Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр післядипломної освіти
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмне забезпечення підтримки діяльності ОСББ.
Веб застосунок
(тема)

Виконав:
студент 4 курсу, групи ПЗПІп-22-2

Грабар О.М.
(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. Кириченко І.В.
(посада, прізвище, ініціали)

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

2024 р.

Харківський національний університет радіоелектроніки

Центр післядипломної освіти
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 Грабар Олександру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмне забезпечення підтримки діяльності ОСББ. Веб застосунок.

Затверджена наказом по університету від 17.06. 2024р. № 588 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.07.2024

3. Вихідні дані до роботи *Передбачити в кваліфікаційній роботі: проектування та розробку веб-застосунку системи для підтримки взаємодії між співвласниками багатоквартирного будинку та керівництвом ОСББ, створення інтерфейсу системи, забезпечення взаємодії з серверною частиною та тестування. Використовувати мову програмування JavaScript та бібліотеку React.*

4. Перелік питань, що потрібно опрацювати у роботі *Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування 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	20.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	28.06.2024	*виконано*
5	Тестування ПЗ	30.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.07.2024	*виконано*
7	Підготовка презентації та доповіді	10.07.2024	*виконано*
8	Попередній захист	12.07.2024	*виконано*
9	Нормоконтроль, рецензування	12.07.2024	*виконано*
10	Здача роботи у електронний архів	12.07.2024	*виконано*
11	Допуск до захисту у зав. кафедри	18.07.2024	*виконано*

Дата видачі завдання 06 травня 2024р.

Студент (ка)_____ Грабар О.М.
(підпис)

Керівник роботи _____ доц. Кириченко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 78 стор., 49 рис., 1 табл., 11 джерел.

ВЕБ-ЗАСТОСУНОК, КЕРУВАННЯ БУДИНКОМ, ЛОКАЛІЗАЦІЯ, НАДАННЯ ПОСЛУГ, I18NEXT, JAVASCRIPT, LEAFLET, REACT.JS.

Об’єкт розробки – клієнтська частина програмної системи для підтримки взаємодії між мешканцями будинку та керівництвом ОСББ.

Мета розробки – створити зручну та ефективну систему, яка зможе полегшити життя мешканцям ОСББ, керівництву ОСББ та допоможе акумулювати додаткові кошти на обслуговування об’єкту і тим самим зменшити квартплату для кожного мешканця.

Метод рішення – середовище розробки Microsoft Visual Studio Code, мова програмування JavaScript з розширенням JSX, бібліотека React.js, платформа Node.js, бібліотека i18next, бібліотека React Leaflet.

У результаті розробки створено веб-систему з використання сучасних технологічних рішень, що вирішує актуальні проблеми взаємодії у свері житлово комунальних господарств між мешканцями та керівництвом неприбуткової організації ОСББ.

WEB APPLICATION, BUILDING MANAGEMENT, LOCATION, PROVISION OF SERVICES, I18NEXT, JAVASCRIPT, LOCALIZATION, REACT.JS.

The object of development is the client part of the software system to support interaction between the residents of the building and the management of the condominium.

The goal of the development is to create a convenient and effective system that can make life easier for the residents of the condominium, management of the condominium

and help to accumulate additional funds for the maintenance of the facility and thereby reduce the rent for each resident.

Solution method – Microsoft Visual Studio Code development environment, JavaScript programming language with JSX extension, React.js library, Node.js platform, i18next library, React Leaflet library.

As a result of the development, a web system using modern technological solutions was created, which solves the actual problems of interaction in the field of housing and communal services between residents and the management of the non-profit organization of condominiums.

Я, Грабар Олександр Миколайович, студент гр. ПЗПІп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмне забезпечення підтримки діяльності ОСББ. Веб застосунок», що буде представлена в екзаменаційну комісію 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

Зараз в Україні все більше і більше людей у багатоквартирних будинках починають створювати власні об’єднання співвласників квартир багатоквартирних будинків (ОСББ) і, тим самим, зголошуються керувати самостійно (установчі збори та шляхом обрання правління ОСББ з мешканців будинку) своєю та загальною власністю. Після створення неприбуткової юридичної особи ОСББ стикається з проблемою бухгалтерського обліку та комунікації між власниками квартир та керівництвом ОСББ, яке вони самі обирають. З кожним роком росте попит у платформі яка б могла задовольнити всі потреби у житті ОСББ. Пропонується ідея платформи яка буде являти собою портал взаємодії керівництва ОСББ з мешканцями будинку. Цей портал надаватиме бухгалтерські послуги, статистику оплати, час спілкування між керівництвом ОСББ за сусідам між собою. Також платформа буде давати можливість мешканцям одного поверху/під’їзду можливість спілкування між собою для вирішення нагальних проблем та шаринг контактів між мешканцями поверху та поверхом нижче чи поверхом вище.

Під час практичного навчання проведено ER-моделювання, концептуальне та логічне моделювання реляційної БД та програми, проведена її фізичне реалізація.

В результаті розроблено додаток, який буде надавати можливість спілкування керівництву ОСББ з мешканцями будинку, а також самим мешканцям спілкуватися між собою. Надаватиме портал для сплати комунальних послуг. Буде можливість створювати заявки на виконання сервісних чи ремонтних робіт для мешканців, наприклад: виклик сантехніка, виклик електрика, виклик майстра по вікнах та дверях, заявка інтернет провайдеру, виклик завгоспа, який зможе щось полагодити у будинку чи ліквідувати аварію. Застосунок є легким для розуміння та швидкого пристосування, отже людина з будь-яким досвідом зможе з легкістю його використовувати.

В ході створення додатку були використані середа розробки PyCharm [8], мова програмування Python, та система управління базами даних Oracle MySQL [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Аналіз предметної області інформаційної платформи для ОСББ дозволяє зрозуміти, що ця область є дуже важливою для ефективного управління будинком.

Ця система надаватиме бухгалтерські послуги, статистику оплати, можливість спілкування між керівництвом ОСББ за сусідам між собою. Також платформа буде давати можливість мешканцям одного поверху/під'їзду можливість спілкування між собою для вирішення нагальних проблем та шеринг контактів між мешканцями поверху та поверхом нижче чи поверхом вище.

Зараз в Україні все більше і більше людей у багатоквартирних будинках починають створювати власні ОСББ і, тим самим, зголошуються керувати самостійно (установчі збори та шляхом обрання правління ОСББ з мешканців будинку) своєю та загальною власністю. Після створення неприбуткової юридичної організації ОСББ стикається з проблемою бухгалтерського обліку та комунікації між власниками квартир та керівництвом ОСББ, яке вони самі обирають. З кожним роком росте попит у платформі яка б могла задовольнити всі потреби у житті ОСББ.

Система дозолятиме забезпечувати життєдіяльність ОСББ. Вона буде містити всю інформацію про ОСББ, його керівництво, всіх проживаючих у будинку в цілому, а також у квартирі, інформацію про прописаних мешканців квартири (це дозволить робити документи про склад родини наприклад), наявність автівки, марку, номер квартири і ім'я її власника. Надаватиме бухгалтерські послуги, розсилатиме нагадування про сплату квартплати, про заборгованість на початок місяця, про послуги якими користується та чи інша квартира, інформування про події у житті будинку, а також прозорий звіт по витратах ОСББ.

Створюватиме додаткові послуги з яких зможе заробляти ОСББ, наприклад: прибирання на поверсі, прибирання у квартирі, замовлення їжі, доставка води, послуга домофону, вивіз сміття і таке інше.

Система буде надавати можливість спілкування керівництву ОСББ з мешканцями будинку, а також самим мешканцям спілкуватися між собою. Надаватиме портал для сплати комунальних послуг, можливість створювати заявки на виконання сервісних чи ремонтних робіт для мешканців, наприклад: виклик сантехніка, виклик електрика, виклик майстра по вікнах та дверях, заявка інтернет провайдеру, виклик завгоспа, який зможе щось полагодити у будинку чи ліквідувати аварію.

У майбутньому планується розширення функціоналу шляхом додавання за додаткову плату користувачам додаткові послуги, наприклад: послуги відео нагляду, чи послуги служби охорони.

