміністрації житлового комплексу та мешканців. Користувачі можуть коментувати та реагувати на оголошення, що сприяє активній взаємодії та обміну інформацією. Це важливий аспект комунікації у спільноті мешканців. Також важливою частиною його функціоналу є можливість проведення голосування дозволяє проводити голосування серед мешканців щодо важливих питань, таких як ремонт, обслуговування будинку або вибір постачальників послуг. Інтерфейс цього модуля забезпечує можливість анонімного голосування та відображення результатів для всіх зацікавлених сторін.

Детальний опис взаємодії різних користувачів з веб-сервісом можна побачити на рисунку 3.1.

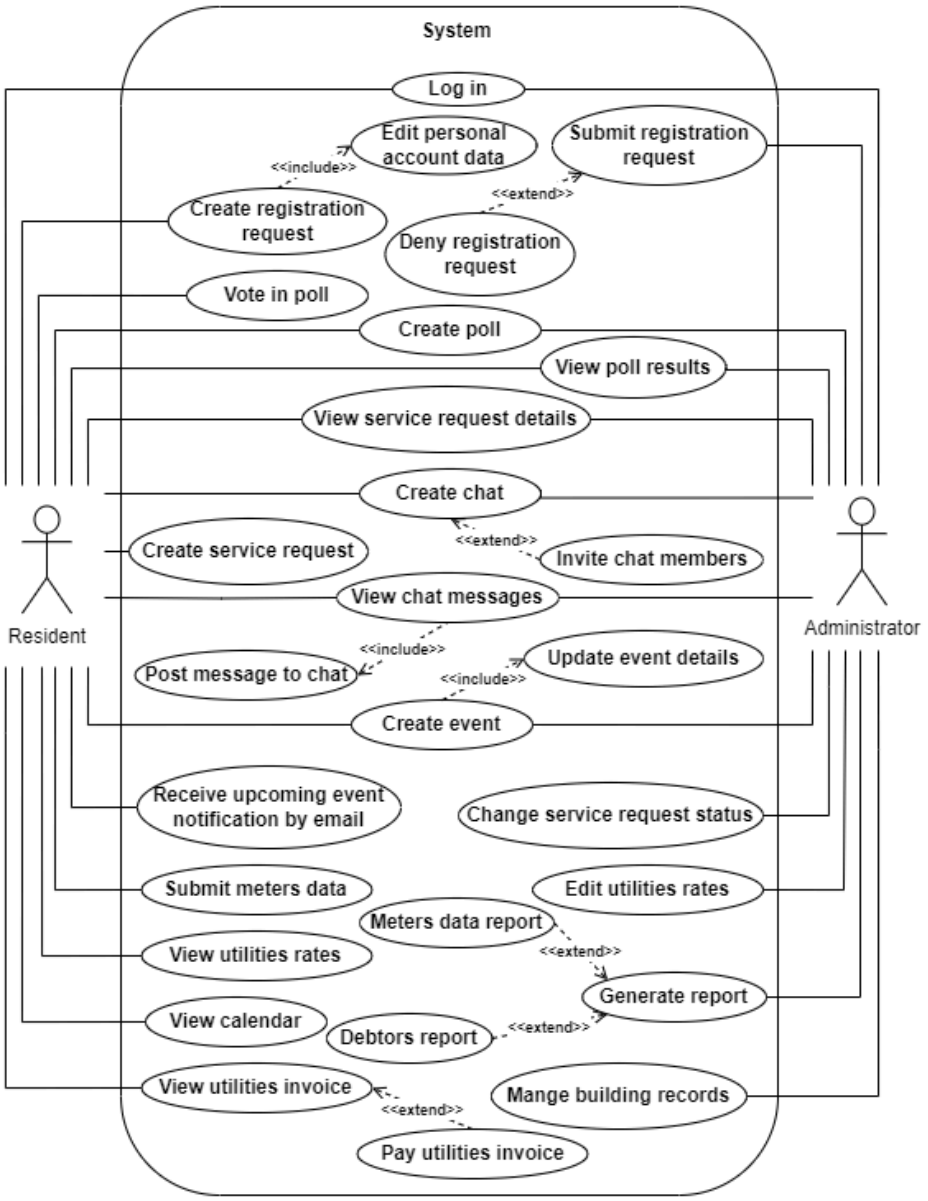


Рисунок 3.1 – Use Case діаграма (виконаний самостійно)

Таким чином, згідно із зазначеними раніше функціональними вимогами, були визначені та проілюстровані основні функції веб-сервісу для житлових комплексів.

Для розробки FrontEnd частини веб-сервісу також важливо розробити діаграма класів, відображає структуру класів системи управління будинками та їхні взаємозв'язки. Вона допомагає розібратися у структурі системи та визначити ролі кожного класу у процесі управління

будинками. З розширеним розумінням цих компонентів ми можемо ефективніше розробляти, вдосконалювати та підтримувати систему для забезпечення найкращого досвіду для мешканців будинку та легкості управління для адміністраторів. Діаграма включає в себе різні типи діяльностей, які відбуваються у будинку, а також засоби для спілкування та управління ресурсами. Вона є важливим інструментом для розуміння структури системи та відображення її основних компонентів, до яких відносяться:

— Activity (діяльність) - цей клас відображає різноманітні діяльності, що відбуваються у системі управління будинками. Він може представляти анонси про події, чати для спілкування між мешканцями будинку, опитування для збору думок та багато іншого;

— AppUser (користувач додатку) - цей клас представляє користувачів системи управління будинками. Він містить інформацію про кожного користувача, включаючи ім'я, адресу електронної пошти, пароль та інші персональні дані;

— Announcement (оголошення) - цей клас є підтипом діяльностей та містить додаткові поля для зберігання інформації про оголошення, які розсилаються мешканцям будинку;

— Chat (чат) - цей клас також є підтипом діяльностей і представляє чати, в яких мешканці можуть спілкуватися один з одним в реальному часі;

— Comment (коментар) - цей клас представляє коментарі до різних діяльностей у системі. Він містить текст коментаря, а також інформацію про автора та час створення коментаря;

— Invoice (рахунок) - цей клас містить інформацію про суму платежу, дату виставлення рахунка та інші деталі;

— MeterData (дані лічильників) - цей клас зберігає дані з лічильників, що використовуються для обліку споживаної енергії, води або інших ресурсів у квартирах;

— Poll (опитування) - цей клас представляє опитування, які проводяться серед мешканців будинку;

— PollOption (варіант опитування) - цей клас містить варіанти відповідей для опитувань. Користувачі можуть вибрати один з цих варіантів у своїх відповідях;

— RegistrationRequest (запит на реєстрацію) - цей клас представляє запити від осіб, які бажають зареєструватися у системі управління будинками;

— ResidentBuilding (будинок з квартирами) - цей клас містить інформацію про будинки, в яких розташовані квартири. Він може включати дані про адресу, кількість квартир тощо;

— ServiceOrder (замовлення на обслуговування) - цей клас представляє замовлення на різні послуги або обслуговування, які надаються мешканцям будинку;

— Photo (фотографія) - цей клас використовується для зберігання фотографій, які можуть бути пов'язані з різними сутностями у системі;

— ActivityAttendee (учасник діяльності) - цей клас представляє відносини між діяльностями та учасниками, тобто користувачами, які беруть участь у певній діяльності;

— PollOptionVote (голос в опитуванні) - цей клас містить інформацію про голоси користувачів за варіанти в опитуваннях;

— Apartment (Квартира) - цей клас відображає квартири, що розташовані у будинках. Він містить дані про номер квартири, площу, кількість кімнат тощо.

На основі описаних вище сутностей була побудована діаграма класів (див. рис. 3.2), згідно із якою буде проектуватися доменний шар веб-сервісу «Житловий комплекс».

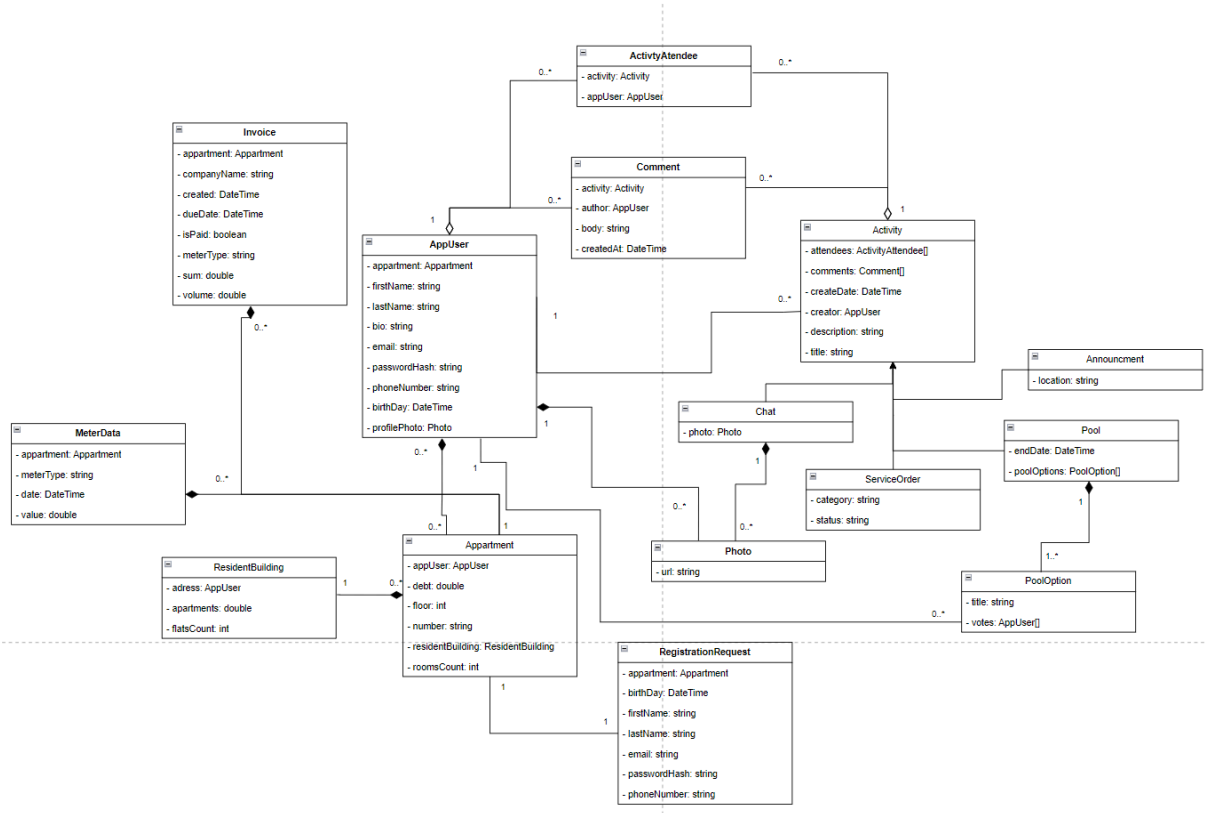


Рисунок 3.2 – UML діаграма класів (виконаний самостійно)

Розроблена діаграма класів допомагає команді розробників, адміністраторам та користувачам краще зрозуміти роль кожного елемента та його внесок у загальну функціональність системи управління будинками.

3.2 Проектування архітектури ПЗ

Проектування архітектури програмного забезпечення для Front-End частини веб-системи є комплексним процесом, що включає кілька

ключових аспектів, спрямованих на створення ефективного, масштабованого та зручного у підтримці додатку.

Основою архітектури є використання компонентного підходу, який надає React. У цій архітектурі додаток розбивається на дрібні, ізольовані та повторно використовувані компоненти, кожен з яких відповідає за певну частину інтерфейсу користувача. Компоненти можуть бути як простими, так і складеними, утворюючи дерево компонентів, яке відображає структуру додатку. В процесі реалізації FrontEnd також важливо використати наступні патерни:

— Module, який дозволяє організувати код у логічні частини або модулі, які можуть бути незалежно розроблені, протестовані та підтримувані. Це сприяє модульності додатка, спрощує управління залежностями та підвищує зрозумілість коду;

— Factory забезпечує створення нових об'єктів без прямого використання відповідних операторів, що дозволяє приховати логіку створення об'єктів і забезпечити більшу гнучкість. Використання фабрики допомагає реалізувати принципи інверсії залежностей та зменшити зв'язність між класами.

Для управління станом додатку буде використовуватися MobX, який дозволяє централізовано зберігати та управляти станом додатку. Це забезпечує передбачуваність і контрольованість змін стану, полегшуючи налагодження та тестування додатку. MobX Middleware можуть бути використані для обробки асинхронних операцій, таких як запити до API.

Маршрутизація здійснюється за допомогою бібліотеки React Router, яка дозволяє створювати динамічні, односторінкові додатки з підтримкою навігації без необхідності для перезавантаження сторінки. React Router

дозволяє легко керувати маршрутами та забезпечує плавний перехід між різними розділами додатку.

Для забезпечення продуктивності та оптимізації завантаження використовуються такі інструменти, як Webpack. Ці інструменти збирають, мінімізують та оптимізують ресурси, забезпечуючи швидке завантаження сторінок.