1.2Аналіз існуючих аналогів

Онлайн сервіс для ОСББ та управляючих компаній. Тут є все необхідне від бухгалтерії до роботи з мешканцями. Програма «Мій дім» [4] економить ваш час та зменшує кількість рутинних завдань. Персональні кабінети мешканців збільшують прозорість роботи та довіру мешканців. Працює на комп'ютері, телефоні, планшеті, не потребує встановлення (див. рис.2.1).

В системі «Мій дім» [4] можна зробити наступне:

- користуватись всіма функціями системи протягом 7 днів: вносити показники, робити розрахунки, формувати квитанції, налаштовувати звітність;
- створити сайт організації;
- зберегти всі налаштування.

«ДАХ» (див. рис.2.2). - веб-кабінет та мобільний застосунок для всіх учасників ОСББ [5]:

- голови та членів правління,
- бухгалтера,
- ревізійної комісії,
- співвласників будинку.

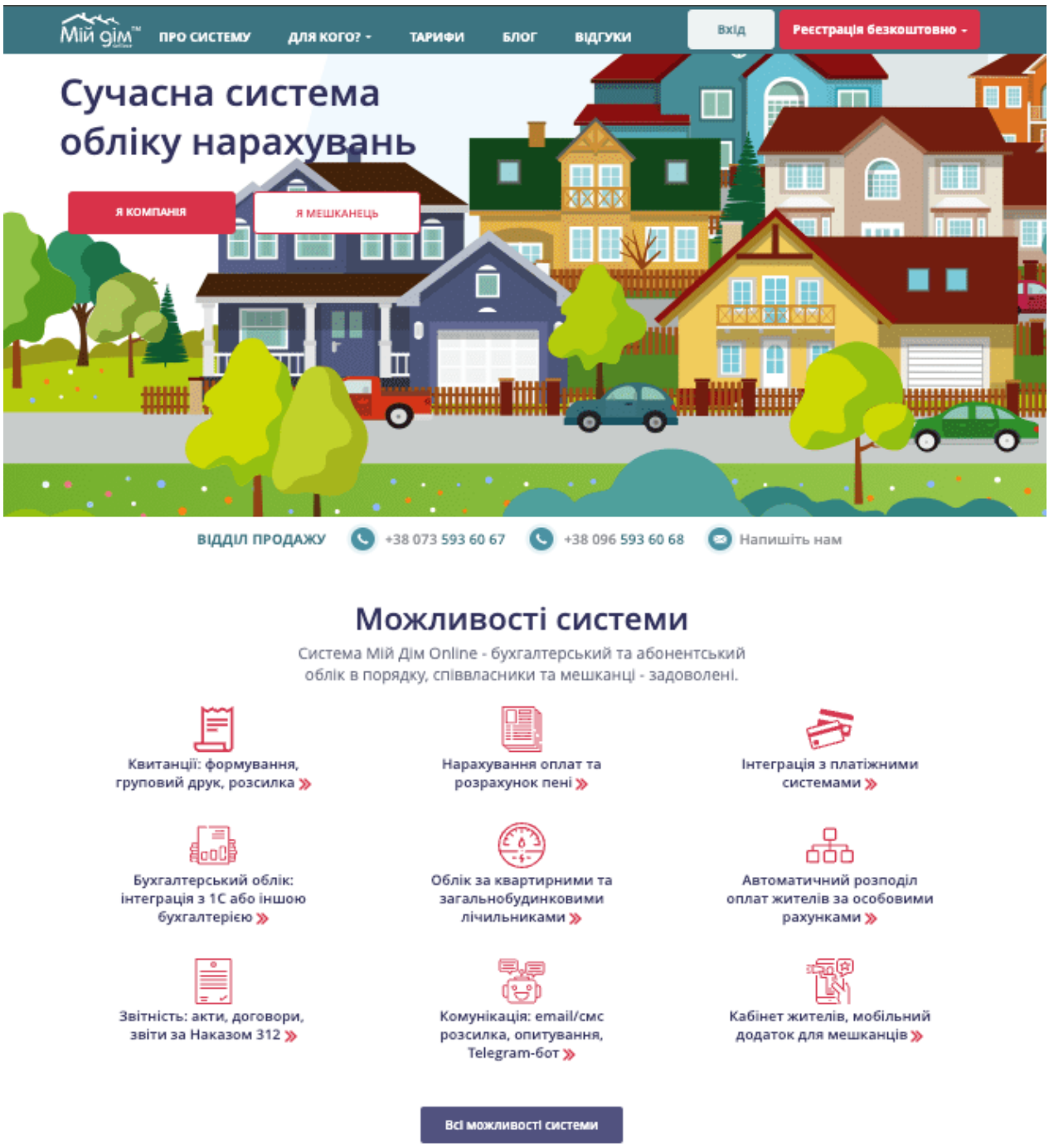


Рисунок 1.1 – Система «Мій дім» [4]

«ОМО» (див. рис.1.3) - дозволяє інтегрувати будинок та його мешканців до загальної системи розумного будинку, мати можливість керувати віддалено пристроями які знаходяться у квартирі або під'їзду, дозволяють надавати доступ до камери спостереження або домофону [6].

Найбільший мінус цього провайдера – це те що у нього немає білінгової системи яка може вести бух облік.

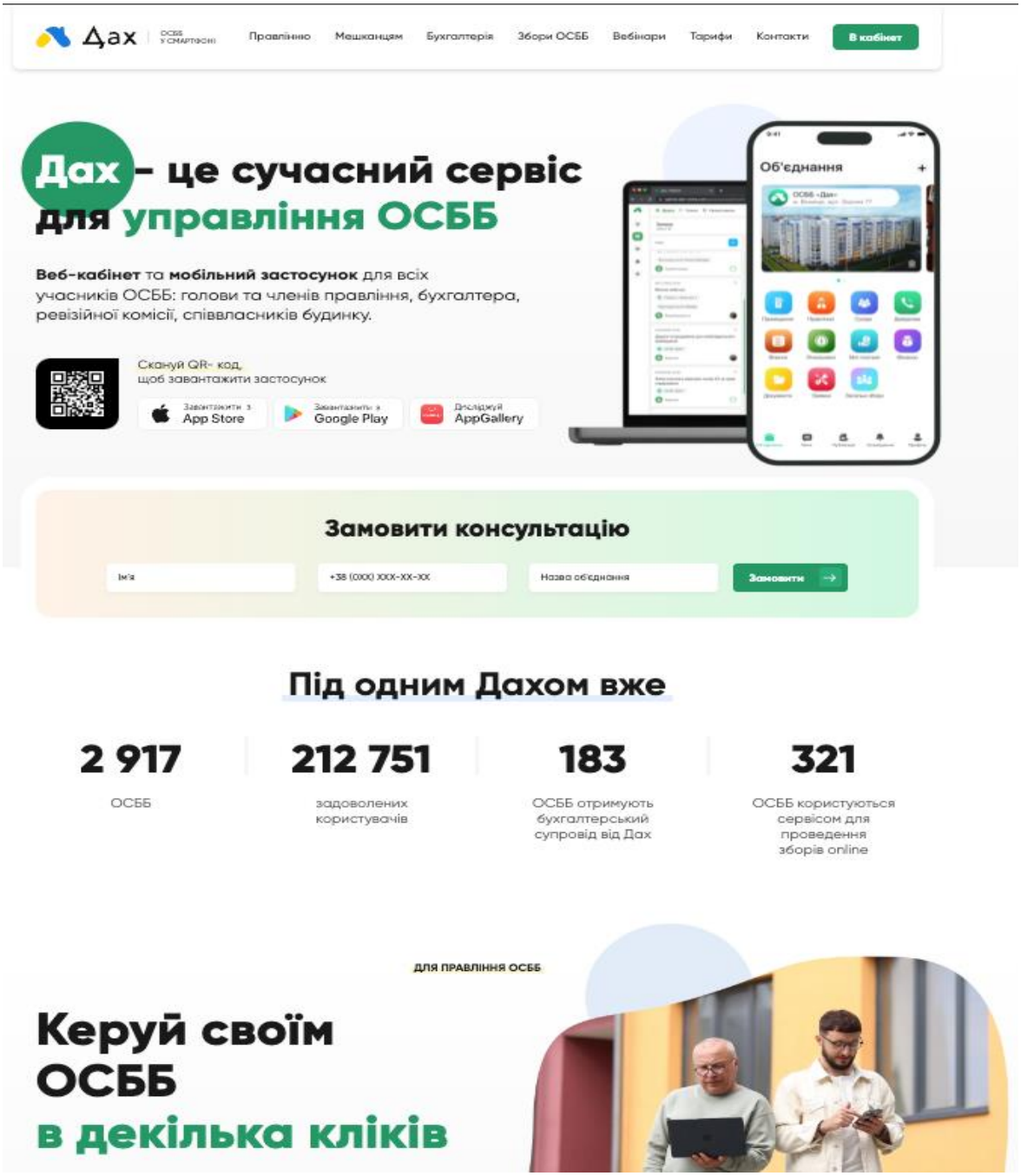


Рисунок 1.2 – Система «ДАХ» [5]

«Моє ОСББ» (див. рис.1.4) - сервіс створений для проведення загальних зборів ОСББ з використанням кваліфікованого електронного підпису (КЕП). Для всіх ОСББ України [7].