З точки зору безпеки, важливо використовувати захищені протоколи зв'язку, такі як HTTPS, і забезпечити захист від поширених атак, таких як XSS та CSRF. Інструменти для управління аутентифікацією та авторизацією, такі як JWT або OAuth, забезпечують безпечний доступ до ресурсу.

Для забезпечення адаптивності та кросплатформеності додатку використовується CSS-in-JS бібліотеки, такі як Bootstrap, які дозволяють писати CSS безпосередньо в TypeScript і створювати стильові правила, що залежать від стану компонентів [9]. Це дозволяє легко створювати адаптивні інтерфейси, що коректно відображаються на різних пристроях і розмірах екрану.

Не менш важливим є моніторинг та логування за допомогою спеціалізованих інструментів, які дозволяють відстежувати помилки та поведінку користувачів у реальному часі. Це забезпечує швидке виявлення та виправлення проблем, підвищуючи надійність додатку.

У підсумку, архітектура Front-End частини веб-системи, побудована з використанням React та суміжних технологій, забезпечує високу продуктивність, безпеку та масштабованість, дозволяючи створювати сучасні, інтерактивні та зручні у використанні веб-додатки. Такий підхід

забезпечує гнучкість у розробці та підтримці, що дозволяє швидко адаптувати додаток до змінних вимог користувачів і бізнесу.

3.3 UI/UX дизайн системи

3.3.1 Візуалізація шляху користувача через інтерфейс системи

UI/UX дизайн є ключовим аспектом розробки програмного забезпечення, оскільки він безпосередньо впливає на досвід користувача. UI (User Interface) дизайн зосереджений на візуальній складовій продукту, забезпечуючи естетично привабливий та інтуїтивно зрозумілий інтерфейс. UX (User Experience) дизайн, у свою чергу, охоплює всі аспекти взаємодії користувача з продуктом, включаючи ефективність, доступність та зручність використання. Разом ці два напрямки допомагають створювати продукти, які не тільки виглядають добре, але й забезпечують позитивний досвід користувача, що є вирішальним для успіху будь-якого ПЗ.

Одним з перших етапів UI/UX проектування є побудова User flow діаграми. User flow діаграма є ключовим інструментом у процесі проектування програмного забезпечення, оскільки вона дозволяє візуалізувати та оптимізувати шлях користувача через інтерфейс продукту [10]. Ця діаграма допомагає командам розробників та дизайнерів зрозуміти послідовність дій, які користувач виконує для досягнення певної мети, наприклад, оформлення замовлення або налаштування параметрів системи. Використання user flow сприяє створенню інтуїтивно зрозумілого та ефективного користувацького інтерфейсу, що, в свою чергу, може підвищити задоволеність користувачів та конверсію. Крім того, така діаграма допомагає уникнути потенційних проблем з навігацією та взаємодією, які можуть виникнути під час розробки, та забезпечує більш ефективне спілкування між членами команди.

Нижче представлена User Flow діаграма, побудована для веб-сервісу «Житловий комплекс» з урахуванням усіх потреб користувачів (див. рис. 3.3).

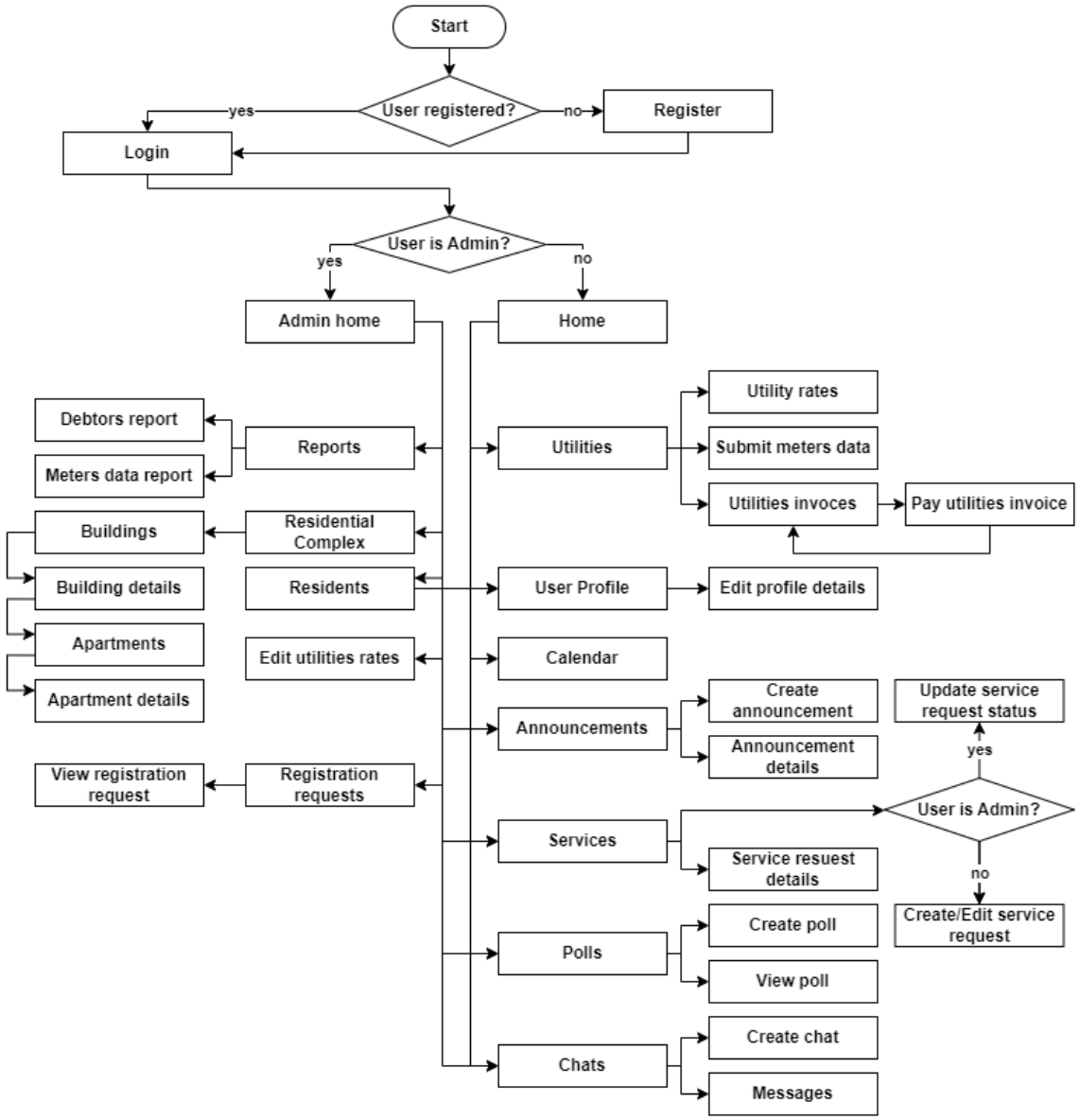


Рисунок 3.3 – User Flow діаграма (виконаний самостійно)

На основі побудованої User Flow діаграми можна створити детальніші wireflows, які включають wireframes або схематичні зображення інтерфейсу, або ж перетворити їх у task flows для окремих завдань користувача.

3.3.2 Моделювання користувацького інтерфейсу

Wireframe діаграма є важливим інструментом в процесі розробки веб-сайтів, мобільних додатків та інших інтерактивних продуктів. Вона представляє собою набір простих схематичних малюнків, які відображають структуру та елементи користувальницького інтерфейсу, такі як кнопки, поля введення, меню і т. д., без детального дизайну (див. рис. 3.4).

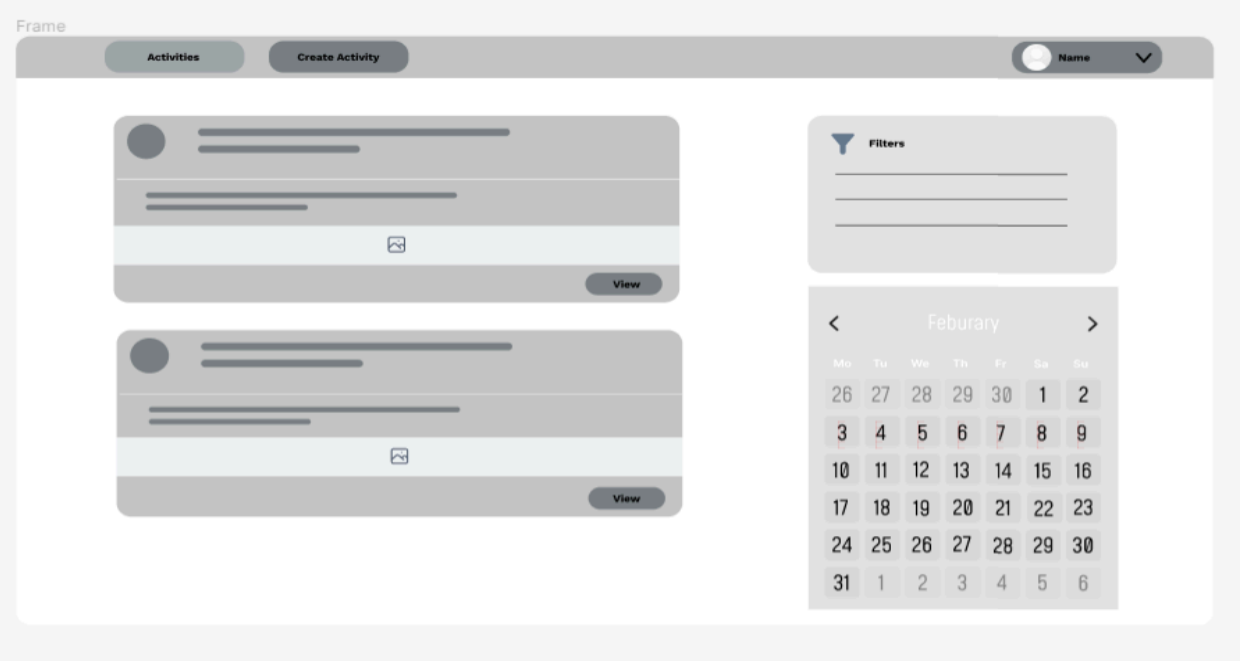


Рисунок 3.4 – Wireframe головне меню (виконаний самостійно)

Wireframe діаграма дозволяє команді розробників і дизайнерів швидко створити прототип інтерфейсу та визначити його структуру, розміщення елементів та навігацію (див. рис. 3.5). Вона допомагає уточнити вимоги до продукту, визначити пріоритети функціональності та забезпечити розуміння між учасниками проекту щодо його концепції [11].

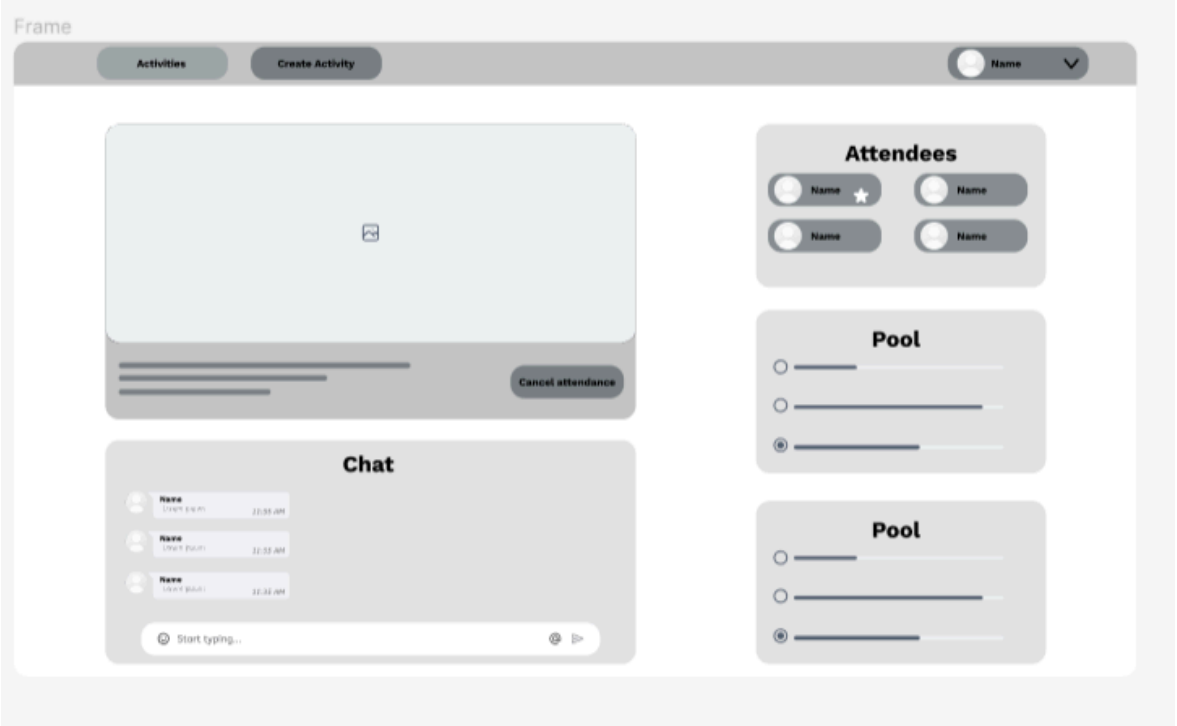


Рисунок 3.5 – Wireframe меню активності (виконаний самостійно)

Створені Wireframe діаграми (див. рис. 3.6) будуть використовуватися в процесі розробки, оскільки вони можуть виступати в якості посібника для розробників, забезпечуючи чітке розуміння того, які компоненти буде необхідно створити та як вони повинні взаємодіяти між собою з боку FrontEnd. Це сприяє більш структурованому підходу до написання коду та знижує ймовірність виникнення непередбачених проблем під час інтеграції різних частин системи.