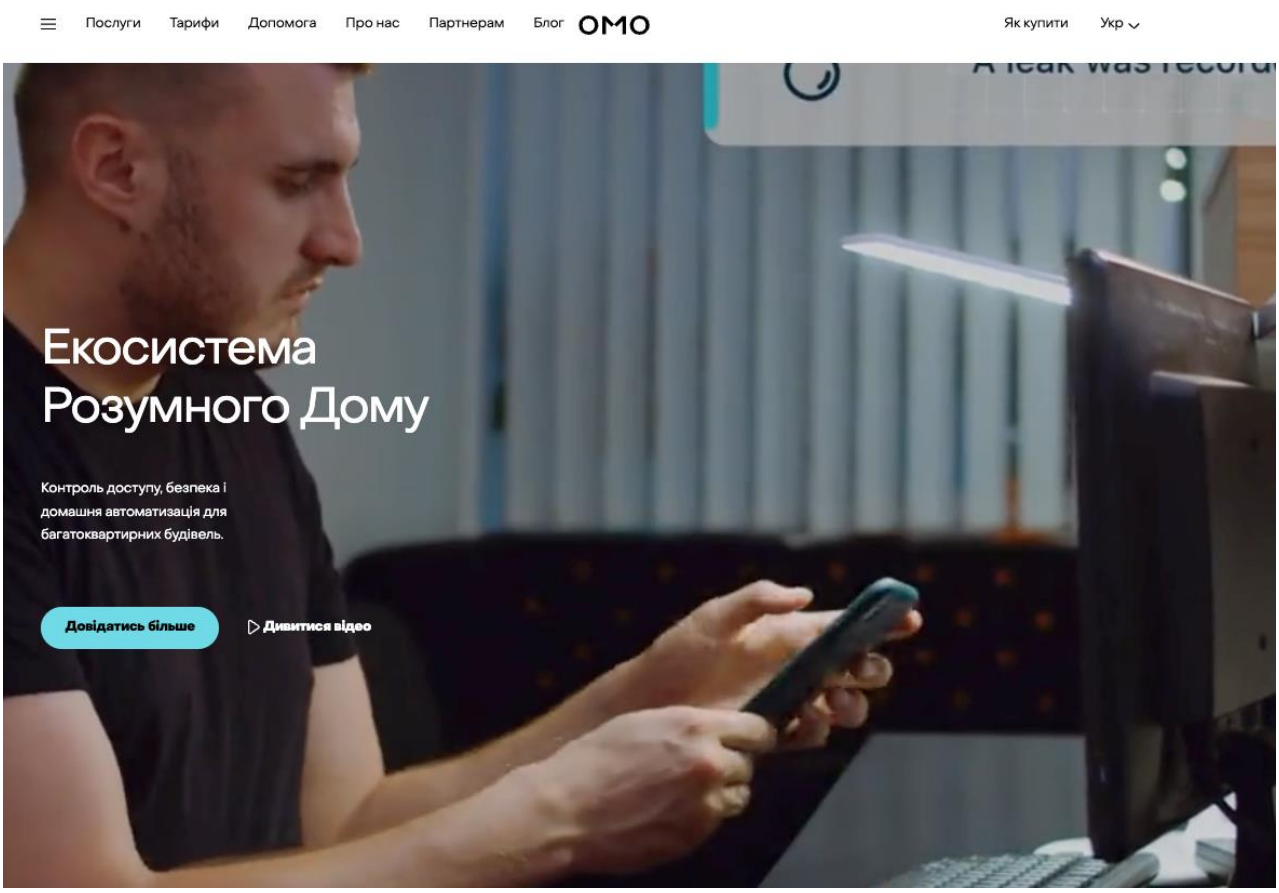


Рисунок 1.3 – Система «ОМО» [6]

Перевага це доступність створення голосування для мешканців ОСББ, яке зазвичай займає багато часу для організації їх а також може бути дуже дорогим бо треба розсилати замовленого листа тим мешканцям які були відсутні на зборах. Мінуси – це те що ця платформа не має додаткової функціональності.

В конкурентів спостерігається найчастіше такий функціонал як:

- можливість отримання довідок;
- інформування про оплату квартири та борг;
- кабінети бухгалтера, мешканця та управління ОСББ;
- опитування, голосування;
- розсилка квитанцій та повідомлень;
- можливість оплати комунальних платежів;
- створення заявки та поточні роботи майстрів.

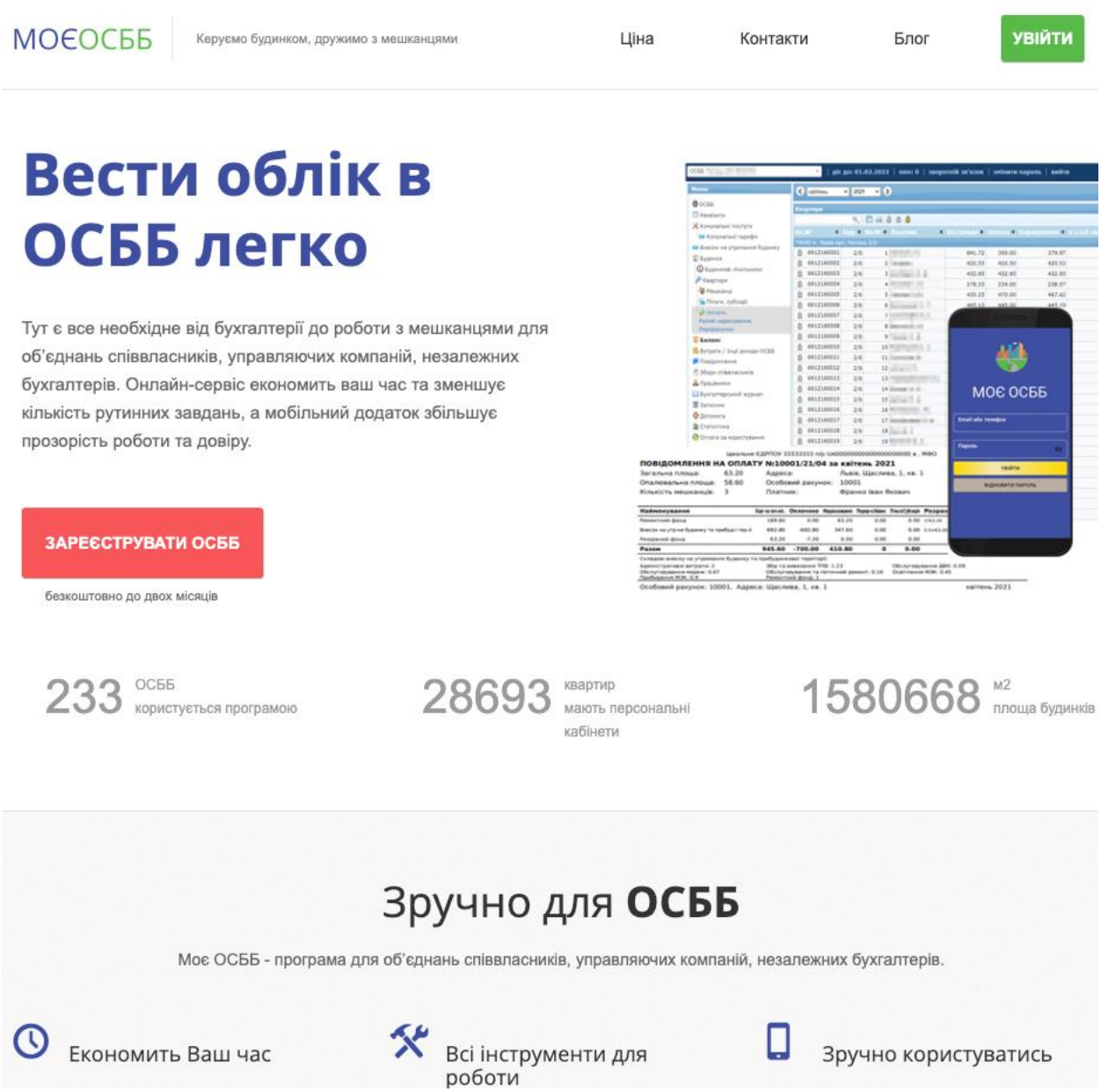


Рисунок 1.4 – Система «Моє ОСББ» [7]

З функцій бухгалтера стандартними є:

- ведення особових рахунків;
- фінансова та податкова звітність;
- ведення первинної документації ОСББ;
- пільги та субсидії;
- ведення кадрового обліку та зарплат.