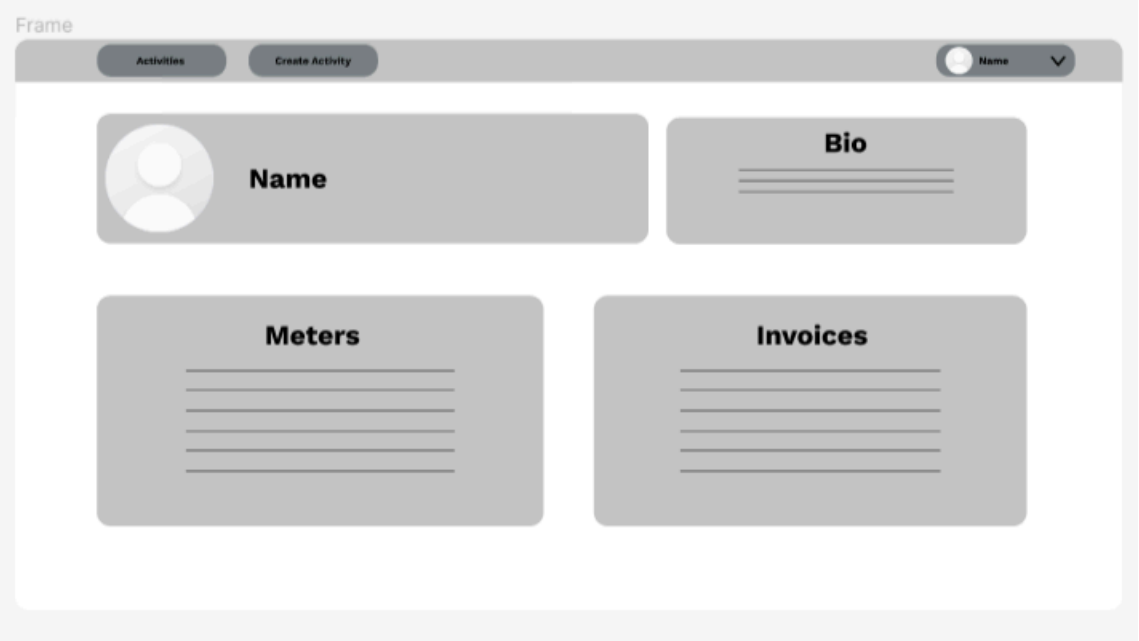


Рисунок 3.6 – Wireframe меню користувача (виконаний самостійно)

Крім того, wireframe діаграми сприяють комунікації між різними учасниками проекту. Вони надають всім зацікавленим сторонам, включаючи замовників, розробників, дизайнерів і користувачів, загальне уявлення про те, як буде виглядати та функціонувати система. Це дозволяє отримати зворотний зв’язок та погодити ключові аспекти проєкту ще до початку розробки, що забезпечує узгодженість та задоволення вимог всіх сторін.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис процесу розгортання та моніторингу сервісу

Для розробки FrontEnd частини веб-сервісу для житлових комплексів було обрано фреймворк React та платформу Azure. Вибір цих технологій обумовлений їхньою популярністю, широкими можливостями для створення інтерактивних інтерфейсів користувача, а також забезпеченням високої продуктивності та надійності.

Основним фреймворком для розробки інтерфейсу користувача є React, який є бібліотекою для побудови користувацьких інтерфейсів, надає можливість створювати компоненти, що можуть бути легко повторно використані та управляти станом додатку ефективно. Однією з ключових переваг React є використання віртуального DOM, який значно покращує продуктивність додатку, дозволяючи швидко оновлювати тільки ті частини інтерфейсу, що змінилися. Це забезпечує швидку реакцію додатку на дії користувачів та покращує загальний досвід використання.

Компонентна архітектура React дозволяє розбити інтерфейс на окремі компоненти, кожен з яких відповідає за свою частину функціональності., що забезпечує модульність та зручність в обслуговуванні коду. Кожен компонент має свій життєвий цикл, що дозволяє контролювати процеси ініціалізації, оновлення та знищення компонентів. Наприклад, для реалізації модуля аутентифікації та авторизації було розроблено компоненти Login (див. рис. 4.1), Register (див. рис. 4.2) та UserProfile, які відповідають за відповідні частини функціональності. Ці компоненти взаємодіють з бекендом через API-запити, обробляючи отримані дані та відображаючи їх користувачам.

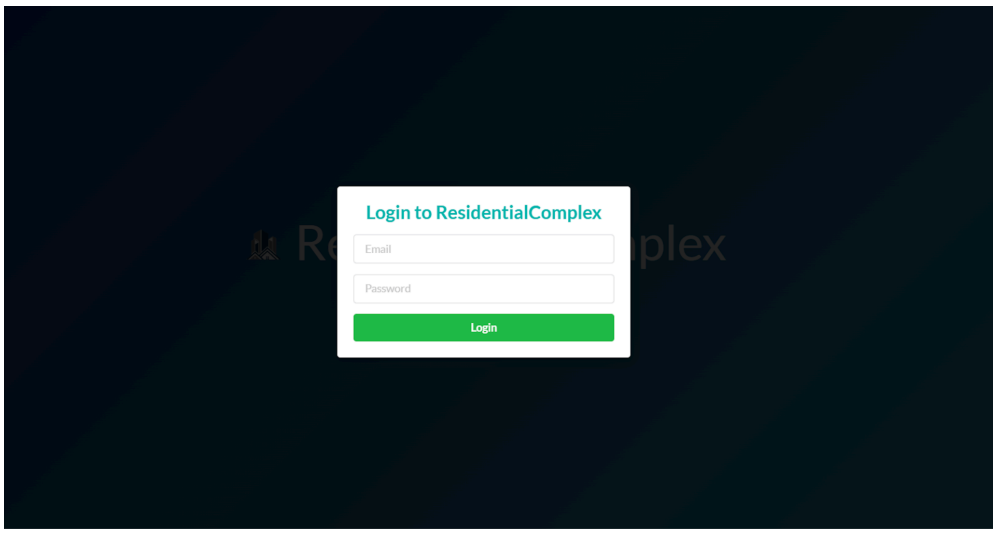


Рисунок 4.1 – Компонент Login для авторизации в веб-сервісі (виконаний самостійно)

Для управління станом додатку ми використовуємо бібліотеку MobX, яка дозволяє централізовано зберігати стан додатку і забезпечує передбачувану поведінку компонентів. MobX дозволяє легко відстежувати зміни стану, дебажити та тестувати додаток. Наприклад, статус користувача зберігається в стані MobX, що дозволяє компонентам, які залежать від цього стану, автоматично оновлюватися при його зміні.

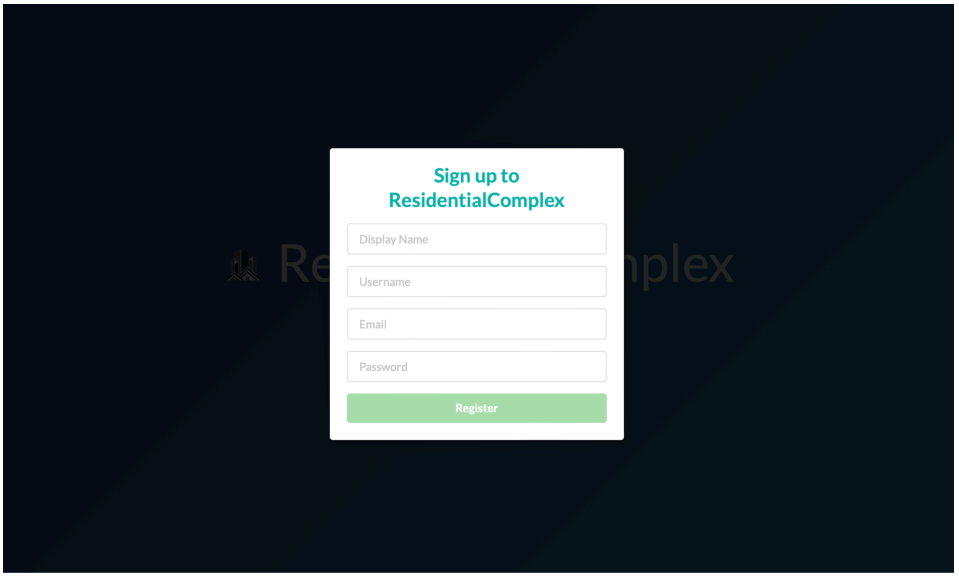


Рисунок 4.2 – Сторінка реєстрації користувача (виконаний самостійно)

Використання MobX також забезпечує масштабованість додатку, дозволяючи легко додавати нові функціональні можливості без значних змін у кодовій базі. В реалізації веб-сервісу також використано фреймворк axios дозволяє реалізувати асинхронні дії, такі як виклики до API або обробку побічних ефектів, що покращує архітектуру додатку та робить її більш стійкою до змін.

Для забезпечення кросплатформності та адаптивності інтерфейсу були використані спеціалізовані CSS-фреймворки, а також власні стилі. Це дозволяє створювати інтерфейси, що коректно відображаються на різних пристроях та екранах різної роздільної здатності. Використання адаптивного дизайну гарантує, що користувачі зможуть зручно користуватися системою як на десктопах, так і на мобільних пристроях. Крім того, використання CSS Grid та Flexbox забезпечує створення складних макетів, які адаптуються під різні розміри екранів, що покращує зручність користування та загальний вигляд додатку (див. рис. 4.3).

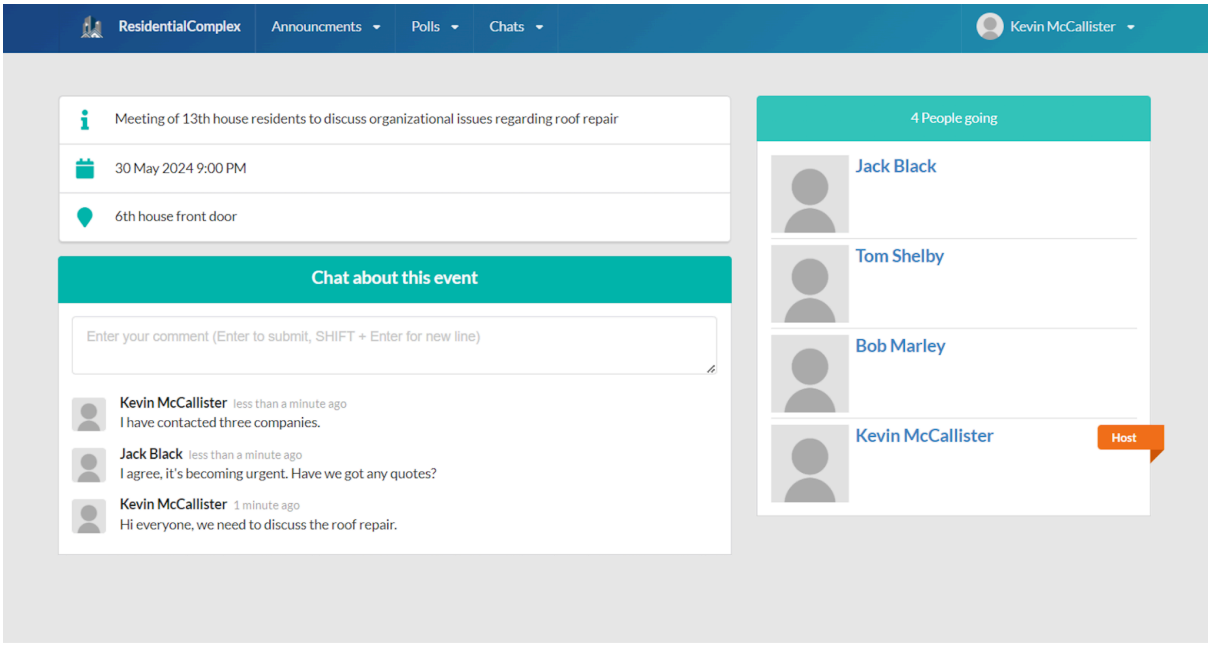


Рисунок 4.3 – Сторінка запланованого заходу, розроблена з використанням grid та flexbox (виконаний самостійно)

Важливою складовою розробки є забезпечення безпеки фронтенд-додатку. Для цього в веб-сервісі використовуються різноманітні механізми захисту, такі як шифрування даних, переданих між клієнтом і сервером, використовуючи HTTPS. Також було впроваджено захист від CSRF та XSS атак, використовуючи відповідні політики безпеки та валідацію даних на клієнтській стороні. Крім того, аутентифікація користувачів реалізована за допомогою токенів, що ускладнює доступ до персональних даних користувачів потенційним зловмисника.

Розроблений функціонал FrontEnd частини веб-сервісу для житлових комплексів відповідає всім вимогам та включає наступні основні модулі та компоненти, реалізовані за допомогою фреймворку React та платформи Azure.

Першим розробленим модулем є модуль замовлення та відстеження послуг. До нього входять компоненти для створення, перегляду та відстеження статусу заявок на побутові послуги. Користувачі можуть створювати нові заявки за допомогою компонента Request, який містить форму для введення деталей заявки. Інтерфейс веб-сервісу також включає функціонал відображення списку заявок користувача та їхній статус, дозволяючи легко відстежувати прогрес виконання послуг.

Другим важливим модулем є модуль оголошень. Компонент Announcements (див. рис. 4.4) дозволяє адміністрації житлового комплексу та мешканцям публікувати оголошення, які можуть переглядати всі користувачі системи. Користувачі можуть коментувати усі оголошення, що підвищує рівень взаємодії між мешканцями.

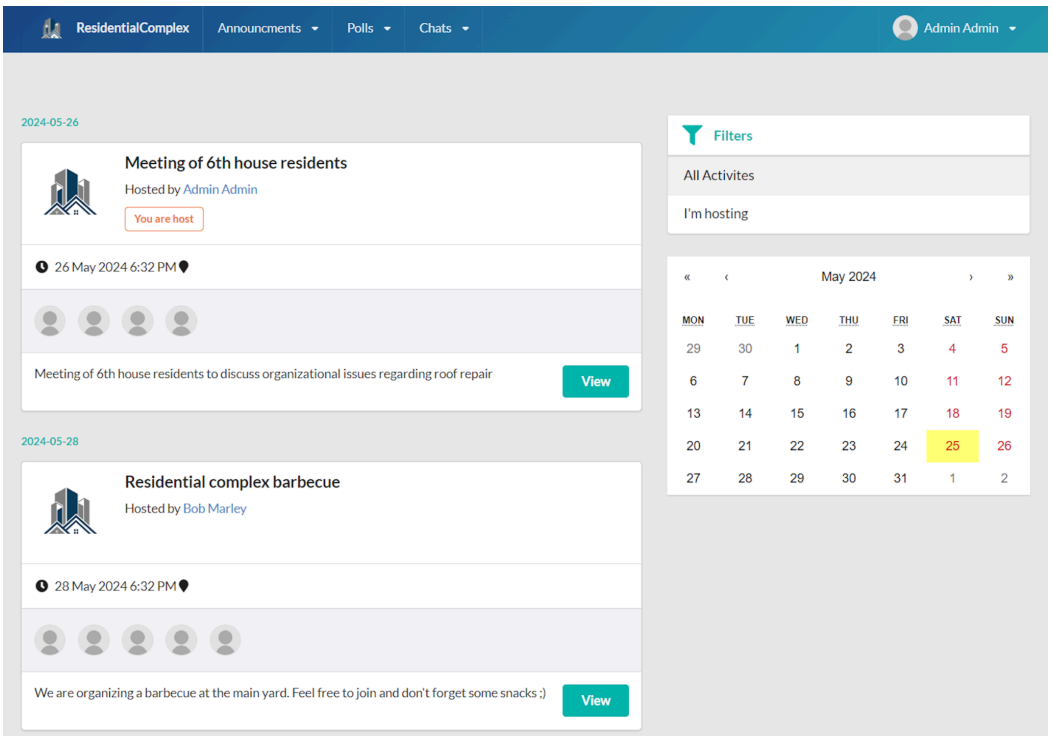


Рисунок 4.4 – Сторінка перегляду оголошень (виконаний самостійно)

Форма для створення оголошень проста та інтуїтивно зрозуміла (див. рис. 4.5), що дозволяє користувачам швидко та легко публікувати інформацію. Для створення нового оголошення необхідно ввести наступну інформацію:

— назва оголошення;
— детальний опис;
— список учасників;
— дату та час запланованого заходу;
— деталі, щодо місця проведення.

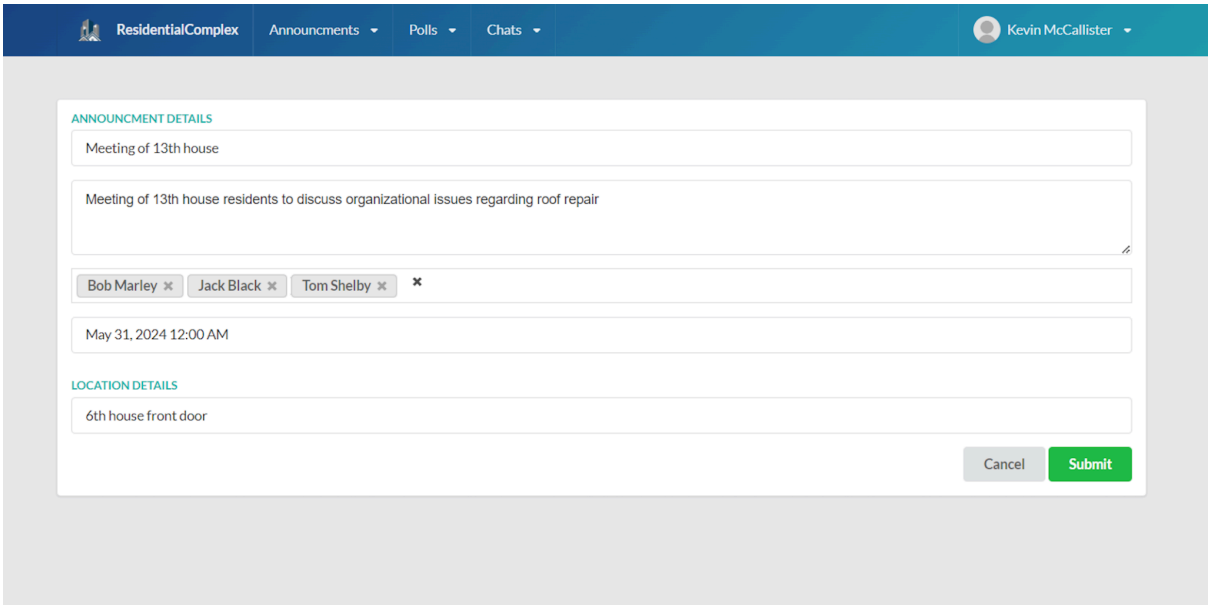


Рисунок 4.5 – Форма для створення оголошення (виконаний самостійно)

До сервісу оголошень також відноситься компонент Poll, який дозволяє проводити голосування серед мешканців з важливих питань (див. рис. 4.6), таких як ремонт, обслуговування будинку або вибір постачальників послуг. Користувачі можуть переглядати активні голосування, брати участь у них та переглядати результати. Інтерфейс для створення голосувань забезпечує адміністрації можливість легко організовувати опитування та збирати думки мешканців.

Іншою важливою складовою веб-сервісу є чати для мешканців. В процесі роботи над проектом було розробили компонент Chat, який дозволяє користувачам створювати та керувати чатами для обговорення питань між мешканцями певного будинку або під'їзду. Цей компонент підтримує основні функції обміну повідомленнями в реальному часі, такі як надсилання та отримання текстових повідомлень, а також підтримку групових чатів.

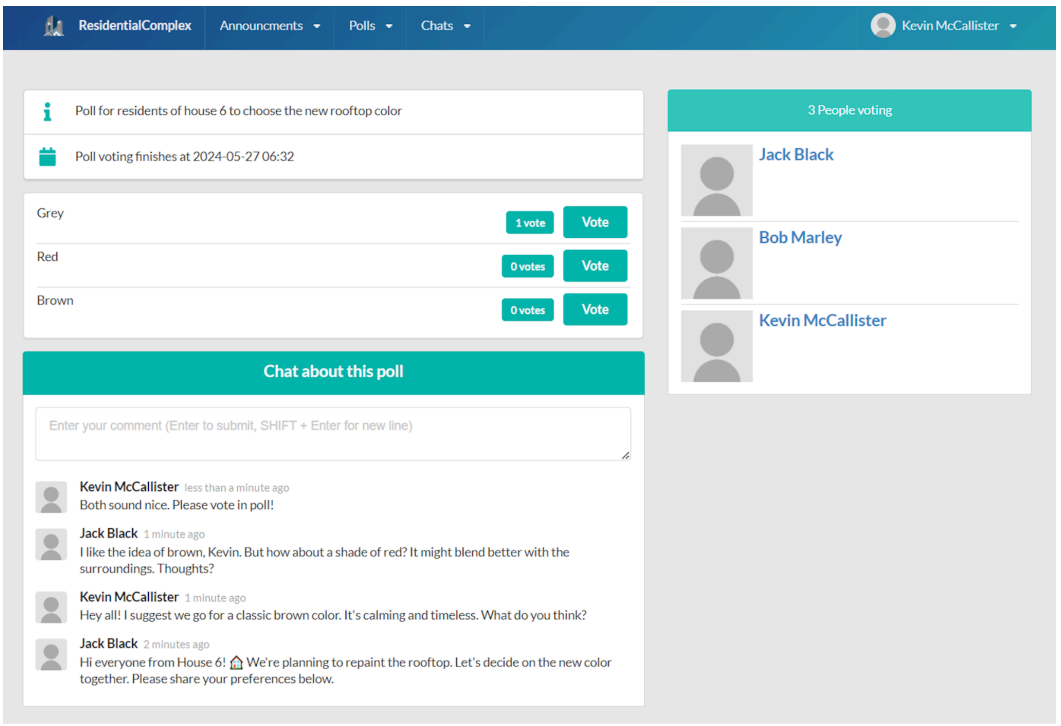


Рисунок 4.6 – Вигляд сторінки опитування (виконаний самостійно)

Реалізація чатів (див. рис. 4.7) забезпечена за допомогою технології SignalR, що дозволяє досягти реального часу обміну повідомленнями з мінімальною затримкою [12].

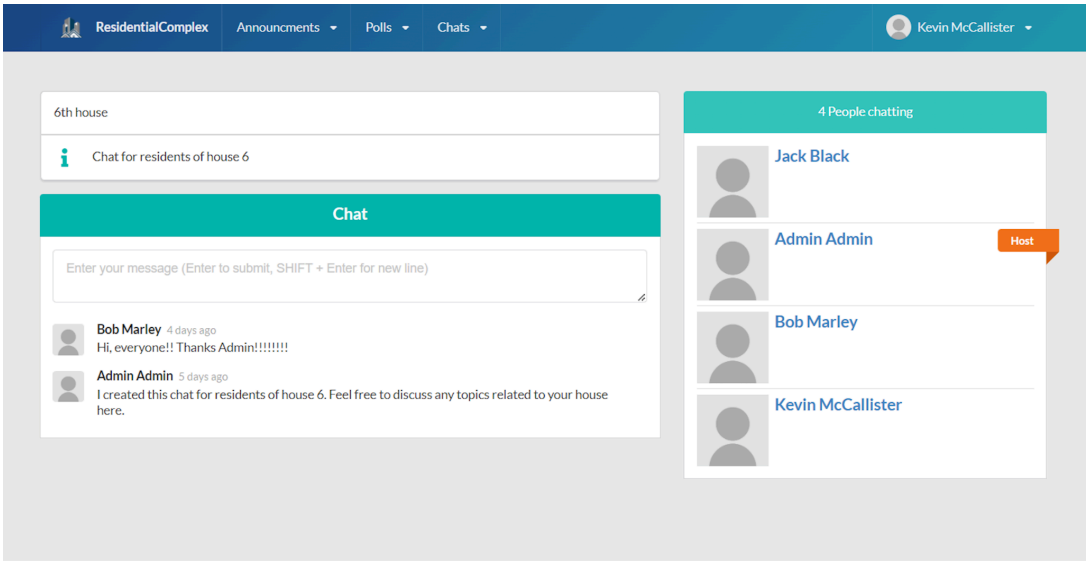


Рисунок 4.7 – Інтерфейс внутрішнього чату будинку (виконаний самостійно)

Таким чином, розроблений функціонал веб-сервісу для житлових комплексів відповідає всім вимогам та забезпечує зручність, безпеку і продуктивність використання. Прийняті програмні рішення в процесі розробки FrontEnd, що базуються на фреймворку React та платформі Azure, забезпечують створення високоякісного, продуктивного та безпечного веб-додатку. Використання сучасних технологій та інструментів дозволяє забезпечити зручність користувачів, високу продуктивність додатку та ефективне управління розробкою та розгортанням системи. Це створює надійну основу для забезпечення всіх необхідних функціональних можливостей та забезпечує гнучкість і масштабованість веб-сервісу для житлових комплексів.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування розробленого програмного забезпечення

Тестування веб-сервісу для житлових комплексів з точки зору FrontEnd розробки є складним і багатоетапним процесом, який забезпечує високий рівень якості, зручності використання, сумісності, продуктивності та безпеки системи. Процес тестування має включати в себе функціональне тестування, тестування зручності користування, кросбраузерне та кросплатформне тестування, тестування продуктивності та безпеки, а також інші аспекти, які забезпечують надійну роботу інтерфейсу користувача.