В таблиці 1 наведено порівняння аналогів програмного забезпечення за виділеними критеріями.

Таблиця 1 – Порівняльна таблиця (таблиця виконана самостійно)

Аналог	Бухгалтерський облік	Звітність	Облік за лічильниками	Інтеграція з платіжними системами	Комунікація всередині платформи	Інтеграція з розумним будинком	Мобільний додаток
ОМО	-	+	+	+	-	+	+
ДАХ	+	+	-	+	+	-	+
Моє ОСББ	+	+	+	+	+	-	-
Мій дім	+	+	+	+	+	-	+
Моя оселя	-	+	-	-	+	-	-

У всіх сервісах присутній мобільний застосунок.

З нечастих функцій можна виокремити:

- чати для спілкування мешканців (для поверху і під'їзду)
- можливість швидкого інформування мешканців;
- відео нагляд;
- служба охорони;
- спрощений доступ до будинку;
- розділ з оголошеннями;
- роль гостя в особистому кабінеті орендаря;
- збори онлайн;
- телеграм бот;
- пошук сусіда за номером квартири, машини, поверху;
- оцінка послуг майстрів.

В жодного конкурента не виявлено послуг доставки води та можливості створення нових послуг.

Також в ході дослідження вдалося з’ясувати деякі цікаві моменти:

- 88% молодих пенсіонерів (віком до 67 років) самостійно здійснюють онлайн покупки за допомогою банківської картки.
- літнім людям зазвичай допомагають з онлайн оплатами молодші члени сім’ї.
- на західному ринку також присутні схожі рішення такого типу, ОСББ там називаються HOA, рішення які там використовуються є багатофункціональними CRM системами.

Онлайн опитування проведене нами представлене на рис.1.5.

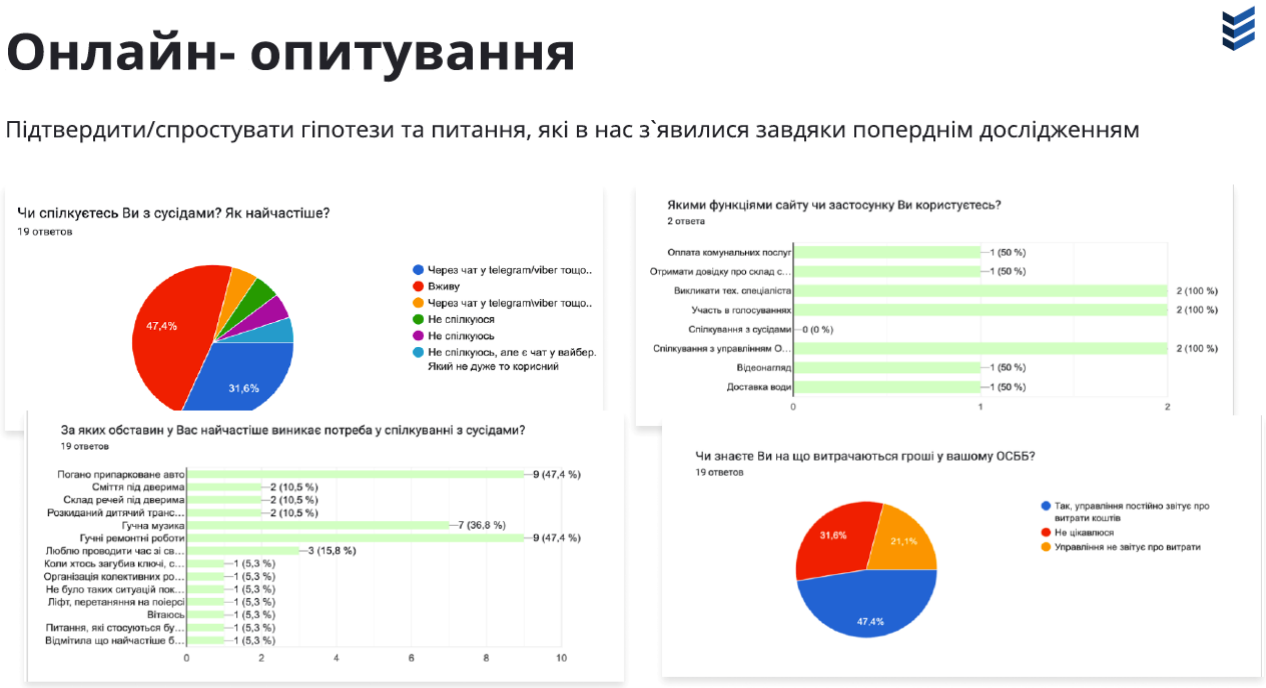


Рисунок 1.5 – Онлайн опитування (рисунок виконаний самостійно)

Як висновок, можна стверджувати, що кожен з аналогів має реалізацію лише частини функціоналу, що найчастіше використовуються користувачем. Серед функціоналу, якого не вистачає більшості даних продуктів слід відмітити можливість інтеграції сторонніх додатків та розширення функціоналу, інтеграції декількох функціональних особливостей.

1.3 Постановка задачі

Необхідно спроектувати систему для збереження інформації стосовно основної роботи житлового кооперативу, інформації про мешканців, засобів обліку та автоматизації. Створити інформаційну систему, що буде забезпечувати функції які наведені нижче.

Система дозолятиме забезпечувати життєдіяльність ОСББ. Вона буде містити всю інформацію про ОСББ, його керівництво, всіх проживаючих у будинку в цілому, а також у квартирі, інформацію про прописаних мешканців квартири (це дозволить робити документи про склад родини наприклад), наявність автівки, марку, номер квартири і ім’я її власника. Надаватиме бухгалтерські послуги, розсилатиме нагадування про сплату квартплати, про заборгованість на початок місяця, про послуги якими користується та чи інша квартира, інформування про події у житті будинку, а також прозорий звіт по витратах ОСББ.

Створюватиме додаткові послуги з яких зможе заробляти ОСББ, наприклад: прибирання на поверсі, прибирання у квартирі, замовлення їжі, доставка води, послуга домофону, вивіз сміття і таке інше.

Система буде надавати можливість спілкування керівництву ОСББ з мешканцями будинку, а також самим мешканцям спілкуватися між собою. Надаватиме портал для сплати комунальних послуг. Буде можливість створювати заявки на виконання сервісних чи ремонтних робіт для мешканців, наприклад: виклик сантехніка, виклик електрика, виклик майстра по вікнах та дверях, заявка інтернет провайдеру, виклик завгоспа, який зможе щось полагодити у будинку чи ліквідувати аварію.

За додаткову плату користувачі системи зможуть отримувати додаткові послуги, наприклад: послуги відео нагляду, чи послуги служби охорони.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Перед тим як приступити до розробки програмної системи, спочатку необхідно чітко визначитися із функціональними вимогами, тобто вирішити як має поводити себе система, та які функції вона повинна виконувати, для задоволення потреб користувачів.

Веб-застосунок призначений для двох категорій користувачів: власників квартир та персоналу ОСББ. Система дозолятиме їм забезпечувати життєдіяльність ОСББ. Вона повинна містити всю інформацію про ОСББ, його керівництво, всіх проживаючих у будинку вцілому, а також у квартирі, інформацію про прописаних мешканців квартири (це дозволить робити документи про склад родини наприклад), наявність автівки, марку, номер квартири і ім’я її власника. Надаватиме бухгалтерські послуги, розсилатиме нагадування про сплату квартплати, про заборгованість на початок місяця, про послуги якими користується та чи інша квартира, інформування про події у житті будинку, а також прозорий звіт по витратах ОСМД.

Система повинна вміти створювати додаткові послуги з яких зможе заробляти ОСМД, наприклад: прибирання на поверсі, прибирання у квартирі, замовлення їжі, доставка води, послуга домофону, вивіз сміття.

Система має надавати можливість спілкування керівництву ОСББ з мешканцями будинку, а також самим мешканцям спілкуватися між собою. Надаватиме портал для сплати комунальних послуг. Буде можливість створювати заявки на виконання сервісних чи ремонтних робіт для мешканців, наприклад: виклик сантехніка, виклик електрика, виклик майстра по вікнах та дверях, заявка інтернет провайдеру, виклик завгоспа, який зможе щось полагодити у будинку чи ліквідувати аварію.

За додаткову плату користувачі системи зможуть отримувати додаткові послуги, наприклад: послуги відео нагляду, чи послуги служби охорони.

Повинно бути створено систему з декількома рівнями доступу для різних користувачів. Кожен користувач може мати різний рівень доступу до функціоналу системи в залежності від своєї ролі та повноважень. Наприклад, бухгалтер має повний доступ до всіх функцій системи, включаючи додавання, редагування та видалення основних засобів та інформації про квартиру та мешканців. Керівник ОСББ має можливість переглядати звіти про оплату, а також генерувати доступні.

Загальна характеристика системи наводиться нижче:

а) система повинна відображати данні:

1) безпосередньо про основні поняття: квартиру, мешканців, замовлені послуги;

2) про пов’язані поняття: інформація про квартиру та про замовлені послуги, про власника квартири;

3) мати функцію реєстрації;

4) мати функцію авторизації;

5) створення заявок;

6) редагування профілю та управлінню аккаунтом;

7) сплата за послуги;

б) система повинна підтримувати арифметичну обробку даних у вигляді обчислювальних полів: стосовно загальної кількості квартир, кількості мешканців в тій чи іншій квартирі; загальної кількості замовлених послуг; інформація по кожній послузі;

в) система повинна підтримувати сортування, пошук та фільтрацію даних:

г) сортувати власників за ПІБ, сортування інформації по квартирі, сортування по послугам;

д) здійснювати пошук інформації про власника за його повним або частковим ПІБ, про квартиру за номером або ID послуги;

е) здійснювати фільтрацію інформації по сервісам, по квартирам;

ж) система повинна підтримувати додавання нових даних власників квартири, прописаних людей у квартирі, створення послуг, замовлення послуг.

з) система повинна підтримувати можливості редагування інформації про власників квартири, прописаних людей у квартирі, створених послуг, замовлених послуг;

и) система повинна підтримувати можливості вилучення інформації про власників квартири, прописаних людей у квартирі, створених послуг, замовлених послуг з підтримкою режиму підтвердження користувачем видалення інформації про поточний об’єкт;

к) Система повинна підтримувати виконання наступних часто виникаючих запити до БД:

1) отримати статистику по оплаті квартплати (номер квартири, ім’я власника, баланс на даний момент, інформація по оплаті за обрану кількість місяців);

2) отримати статистику топ 5/10 боржників (номер квартири, ПІБ боржника, баланс на даний момент);

3) отримати статистику отримати статистику по оплаті сервісів (номер квартири, ім’я власника, баланс на даний момент, інформація по оплаті за обрану кількість місяців);

4) отримати статистику найпопулярнішого сервісу (назва сервісу, кількість кількість підключень, загальна сума сплати за сервіс);

л) система повинна підтримувати можливість формування довільного запиту до БД на мові SQL з підтримкою користувача інформацією стосовно схеми БД;

м) система повинна підтримувати підготовку та друк наступних звітів:

1) надавати звіт по оплаті за поточний місяць/рік/обраний період плюс повна сума по оплаті за обраний період (відображати ПІБ власника, № квартири, статистика по місяцях за обраний період);

2) звіт по підключеним послугам (відображати № квартири, підключені послуги)

н) реалізувати задачі автоматизації: автоматичне формування таблиці боржників на перше число місяця та розсилка платіжок на електронну пошту.

Веб-додаток повинен підтримувати локалізацію англійською та українською мовами. Система має бути стабільною та надійною. У випадку некоректних дій користувача або виникнення помилки, програма повинна обробити помилку належним чином та запобігти виникненню будь-яких несправностей, а користувач має отримати чітке та зрозуміле повідомлення про помилку, яке пояснить, що сталося і як її можна виправити.

Інтерфейс веб-додатка повинен бути зручним та інтуїтивно зрозумілим для користувачів. Він має адаптуватися до різних розмірів та орієнтацій екрану, сайт повинен дотримуватися загальних шаблонів, які використовуються при створенні інтерфейсу, та використовувати загальноприйняті елементи інтерфейсу.

Для розробки веб-додатка слід використовувати мову програмування JavaScript з розширенням JSX, бібліотеку React.js та платформу Node.js. Для реалізації перекладу буде використано бібліотеку i18next, а для роботи з картами – бібліотеку React Leaflet.

Для моделювання системи слід використовувати сервіс Draw.io та програму Visual Paradigm. Для проектування користувацького інтерфейсу слід застосовувати онлайн-сервіс Figma, а для розробки окремих елементів дизайну – Adobe Illustrator.

2.2 Нефункціональні вимоги

Після визначення функціональних вимог, наступним кроком є формулювання нефункціональних вимог, які не стосуються безпосередньої поведінки системи, але визначають її характеристики та властивості. Ці вимоги забезпечують доступність і зручність використання системи для широкого кола користувачів.

Нижче наведено список нефункціональних вимог, яким має відповідати система:

Кросбраузерність: система повинна коректно працювати в основних браузерах, таких як Google Chrome, Microsoft Edge, Mozilla Firefox, та Opera, щоб забезпечити максимальну доступність для користувачів.

Надійність: система повинна запобігати виникненню помилок та збоїв, бути стабільною, щоб забезпечити безперебійну роботу.

Відмовостійкість: у разі виникнення помилок система повинна належним чином обробляти їх, запобігаючи виникненню несправностей, щоб зберегти систему у робочому стані.

Збереження даних: у випадку помилок система повинна забезпечити збереження введених користувачем даних, щоб уникнути повторного введення.

Інформативність: у разі некоректних дій користувача або виникнення помилки, система повинна надавати чітке та зрозуміле повідомлення про помилку, яке пояснює, що сталося і як виправити помилку.

Швидкість: система повинна забезпечувати швидке завантаження сторінок, щоб зменшити час очікування користувачів. Допустимий час завантаження - до 4 секунд.

Зручність взаємодії: інтерфейс сайту повинен бути зручним та інтуїтивно зрозумілим, використовувати загальноприйняті шаблони та елементи інтерфейсу.

Локалізація: сайт повинен підтримувати локалізацію англійською та українською мовами, щоб забезпечити зручність та доступність для користувачів з різних країн та регіонів.

Адаптивність (пропорції): інтерфейс повинен адаптуватися до різних розмірів та пропорцій екрану, щоб забезпечити зручність використання на різноманітних пристроях.

Адаптивність (орієнтація): інтерфейс повинен підтримувати різні орієнтації екрану для забезпечення зручності використання на різних пристроях, таких як смартфони, планшети та настільні комп'ютери.

Естетика: інтерфейс сайту повинен бути привабливим та естетичним, а дизайн сайту повинен відображати характер проекту.

Вимоги до даних - ІС повинна зберігати свої дані у базі даних MySQL чи сумісній.

Вимоги до інтерфейсу - ІС повинна мати інтерфейс у вигляді веб сторінки.

Вимоги до продуктивності - Система має працювати на системі еквівалентній 1 CPU, 2 Гб оперативної пам’яті.

Вимоги до безпеки – програмний код інформаційної системи та персональні дані користувачів мають бути розміщені на серверах в межах України або Європейського Союзу.

Експлуатаційні вимоги - ІС має функціонувати на базі Microsoft Windows, з моніторами з роздільною здатністю від FHD.