Функціональне тестування можна вважати основним етапів, що перевіряє, чи правильно працюють всі елементи інтерфейсу відповідно до функціональних та нефункціональних вимог. Це включає перевірку наступних складових інтерфейсу веб-сервісу:

— форм реєстрації та входу в систему;

— перевірку відображення різних елементів сторінок для користувачів відповідно до їх ролі;

— функціонал, пов'язаний з замовленням та відстеження послуг;

— коректне відображення в реальному часі та додавання нових повідомлень до чатів;

— система оголошень, що також включає в себе можливість додавання коментарів та голосувань.

Кожен з цих елементів має бути протестований як вручну, так і за допомогою автоматизованих тестів (див. рис. 5.1). Автоматизовані тести дозволяють перевірити, чи правильно працюють інтерактивні елементи, чи

відповідають дані в формах, чи коректно відображаються повідомлення про помилки та інші аспекти функціональності. Автоматизація тестування значно скорочує час, необхідний для перевірки кожної зміни в коді, і забезпечує стабільність роботи системи при додаванні нових функцій.

```
import CommentStore from './CommentStore';
import { store } from '../mocks/store';
import { HubConnectionBuilder } from '@microsoft/signalr';

jest.mock('@microsoft/signalr');
jest.mock('../mocks/store');

describe('CommentStore', () => {
    let commentStore;

    beforeEach(() => {
        commentStore = new CommentStore();
    });

    it('should create hub connection', async () => {
        commentStore.createHubConnection('1');
        expect(HubConnectionBuilder.prototype.withUrl).toHaveBeenCalledWith(${process.env.REACT_APP_CHAT_URL}?activityId=1, { accessTokenFactory: expect.any(Function) });
        expect(HubConnectionBuilder.prototype.withAutomaticReconnect).toHaveBeenCalled();
        expect(HubConnectionBuilder.prototype.configureLogging).toHaveBeenCalledWith('Information');
        expect(commentStore.hubConnection.start).toHaveBeenCalled();
    });

    it('should stop hub connection', async () => {
        commentStore.createHubConnection('1');
        await commentStore.stopHubConnection();
        expect(commentStore.hubConnection.stop).toHaveBeenCalled();
    });

    it('should clear comments and stop connection', async () => {
        commentStore.comments = [{ id: 1, body: 'Test comment', createdAt: new Date() }];
        commentStore.createHubConnection('1');
        commentStore.clearComments();
        expect(commentStore.comments).toHaveLength(0);
        expect(commentStore.hubConnection.stop).toHaveBeenCalled();
    });

    it('should load comments', async () => {
        const comments = [{ id: 1, body: 'Test comment', createdAt: '2024-01-01T00:00:00' }];
        commentStore.createHubConnection('1');
        commentStore.hubConnection.on.mock.calls[0][1](comments);
        expect(commentStore.comments[0].createdAt).toEqual(new Date('2024-01-01T00:00:00Z'));
    });

    it('should receive comment', async () => {
        const comment = { id: 1, body: 'New comment', createdAt: '2024-01-01T00:00:00' };
        commentStore.createHubConnection('1');
        commentStore.hubConnection.on.mock.calls[1][1](comment);
        expect(commentStore.comments[0].createdAt).toEqual(new Date('2024-01-01T00:00:00'));
    });
```

Рисунок 5.1 - Автоматичний тест роботи коментарів на сторінці оголошення (виконаний самостійно)

Тестування зручності користування, в свою чергу, зосереджується на тому, наскільки інтерфейс є інтуїтивно зрозумілим та легким у використанні для кінцевих користувачів. Це включає перевірку логічності навігації, розміщення елементів, зручності заповнення форм та доступності основних функцій. Таке тестування зазвичай проводиться з участю реальних користувачів, які виконують типові задачі та надають

зворотний зв'язок щодо їхнього досвіду, або за допомогою спеціалізованих інструментів які аналізують поведінку користувачів і виявляють проблемні зони в інтерфейсі. В нашому випадку вкрай важливо впевнитися, щоб всі користувачі, незалежно від їхнього технічного рівня, могли легко і зручно використовувати систему.

Іншим важливим видом тестування є кросбраузерне та кросплатформне тестування, в ході якого перевіряється, чи правильно працює веб-сервіс на різних браузерах та платформах. Це включає перевірку сумісності з основними браузерами та різними операційними системами. Крім того, важливо перевіряти не тільки десктопні, але й мобільні версії сайту, оскільки користувачі можуть звертатися до системи з різних пристроїв.

Тестування продуктивності є критичним для забезпечення швидкого завантаження сторінок та реакції на дії користувачів, особливо під час пікових навантажень. Це включає аналіз часу завантаження ресурсів, роботи з великими обсягами даних, швидкості відповіді серверу та рендерингу сторінок. Тестування продуктивності допомагає виявити вузькі місця в системі та оптимізувати їх, забезпечуючи стабільну роботу навіть при значному збільшенні кількості користувачів. Наприклад, використання CDN та кешування може значно прискорити завантаження статичних ресурсів.

Тестування безпеки на фронтенді включає перевірку захисту від таких загроз, як XSS, CSRF та інших вразливостей. Це включає перевірку коректного впровадження механізмів захисту, таких як Content Security Policy, а також належної обробки користувацьких даних для запобігання ін'єкціям та іншим атакам.

Загалом, тестування фронтенду веб-сервісу для житлових комплексів було багато етапним процесом перевірки функціональності, що включав в себе модульні та інтеграційні тести, а також перевірку зручності використання, кроссбраузерной та кросплатформної сумісності, продуктивності та безпеки. Кожен з цих аспектів є критично важливим для забезпечення високої якості інтерфейсу користувача, його стабільності та відповідності вимогам, що в кінцевому підсумку забезпечує успішне функціонування системи в реальних умовах. Тестування повинно бути ретельним і регулярним, щоб вчасно виявляти і усувати проблеми, підтримуючи високу якість продукту на всіх етапах його життєвого циклу.

ВИСНОВКИ

Розробка веб-сервісу для житлових комплексів з точки зору FrontEnd дозволила досягти кількох важливих цілей, що сприяють ефективному управлінню житловими комплексами та забезпечують високий рівень зручності для користувачів. Для розробки була використана платформа Azure, що дозволило реалізувати всі необхідні функціональні можливості та забезпечити масштабованість і гнучкість системи.

Використання React забезпечило створення динамічного та інтерактивного інтерфейсу, який відповідає сучасним вимогам до веб-додатків. Компонентна архітектура дозволила розбити інтерфейс на незалежні частини, що спрощує розробку, тестування та підтримку коду. Реалізація компонентів для аутентифікації, модулів замовлення та відстеження послуг, внутрішніх чатів, оголошень та інших забезпечила повний набір функцій для користувачів різних ролей.

Впровадження адаптивного дизайну та використання CSS-фреймворків забезпечило коректне відображення додатку на різних пристроях, що підвищило зручність використання системи для користувачів.

Особлива увага була приділена безпеці додатку. Використання HTTPS для шифрування даних, переданих між клієнтом і сервером, забезпечило конфіденційність інформації.Таким чином, розробка веб-сервісу для житлових комплексів з точки зору FrontEnd виявилася успішною, забезпечивши високу продуктивність, надійність, безпеку та зручність використання системи. Використання сучасних технологій та інструментів дозволило створити масштабований та гнучкий додаток, що відповідає вимогам усім актуальним вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Front-End Development. URL: https://softwareconnect.com/property-management/condo-control-central/ (дата звернення: 09.05.2024)

2. Condo Control Central 2023 Review, Pros & Cons. URL: https://softwareconnect.com/property-management/condo-control-central/ (дата звернення: 09.05.2024)

3. BuildingLink Software review. URL: https://www.softwareadvice.com/property/buildinglink-profile/ (дата звернення: 09.05.2024)

4. Yardi Voyager Property Management. URL: https://softwareconnect.com/property-management/yardi-voyager/ (дата звернення: 09.05.2024)

5. Roth R. User Interface and User Experience (UI/UX) Design. *Geographic Information Science & Technology Body of Knowledge*. 2017. T. 2017, Q2. URL: https://doi.org/10.22224/gistbok/2017.2.5 (дата звернення: 09.05.2024)

6. Peguero K., Cheng X. CSRF protection in JavaScript frameworks and the security of JavaScript applications. *High-Confidence Computing*. 2021. T. 1, № 2. С. 100035. URL: https://doi.org/10.1016/j.hcc.2021.100035 (дата звернення: 09.05.2024)

7. Peguero K., Cheng X. CSRF protection in JavaScript frameworks and the security of JavaScript applications. *High-Confidence Computing*. 2021. T. 1, № 2. С. 100035. URL: https://doi.org/10.1016/j.hcc.2021.100035 (дата звернення: 09.05.2024)

8. About MobX. URL: https://mobx.js.org/README.html (дата звернення: 09.05.2024)

9. What is Bootstrap - Everything you need to know. URL: https://www.hostinger.com/tutorials/what-is-bootstrap/ (дата звернення: 09.05.2024)

10. What Are User Flows in UX Design?. URL: https://careerfoundry.com/en/blog/ux-design/what-are-user-flows (дата звернення: 09.05.2024)

11. Wireframe Basics. *Web Development with the Mac®*. Indianapolis, IN, USA, 2011. С. 241–257. URL: https://doi.org/10.1002/9781118255759.ch10 (дата звернення: 09.05.2024)

12. ASP.NET Core SignalR JavaScript Client. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/JavaScript-client?view=aspnetcore-8.0&tabs=visual-studio (дата звернення: 09.05.2024)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016301014

Дата перевірки:
30.05.2024 21:45:43 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
30.05.2024 21:48:05 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-1_Янов_Д_І

Кількість сторінок: 49 Кількість слів: 6714 Кількість символів: 55765 Розмір файлу: 6.01 MB ID файлу: 1016089044

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

12.2%
Схожість

Найбільша схожість: 8.5% з джерелом з Бібліотеки (ID файлу: 1016089043)

Пошук збігів з Інтернетом не проводився

12.2% Джерела з Бібліотеки 428 ..... Сторінка 51

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 9 сторінок

ДОДАТОК Б

Слайди презентації

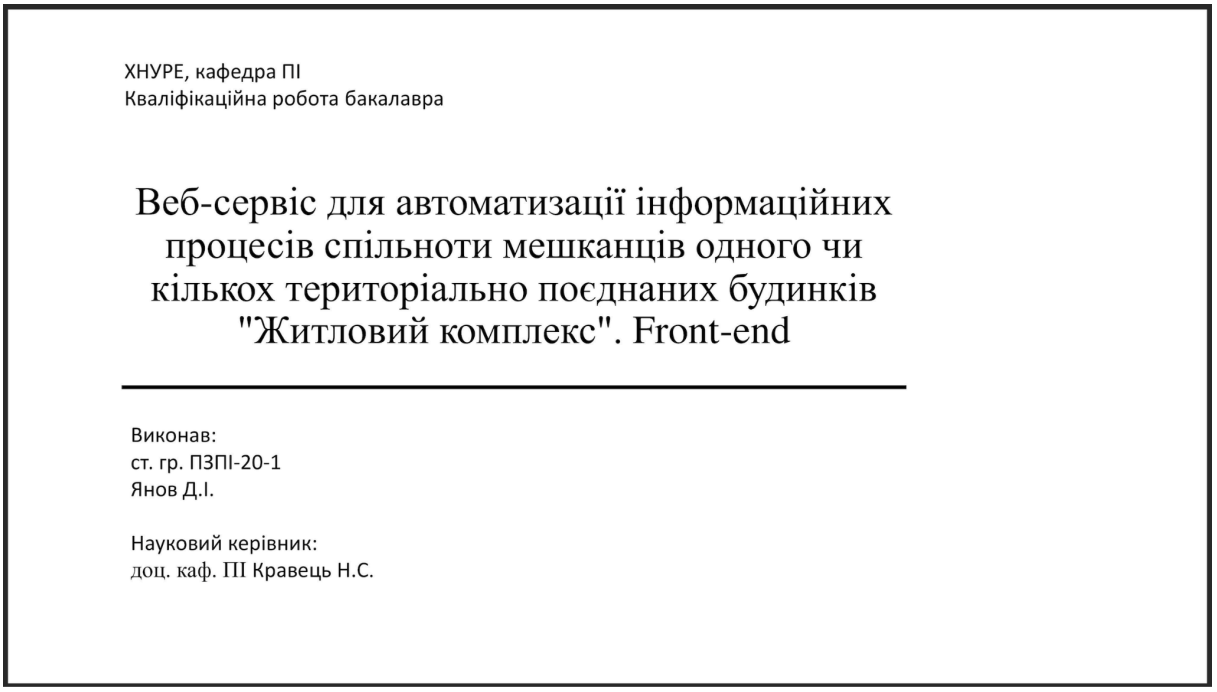


Рисунок Б.1 – Слайд 1 (рисунок виконаний самостійно)