Політичні та юридичні вимоги – система має виконувати Закон України «Про захист персональних даних» от 01.06.2010 № 2297-VI

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування програмного забезпечення

Після визначення всіх вимог до програмної системи, наступним етапом є проектування системи та створення діаграм для відображення різних аспектів системи і кращого розуміння її архітектури.

Для моделювання системи та розробки діаграм використовувалася програма Draw.io, яка є зручним, ефективним і потужним інструментом з багатьма спеціалізованими функціями для створення різних UML-діаграм.

Для кращого розуміння взаємодії користувачів із системою та визначення основних сценаріїв взаємодії була створена Use Case діаграма. На цій діаграмі (див. рис. 3.1) показані всі можливі варіанти використання системи надавачем послуг.

Згідно з діаграмою, надавач послуг може налаштовувати свій профіль, змінюючи таку інформацію, як прізвище, ім’я, дата народження, стать, адреса електронної пошти, номер телефону та фотографія.

Кожен користувач може мати різний рівень доступу до функціоналу системи в залежності від своєї ролі та повноважень. Наприклад, бухгалтер має повний доступ до всіх функцій системи, включаючи додавання, редагування та видалення основних засобів та інформації про квартиру та мешканців. Керівник ОСББ має можливість переглядати звіти про оплату, а також генерувати доступні.

Опис концептів програмної області, їх властивостей та зв'язків між ними зобразимо у вигляді загальної діаграми класів представлено на рисунку 3.2.

Зазначимо перелік характеристик для кожного з наведених понять. Поняття квартири має наступні властивості: номер квартири, ПІБ власника, кількість проживаючих людей, поверх, під’їзд. Поняття підключена послуга включає в себе ID оплати, номер квартири, дата сплати, ID послуги та сплачена сума. Назва або ідентифікатор послуги мають зазначені атрибути: ID послуги, назва пакету, ціна послуги або ціна за одиницю, Провайдер або постачальник послуги.

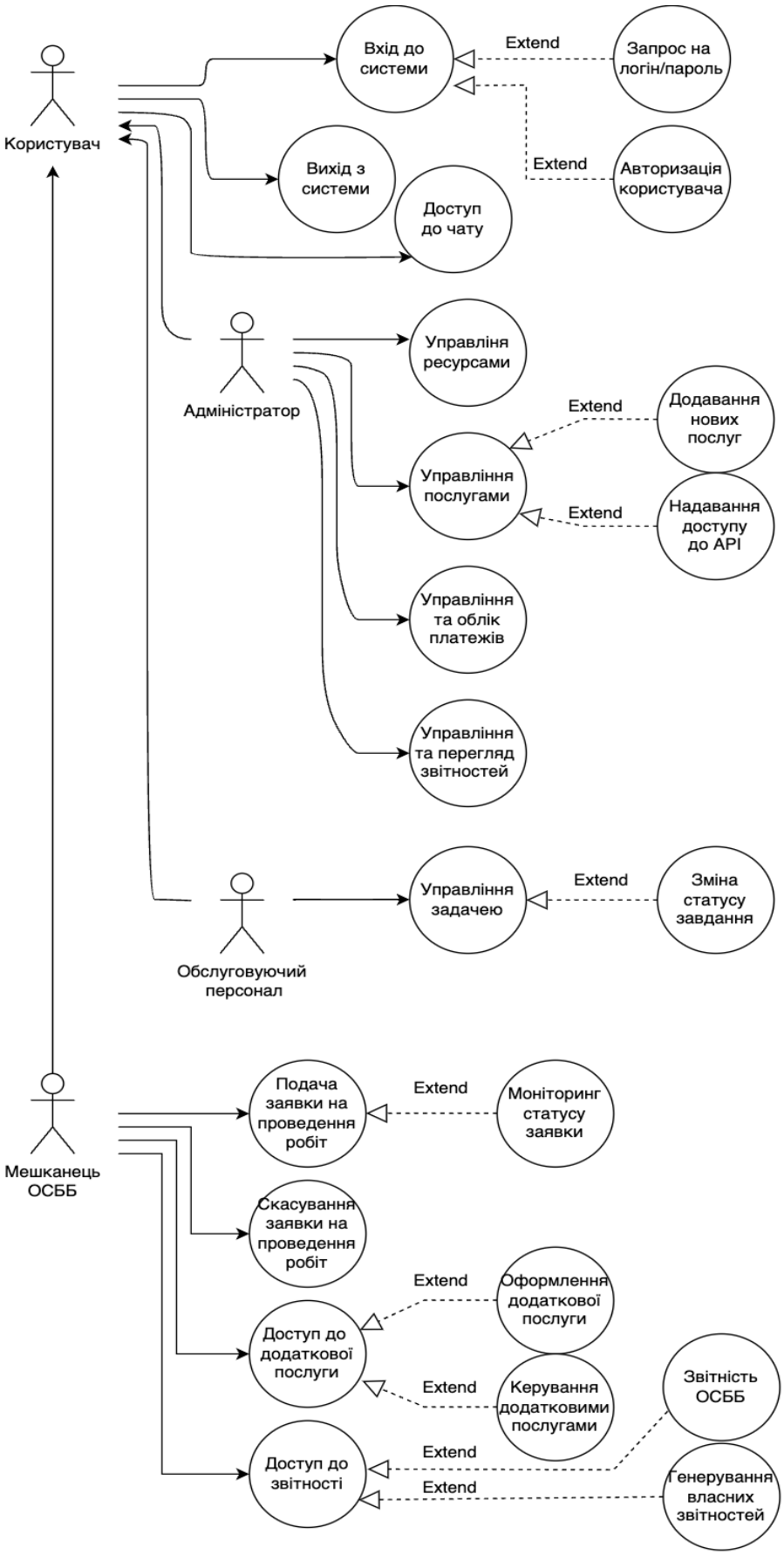


Рисунок 3.1 – Загальна USE-CASE діаграма (рисунок виконаний самостійно)

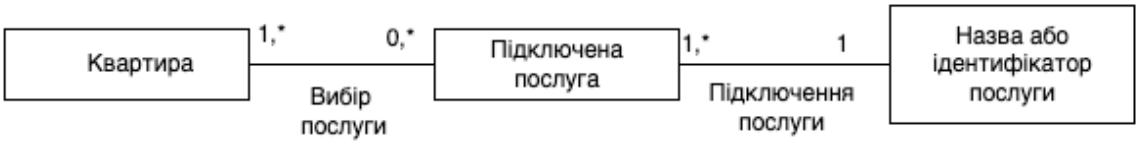


Рисунок 3.2 – Загальна діаграма класів (рисунок виконаний самостійно)

Для спрощення роботи бухгалтера та керуючого ОСББ під час роботи з інформацією вони повинні мати можливість:

- сортувати мешканців за ПІБ, за номером квартири, за поверхом, на під'їздом, за площею квартири; за кількістю проживаючих людей;
- здійснювати пошук інформації про власника за його повним або частковим ПІБ, про квартиру за її номером або іменем власника;
- здійснювати фільтрацію інформації про оплату по даті, по квартирі, по прізвищу власника.
- про склад сім'ї;
- про статистику оплати послуг;
- про кількість та перелік боргів обраної квартири.
- про кількість божників за поверхом або під'їздом.

На діаграмі 3.3 показана автоматизація щомісячної розсилки електронних листів на пошту боржників. Першого числа кожного місяця платформа робить запит у базу даних по балансу всіх квартир у яких баланс менше нуля і потім робить ще один запит на електронно адреси уже тільки боржників, після цього відправляється список на мейл сервер для розсилки боржникам листа з пропозицією сплатити борг.

Опис існуючого документообігу складається з наступних документів:

- звіт «Список боржників»: номер квартири, ПІБ власника, його емейл;
- «Звіт з поточної оплати по сервісах»: номер квартири, назва сервісу, баланс по заборгованості.

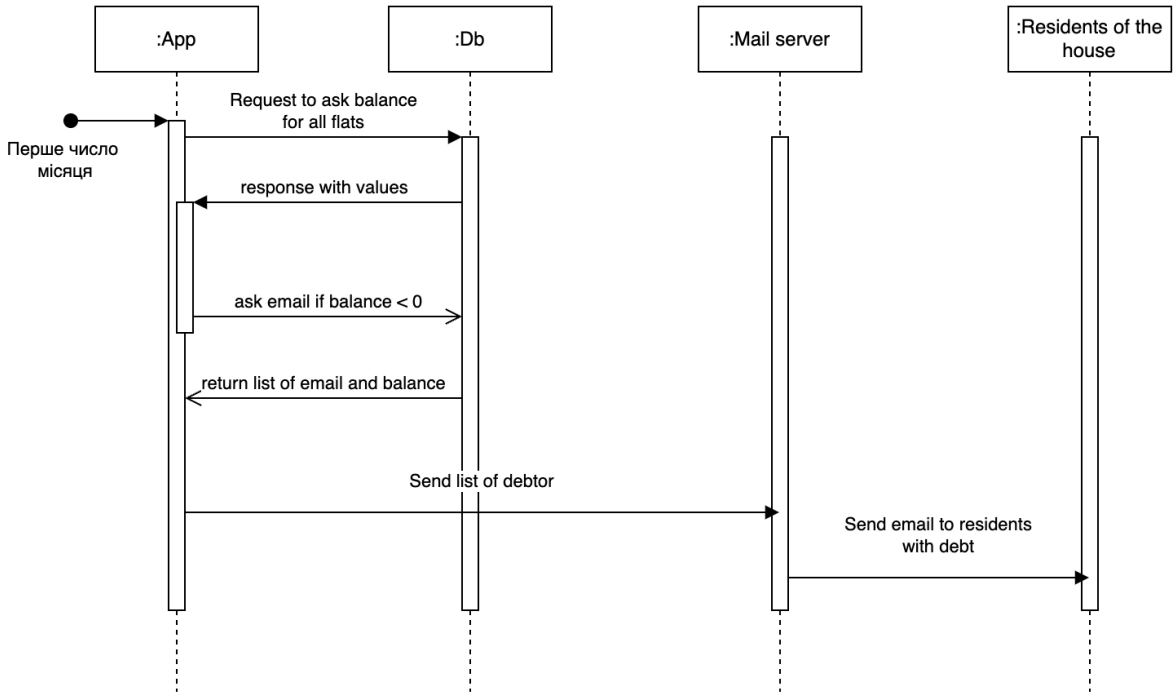


Рисунок 3.3 – Діаграма послідовностей автоматизації (рисунок виконаний самостійно)

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно ідентифікації:

- оскільки кількість квартир вже стала і не змінюється то може мінятися ім’я власника квартири під час перепродажі майна, тому номер квартири ніколи не змінюється бо він унікальний ідентифікатор.
- кожна підключена послуга до квартири ідентифікується сурогатним ідентифікатором ID оплати а також номером квартири, також додається ID;
- кожна існуюча послуга має всій ID як сурогатний ідентифікатор.

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно зв’язків:

- кожна квартира може мати багато підключених послуг і навіть жодної;
- кожна послуга має бути підключена тільки раз до квартири, не може бути дублів, але кількість послуг і самі послуги можуть змінюватися з часом;

З урахуванням ліцензійного ПЗ, що є в ОСББ необхідно дотримуватися наступних обмежень:

– всі документи формувати для редакторів MS Word або MS Excel;

– використовувати СУБД Oracle.

Діаграма послідовності (див. рис.3.4) описує послідовність дій від того моменту коли мешканець залогувався до моменту сплати і отримання підтвердження виконання дії, а саме:

– сторінка логіну та перевірка автентифікації;

– перехід на сторінку сплати та відображення самої сторінки сплати;

– виклик API сплата та саме виконання сплати;

– отримання результату сплати та сповіщення про сплату.

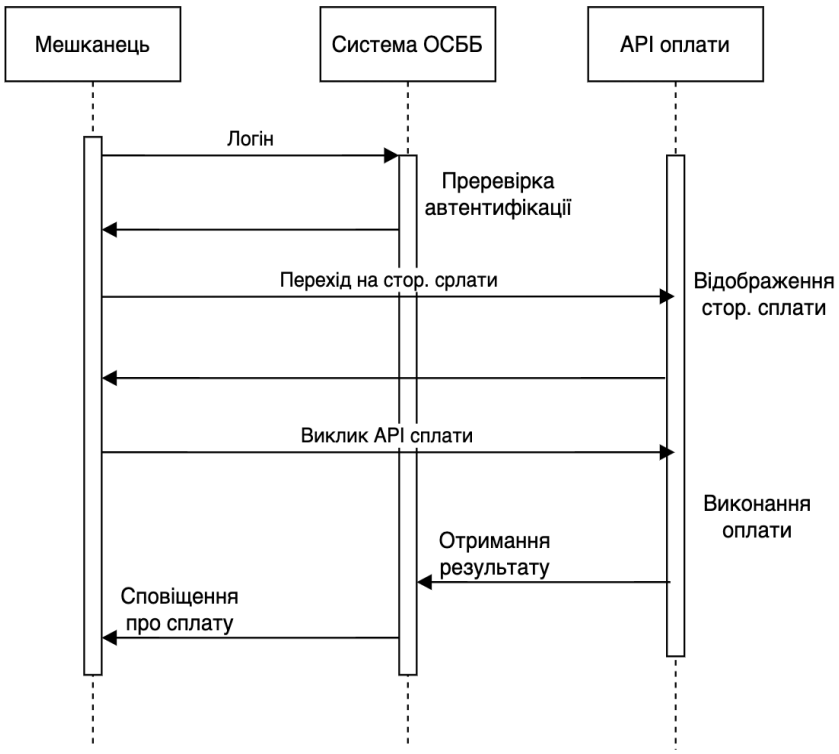


Рисунок 3.4 – Діаграма послідовностей оплати (рисунок виконаний самостійно)

Також цей функціонал зображений на діаграмі діяльності процесу оплати (рис. 3.5) де відображається більш докладніше цей процес з можливими відображення підменю а також меню сплати послуг, а саме:

- шлях починається від моменту вводу облікових даних;
- наступний крок це автентифікація;
- якщо автентифікація завершилася помилкою то шлях починається спочатку, а якщо успіхом то відбувається перехід до наступного кроку з відображенням сторінки оплати;
- з відкриттям сторінки оплати починає працювати API оплати;
- якщо процес оплати закінчується помилко ю то відображається помилка і процес закінчується з помилкою, якщо процес відбувається з успіхом то відображається повідомлення про оплату і також закінчується процес оплати.

Рисунок 3.5 – Діаграма діяльності процесу оплати послуг (рисунок виконаний самостійно)

Також важливим функціоналом повинен стати доступ користувачів до генерації різних видів довідок та виписок які зможуть бути завірені управляючим ОСББ та допоможуть мешканцям під час оформлення якихось державних послуг. Процес отримання таких довідок буде наведений на рисунку 3.6 – 3.8.