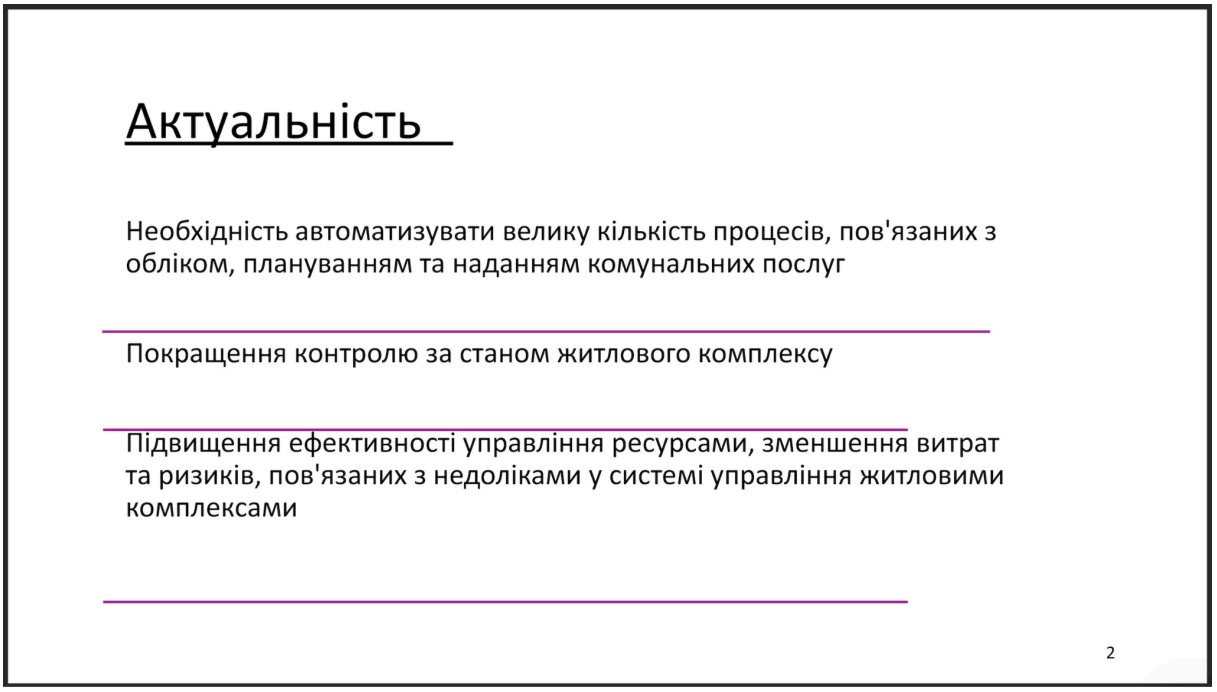


Рисунок Б.2 – Слайд 2 (рисунок виконаний самостійно)

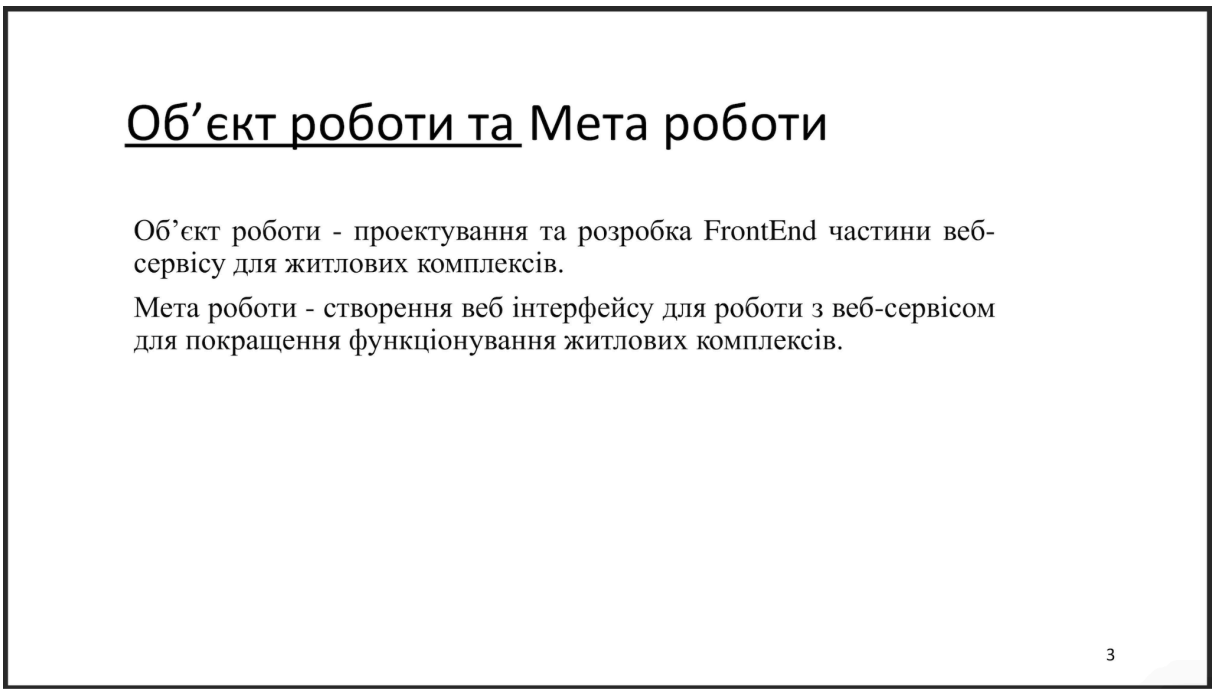


Рисунок Б.3 – Слайд 3 (рисунок виконаний самостійно)

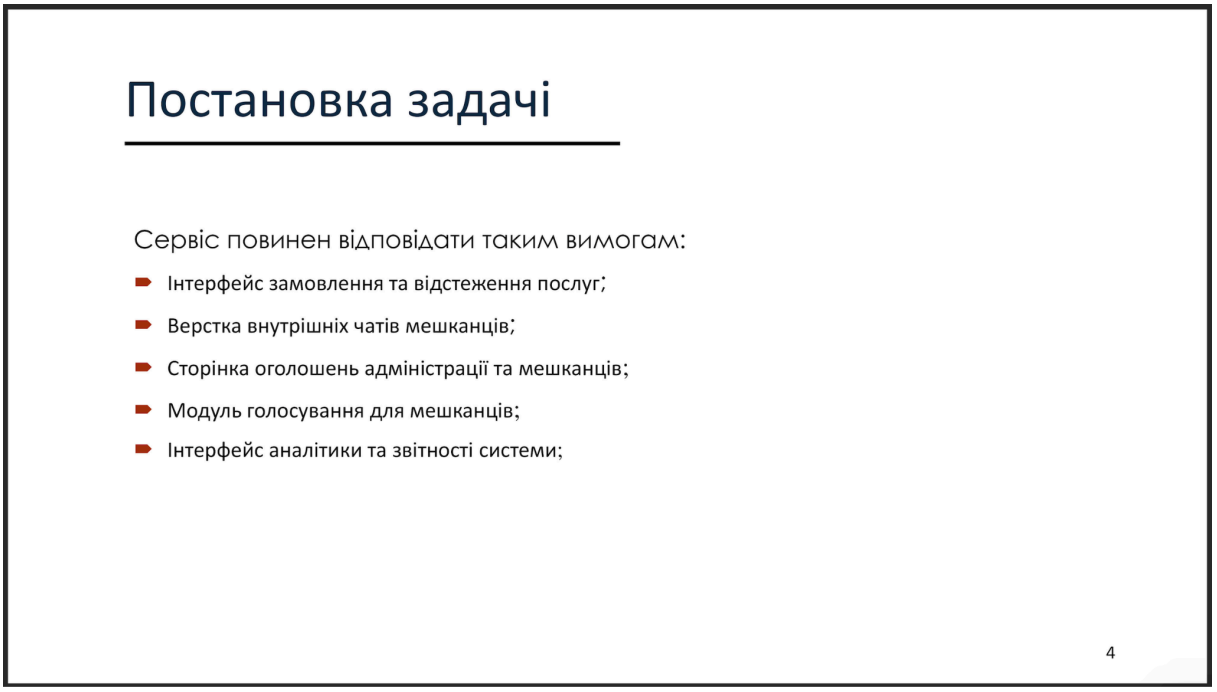
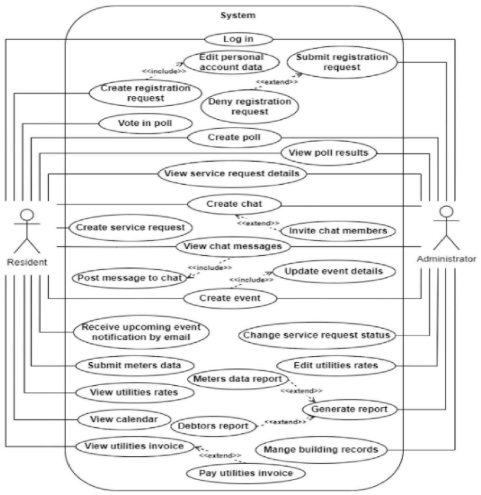


Рисунок Б.4 – Слайд 4 (рисунок виконаний самостійно)

Use Case Diagram



5

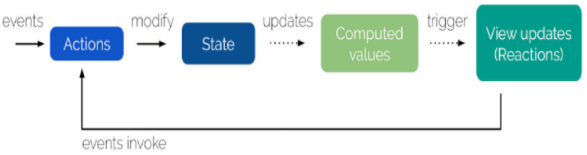
Рисунок Б.5 – Слайд 5 (рисунок виконаний самостійно)

Чому Mobx

Mobx - легковаго, крос-платформеного state management бібліотека для JavaScript додатків.

Переваги:

- Легкість використання;
- Легкість використання;
- Підтримка реактивного програмування;



6

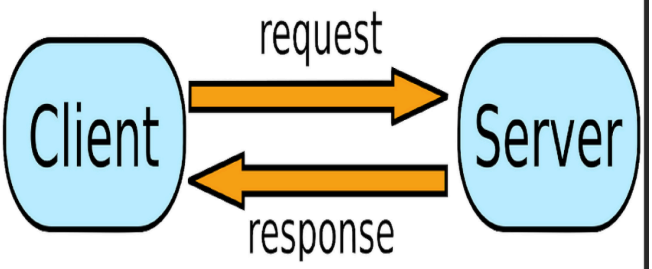
Рисунок Б.6 – Слайд 6 (рисунок виконаний самостійно)

Чому Axios

Axios - легковаго, крос-платформеного HTTP-клієнта для роботи з API в JavaScript додатках.

Переваги:

- Легкість використання;
- Підтримка різних форматів даних;
- Підтримка обробки асинхронних запитів;



7

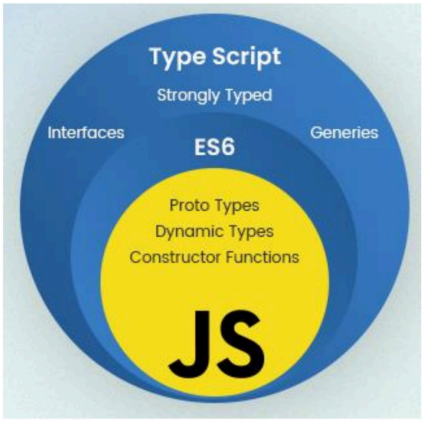
Рисунок Б.7 – Слайд 7 (рисунок виконаний самостійно)

Чому Typescript

Typescript - надмножина JavaScript для побудови масштабованих додатків який додає **статичну типізацію**

Переваги:

- Статична типізація;
- Покращена розробка з підтримкою IDE;
- Сумісність з JavaScript.

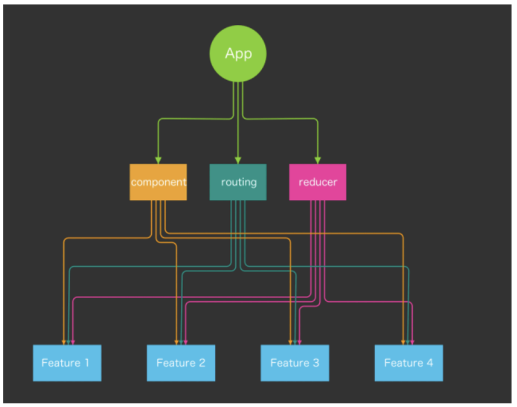


8

Рисунок Б.8 – Слайд 8 (рисунок виконаний самостійно)

Загальна архітектура проекту

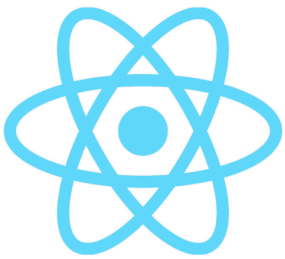
- **React**: Бібліотека для побудови користувацьких інтерфейсів. Використовується для створення компонентів, які відображають дані і реагують на дії користувача.
- **MobX**: Бібліотека для управління станом додатка. Використовується для зберігання та маніпуляції даними, які необхідні для відображення в компонентах React.
- **SignalR**: Бібліотека для реалізації реального часу. Використовується для підтримки WebSocket-з'єднань між клієнтом і сервером для обміну даними в реальному часі.



9

Рисунок Б.9 – Слайд 9 (рисунок виконаний самостійно)

Технології



TS

Mobx

SignalR

AXIOS

JWT

10

Рисунок Б.10 – Слайд 10 (рисунок виконаний самостійно)

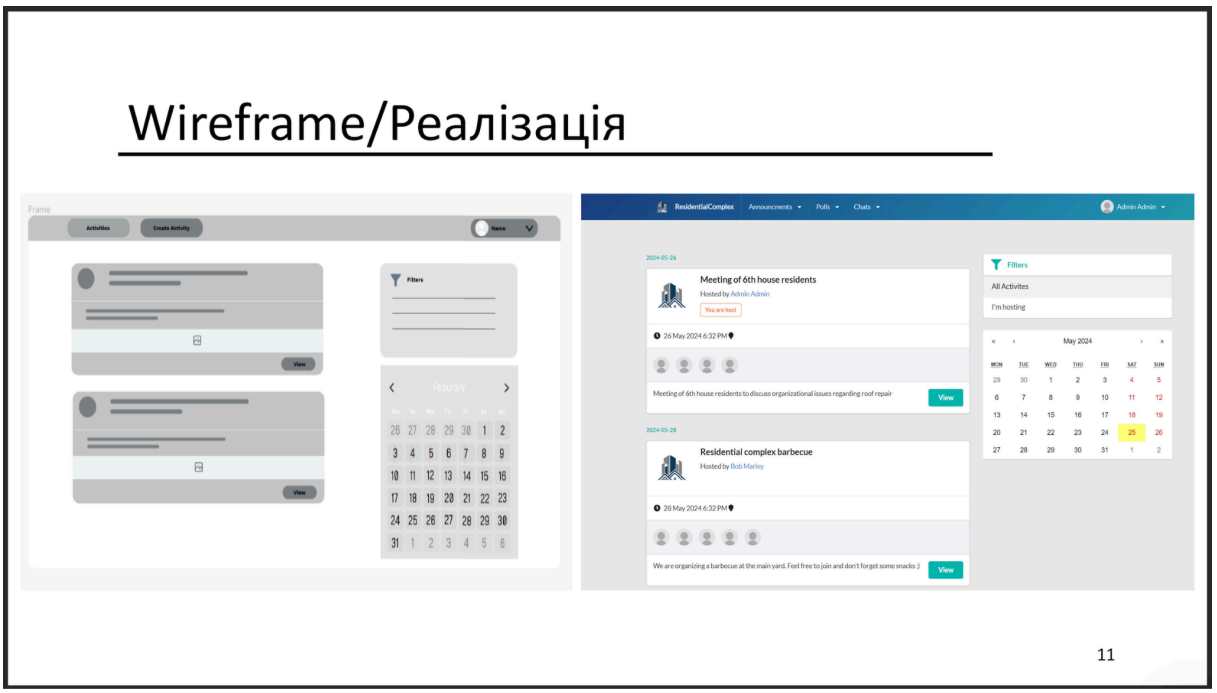


Рисунок Б.11 – Слайд 11 (рисунок виконаний самостійно)

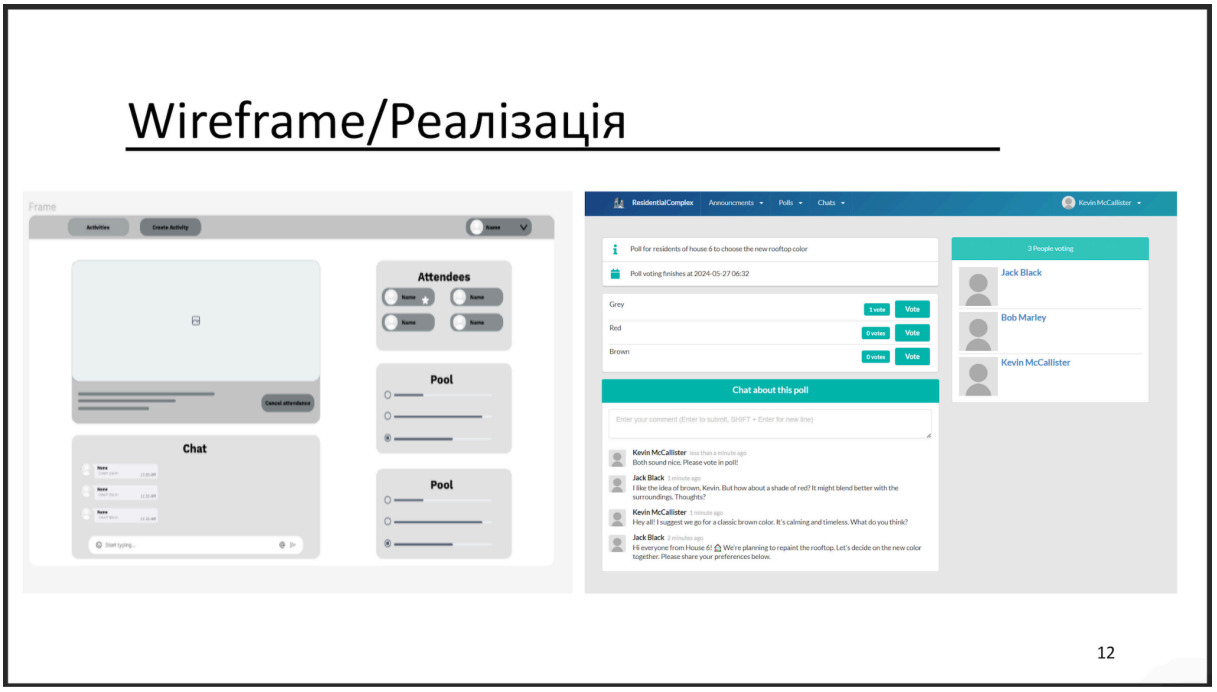


Рисунок Б.12 – Слайд 12 (рисунок виконаний самостійно)

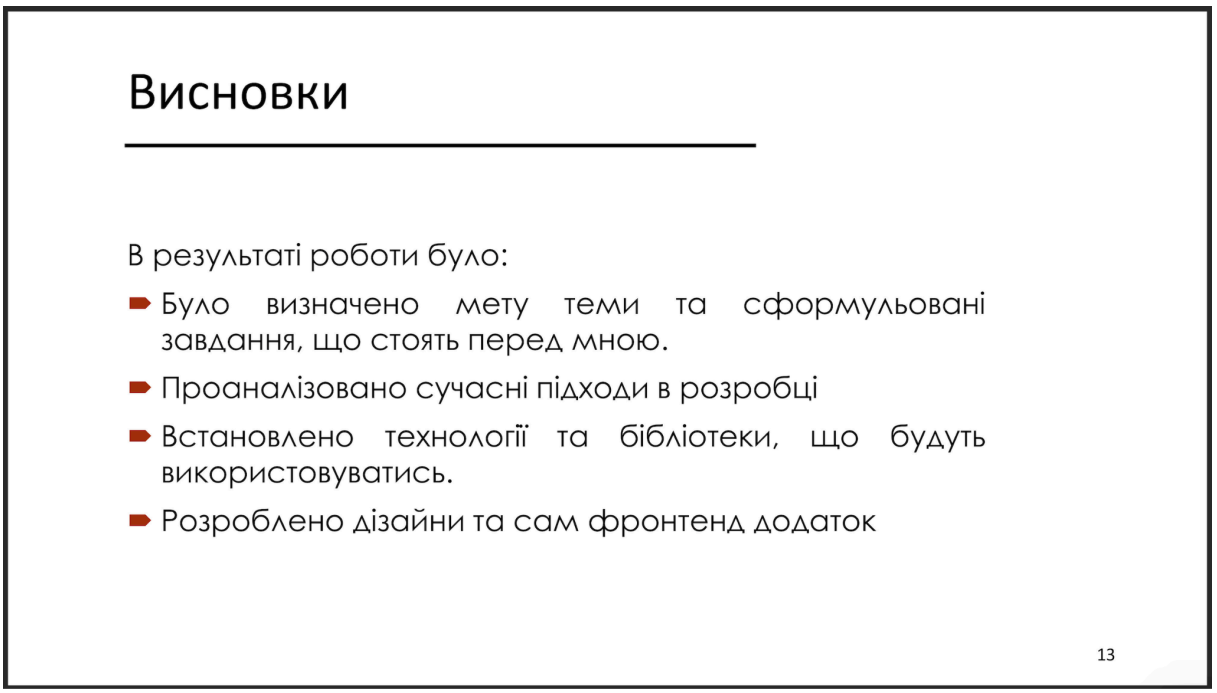


Рисунок Б.13 – Слайд 13 (рисунок виконаний самостійно)

ДОДАТОК В

Специфікація вимог до програмного продукту

Специфікація ПЗ

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

веб-сервіс для автоматизації інформаційних процесів спільноти мшканців одного чи кількох територіально поєднаних будинків "Житловий комплекс". FrontEnd

Студент гр. ПЗПІ-20-1 _____ Волосніков М.С.

Студент гр. ПЗПІ-20-1 _____ Жеребний В.В.

Студент гр. ПЗПІ-20-1 _____ Янов Д.І.

Харків

2024 р.

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

Програмний продукт є веб-сервісом для житлових комплексів, який забезпечує інтеграцію між мешканцями та адміністрацією житлових комплексів. Він надає можливості управління запитами на обслуговування, сповіщення мешканців про події та новини, а також здійснення онлайн-платежів. BackEnd частина реалізована на платформі .NET, FrontEnd – на React, база даних використовує Azure SQL, процеси CI/CD налаштовані за допомогою GitHub Actions, і все розгортається в Docker. Система створена з метою оптимізації взаємодії між мешканцями та адміністрацією, забезпечення прозорості та ефективності управління житловими комплексами, а також підвищення рівня задоволеності мешканців через покращену комунікацію та швидке вирішення проблем.

1.2 Мета

Метою розробки є створення надійного та безпечного веб-сервісу, який полегшить управління житловими комплексами, покращить комунікацію між мешканцями та адміністрацією, забезпечує швидке обслуговування запитів та підтримку різних видів онлайн-платежів. Програмний продукт повинен забезпечити ефективний процес управління запитами на обслуговування, знизити адміністративні витрати за рахунок автоматизації рутинних завдань, та забезпечити прозорість процесів для мешканців і адміністрації. Основна мета полягає у створенні інтуїтивно зрозумілого інтерфейсу для мешканців, що дозволить їм легко подавати запити, відстежувати їх статус та здійснювати платежі, а також надання адміністрації інструментів для ефективного управління житловим комплексом.

1.3 Межі

Продукт розрахований на використання в житлових комплексах і не призначений для управління комерційними або промисловими об'єктами. Він включає обмеження по кількості одночасних користувачів, розмірів бази даних і інтеграції з іншими системами. Продукт буде розроблений на основі веб-технологій і не включає розробку мобільних додатків. Межі також включають обмеження по функціоналу, зосереджуючись на ключових аспектах управління житловими комплексами, таких як управління запитами на обслуговування, сповіщення та онлайн-платежі. Не передбачається інтеграція з широким спектром зовнішніх систем, крім необхідних платіжних шлюзів та сервісів сповіщень.

1.4 Посилання

Цей веб-сервіс для житлових комплексів призначений для спрощення та автоматизації управління, забезпечуючи мешканцям зручний доступ до необхідної інформації та сервісів. Впровадження даного продукту дозволить знизити кількість ручної роботи, оптимізувати комунікацію між адміністраторами та мешканцями, а також покращити загальну ефективність управління житловим комплексом. Система аутентифікації та авторизації забезпечить безпечний доступ та контроль над даними, що є критично важливим для забезпечення конфіденційності інформації.

Цей веб-сервіс допоможе управляючим компаніям раціоналізувати процеси, що дозволяє більш ефективно використовувати ресурси та досягати поставлених цілей. Інтеграція з Azure SQL забезпечує високу надійність та продуктивність зберігання даних, а CI/CD процеси, налаштовані за допомогою GitHub Actions, дозволять швидко і безпечно впроваджувати нові функціональні можливості та оновлення.

Використання Docker дозволить легко масштабувати систему та забезпечити її стабільну роботу у різних середовищах.

Загалом, впровадження цього веб-сервісу з back-end на .NET, front-end на React, та базою даних Azure SQL дозволить житловим комплексам значно покращити управління, підвищити прозорість процесів та задоволеність мешканців. Веб-сервіс також забезпечує можливість для створення детальних звітів та аналізу даних, що допоможе приймати більш обґрунтовані управлінські рішення та розробляти ефективні стратегії для покращення обслуговування мешканців.

1.5 Означення та абревіатури

— CI/CD – безперервна інтеграція та доставка (Continuous Integration/Continuous Delivery);

— Azure SQL – керована реляційна база даних від Microsoft Azure;

— Docker – платформа для контейнеризації додатків;

— .NET – програмна платформа від Microsoft для розробки додатків;

— React – бібліотека JavaScript для створення користувацьких інтерфейсів;

— API – програмний інтерфейс додатків (Application Programming Interface);

— HTTPS – протокол захищеного перенесення гіпертексту (HyperText Transfer Protocol Secure);

— MFA – багатофакторна аутентифікація (Multi-Factor Authentication);

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Веб-сервіс для житлових комплексів має на меті стати невід’ємною частиною сучасного управління житловими приміщеннями, полегшуючи взаємодію між адміністрацією та мешканцями. Очікується, що він буде постійно оновлюватися та розширюватися, інтегруючись з новими технологіями та пристроями, такими як IoT (Internet of Things). Майбутні версії можуть включати підтримку автоматизації управління енергоспоживанням, інтеграцію з розумними пристроями для підвищення комфорту мешканців та розширені функції аналітики для більш ефективного управління ресурсами. Перспективи продукту також включають можливість створення мобільних додатків для зручнішого доступу мешканців до послуг, розширення функціоналу за рахунок інтеграції з іншими платформами управління житловими комплексами та покращення користувацького досвіду через персоналізовані сповіщення та рекомендації.

2.2 Функції продукту

Продукт включає наступні функції: управління запитами на обслуговування (створення, перегляд, оновлення та закриття запитів), сповіщення мешканців про події та новини (створення та розсилка сповіщень через різні канали, такі як email та push-повідомлення), підтримка онлайн-платежів (можливість здійснення платежів через інтегровані платіжні системи), аналітика та звітність (генерація звітів про діяльність та взаємодію мешканців з адміністрацією). Крім основних функцій, продукт включає модулі для управління профілями користувачів, налаштування прав доступу для адміністрації, ведення історії взаємодій та інтеграції з зовнішніми сервісами для сповіщень та платежів. Функціонал

також включає можливість адміністрування та управління різними житловими комплексами з єдиного інтерфейсу, забезпечуючи зручний та ефективний контроль за всіма об’єктами в одному місці.

2.3 Характеристики користувачів

Користувачі продукту включають адміністрацію житлових комплексів, мешканців та технічний персонал. Адміністрація матиме доступ до функцій управління та звітності, мешканці – до функцій створення запитів та здійснення платежів, технічний персонал – до функцій обслуговування запитів. Адміністрація використовуватиме систему для відстеження запитів мешканців, управління обслуговуючим персоналом та генерації звітів для аналізу ефективності роботи. Мешканці використовуватимуть веб-сервіс для подання запитів на обслуговування, відстеження статусу своїх запитів, отримання сповіщень про новини та події, а також здійснення онлайн-платежів. Технічний персонал буде використовувати систему для отримання інформації про запити, призначення завдань та відстеження виконання робіт.

2.4 Загальні обмеження

Обмеження включають максимальну кількість одночасних користувачів, обсяг даних, що зберігається в базі даних, обмеження по пропускній здатності мережі, і обмеження по інтеграції з зовнішніми системами. Продукт має обмеження по обсягу даних, що зберігаються в Azure SQL, враховуючи необхідність збереження історичних даних для аналітики та звітності. Обмеження також стосуються можливості одночасної роботи великої кількості користувачів, що може вимагати додаткової оптимізації і масштабування для забезпечення стабільної роботи під час пікових навантажень. Крім того, система обмежена у

можливостях інтеграції з деякими зовнішніми сервісами через технічні або безпекові причини.

2.5 Припущення й залежності

Припускається, що користувачі мають доступ до сучасних веб-браузерів і стабільного інтернет-з'єднання. Залежність від хмарної платформи Azure передбачає необхідність наявності відповідного підписки та підтримки з боку Azure для забезпечення безперебійної роботи бази даних. Припускається також, що користувачі мають базові навички роботи з комп'ютером та інтернетом, що дозволить їм без проблем користуватися веб-сервісом. Залежність від Docker і GitHub Actions вимагає, щоб інфраструктура підтримувала контейнери та безперервну інтеграцію і доставку, що забезпечує швидке та надійне розгортання оновлень та виправлень.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс користувача повинен бути розроблений на основі React, забезпечуючи зручний і інтуїтивно зрозумілий досвід користування. Користувацький інтерфейс повинен включати форми для створення запитів, інформаційні панелі для перегляду статусу запитів, функціонал для здійснення онлайн-платежів, та налаштування для керування профілем користувача. Інтерфейс має бути адаптивним, підтримувати різні розміри екранів та забезпечувати однаковий рівень зручності як на настільних комп’ютерах, так і на мобільних пристроях. Користувачам повинні бути доступні інтуїтивно зрозумілі меню та навігаційні елементи, що полегшують пошук потрібної інформації та виконання основних завдань. Інтерфейс також повинен включати функції доступності для користувачів з обмеженими можливостями, забезпечуючи доступність для широкого кола користувачів.

3.1.2 Апаратний інтерфейс

Веб-сервіс не вимагає спеціалізованого апаратного забезпечення і може працювати на будь-якому сучасному серверному обладнанні або віртуальній машині, що підтримує Docker. Система повинна бути оптимізована для роботи на хмарних платформах, таких як Azure, що забезпечить масштабованість та гнучкість у розгортанні. Необхідно забезпечити підтримку резервного копіювання та відновлення даних, що дозволяє зберігати цілісність даних у випадку апаратних збоїв або інших технічних проблем. Апаратний інтерфейс також має забезпечувати ефективну взаємодію з мережею для забезпечення швидкої передачі даних між сервером та користувачами.

3.1.3 Програмний інтерфейс

Програмний інтерфейс включає RESTful API для взаємодії між FrontEnd та BackEnd, а також інтеграційні точки для взаємодії з зовнішніми сервісами, такими як платіжні системи та сервіси сповіщень. API повинен підтримувати основні операції CRUD (створення, читання, оновлення, видалення) для управління даними, а також забезпечувати безпечну аутентифікацію та авторизацію користувачів. Програмний інтерфейс має бути добре задокументований, що дозволить розробникам легко інтегруватися з системою та розширювати її функціонал. Інтерфейси мають бути розроблені з урахуванням принципів безпеки, забезпечуючи захист даних від несанкціонованого доступу та забезпечуючи шифрування конфіденційної інформації.

3.1.4 Комунікаційний протокол

Комунікація між компонентами системи повинна здійснюватися через протокол HTTPS для забезпечення безпеки переданих даних. Взаємодія між FrontEnd та BackEnd відбувається за допомогою REST API, що забезпечує стандартизовану та ефективну передачу даних. Використання HTTPS гарантує, що всі дані, передані між клієнтом і сервером, захищені від перехоплення та підміни. Комунікаційний протокол має бути налаштований для забезпечення високої швидкості передачі даних, мінімізуючи затримки та забезпечуючи швидкий відгук системи на запити користувачів. Додатково, протокол повинен підтримувати механізми аутентифікації та авторизації для контролю доступу до ресурсів системи.

3.1.5 Обмеження пам’яті

Продукт має бути оптимізованим для роботи в середовищі з обмеженими ресурсами, включаючи обмеження по оперативній пам'яті і

дисковому просторі для контейнерів Docker. Система повинна ефективно використовувати доступні ресурси, забезпечуючи високу продуктивність навіть при обмеженій кількості оперативної пам’яті та дискового простору. Необхідно забезпечити механізми для моніторингу та управління ресурсами, що дозволить вчасно виявляти та усувати проблеми, пов'язані з недостатністю ресурсів. Оптимізація коду та архітектури системи дозволить зменшити навантаження на пам’ять та забезпечити стабільну роботу навіть при значних навантаженнях.

3.1.6 Операції

Основні операції включають обробку запитів на обслуговування, генерацію сповіщень, обробку платежів, а також резервне копіювання та відновлення даних. Всі операції мають бути автоматизовані і здійснюватися через CI/CD пайплайн за допомогою GitHub Actions. Операції повинні виконуватися швидко та ефективно, забезпечуючи високу продуктивність системи та задоволення користувачів. Система має забезпечувати можливість масштабування операцій для обробки збільшеного обсягу запитів та даних без втрати продуктивності. Крім того, операції повинні бути задокументовані та забезпечувати можливість відстеження та моніторингу для виявлення та усунення проблем.

3.2 Атрибути програмного продукту

3.2.1 Надійність

Програмний продукт повинен забезпечувати високу надійність через впровадження механізмів автоматичного відновлення і дублювання даних. Azure SQL забезпечує автоматичне резервне копіювання та відновлення, що зменшує ризик втрати даних. Надійність системи досягається за рахунок використання стійкої архітектури, яка включає резервні копії даних, механізми відновлення після збоїв та автоматичне моніторинг стану

системи. Додатково, система повинна підтримувати тестування на стресові та навантажувальні умови для виявлення потенційних слабких місць та забезпечення їх усунення до розгортання в реальне середовище.

3.2.2 Доступність

Продукт повинен бути доступний 24/7 з мінімальним часом простою. Використання Azure SQL і Docker забезпечує високу доступність та швидке відновлення сервісу у випадку збоїв. Доступність досягається за рахунок використання хмарних технологій, що забезпечують автоматичне масштабування та розподіл навантаження між серверами. Система повинна включати механізми резервного копіювання та відновлення, що забезпечує можливість швидкого відновлення роботи після збоїв або інших непередбачених ситуацій. Крім того, необхідно забезпечити надійність мережевих з'єднань та резервування ключових компонентів системи для забезпечення безперервної роботи.

3.2.3 Безпека

Безпека є критичним аспектом, включаючи захист даних, що зберігаються та передаються. Azure SQL забезпечує шифрування даних, а використання HTTPS гарантує безпечну передачу даних між компонентами системи. Крім того, аутентифікація та авторизація користувачів повинні бути налаштовані з використанням сучасних методів захисту, таких як OAuth. Безпека системи також включає захист від атак типу SQL-ін'єкцій, XSS та CSRF, а також регулярне проведення аудитів безпеки та тестування на проникнення. Всі дані користувачів повинні Бути захищені відповідно до вимог GDPR або інших відповідних нормативних актів.

3.2.4 Супроводжуваність

Система повинна бути легко супроводжуваною завдяки добре структурованому коду та детальній документації. Використання .NET та React спрощує процес супроводу та оновлення компонентів. Супроводжуваність забезпечується за рахунок модульної архітектури, що дозволяє легко вносити зміни та додавати новий функціонал без впливу на інші частини системи. Документація повинна включати опис всіх компонентів, API та інструкції з розгортання та супроводу системи. Регулярні оновлення та виправлення помилок повинні здійснюватися через налаштований CI/CD пайплайн, що забезпечує швидке та безпечне впровадження змін.

3.2.5 Переносимість

Продукт повинен бути переносимим між різними середовищами завдяки використанню Docker. Контейнери забезпечують незалежність від апаратного забезпечення та операційної системи, що дозволяє легко переміщувати систему між локальними та хмарними середовищами. Переносимість досягається за рахунок стандартизованих контейнерів, що включають всі необхідні залежності та налаштування для запуску системи. Використання Docker Compose спрощує процес розгортання та конфігурації системи в різних середовищах, забезпечуючи можливість швидкого та безпроблемного переміщення між середовищами розробки, тестування та продуктивного середовища.

3.2.6 Продуктивність

Система повинна забезпечувати високу продуктивність та швидкий відгук на запити користувачів. Оптимізація коду та використання ефективних алгоритмів дозволять зменшити затримки та забезпечити

швидке виконання операцій. Azure SQL забезпечує високу продуктивність бази даних завдяки оптимізації запитів та ефективному управлінню ресурсами. Крім того, необхідно забезпечити моніторинг продуктивності та регулярне тестування системи на навантаження для виявлення та усунення потенційних вузьких місць. Використання масштабованих архітектурних рішень дозволить забезпечити високу продуктивність навіть при значному збільшенні навантаження на систему.

3.3 Вимоги бази даних

База даних повинна підтримувати зберігання великої кількості даних та швидкий доступ до них. Azure SQL забезпечує високу продуктивність та надійність завдяки вбудованим механізмом оптимізації запитів, автоматичному резервному копіюванню та відновленню. База даних повинна підтримувати складні запити та транзакції, забезпечуючи швидке виконання операцій та збереження цілісності даних. Необхідно забезпечити надійний захист даних, включаючи шифрування та контроль доступу на основі ролей (RBAC). Крім того, база даних повинна підтримувати механізми реплікації для забезпечення високої доступності та можливості відновлення даних у випадку збоїв.