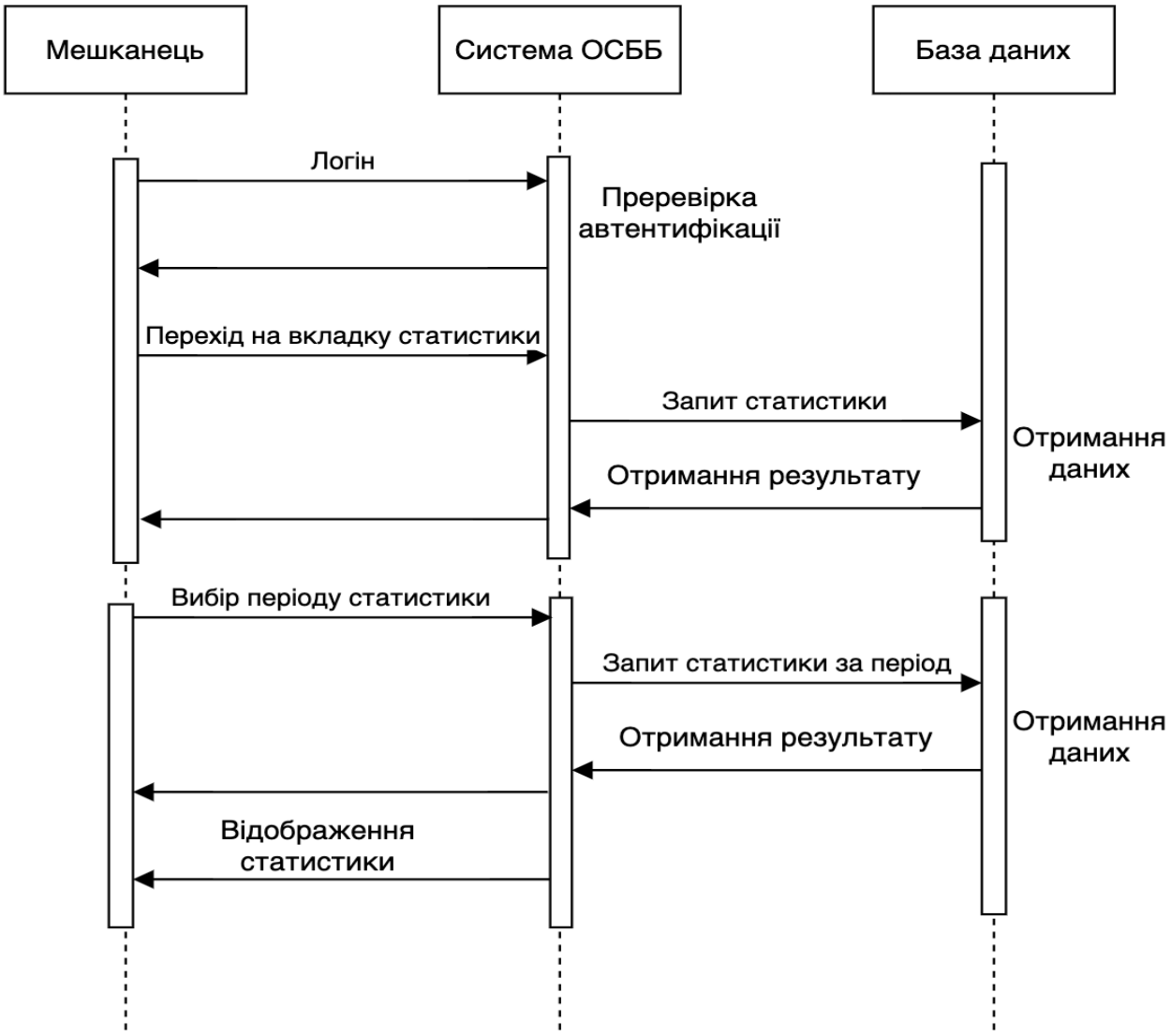


Рисунок 3.6 – Діаграма послідовностей запиту до бази даних (рисунок виконаний самостійно)

На рисунку 3.6 зображена діаграма запиту до бази даних, після моменту автентифікації, за допомогою якої можна зробити запит та отримати різні види статистики:

– запит статистики по користувачу на поточний період;

– запит статистики за обраний період.

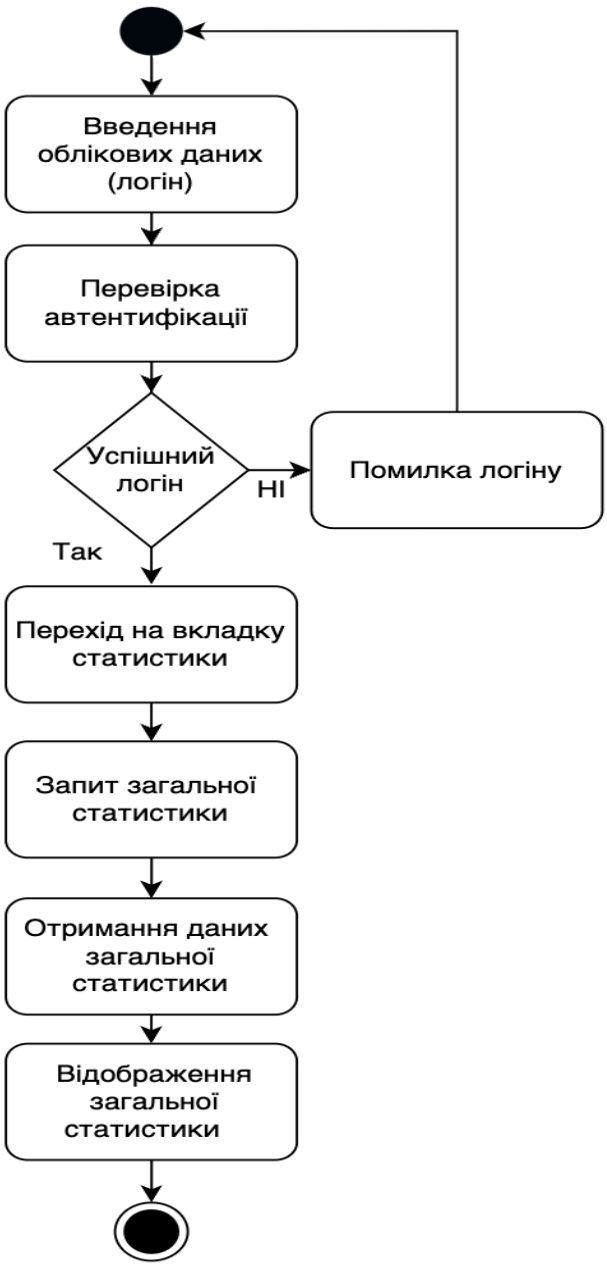


Рисунок 3.7 – Діаграма послідовностей запиту загальної статистики по мешканцю (рисунок виконаний самостійно)

Рисунок 3.7 зображує процес запиту загальної статистики по користувачу (мешканцю будинку):

– починається процес від моменту логіну та перевіки автентифікації;

– якщо логін неуспішний то процес починається спочатку, якщо ж успішний то відбувається перехід на наступний – відкриважться торінка статитсики;

– після запиту загальнох статистики користуач отримує дані і відображається статитика.

Рисунок 3.8 зображує процес отримання статистики за обраний період:

– починається від вибору періоду для отримання статистики та тримання даних;

– якщо виникає необхідність формування звіти і він успішний то формується файл зі звітом, в іншому випадку закінчується процес.

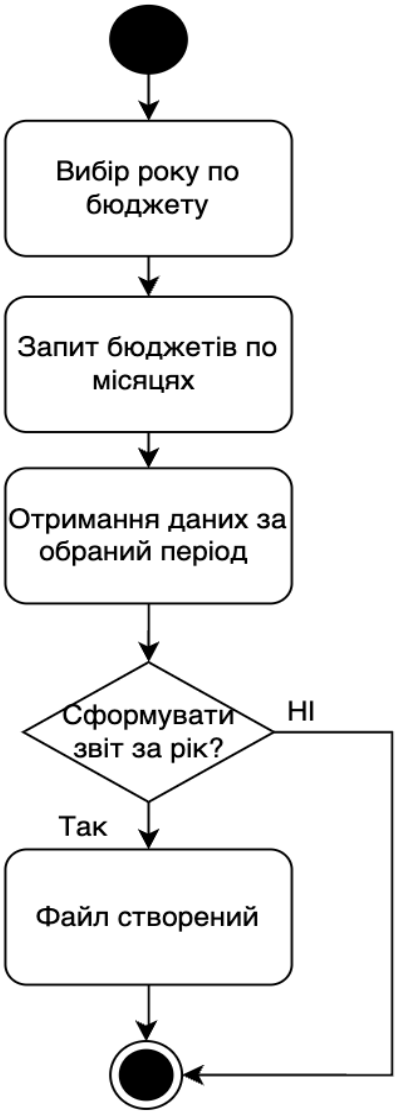


Рисунок 3.8 – Діаграма послідовностей запиту статистики за обраний період по мешканцю (рисунок виконаний самостійно)

В межах роботи системи відображення цієї інформації та функціоналів є критично необхідними для мешканців та керівництва ОСББ.

3.2 Приклади методів та алгоритмів

Під час проектування веб частини було реалізовано функціонал, деякі частини якого представлені нижче.

Цей компонент (див. рис. 3.9) створює структуровану розмітку з текстом та іконками. Його можна використовувати в іншому компоненті або сторінці для відображення цієї інформації.

Frame.jsx 1.29 KiB

```jsx
import React from "react";
import { Chips } from "../Chips";
import { FeatherIconArrowUpRight1 } from "../../icons/FeatherIconArrowUpRight1";
import { FeatherIconMoreVertical } from "../../icons/FeatherIconMoreVertical";
import "./style.css";

export const Frame = () => {
  return (
    <div className="frame">
      <div className="div-2">
        <div className="frame-wrapper">
          <div className="div-wrapper">
            <div className="frame-wrapper-2">
              <div className="div-3">
                <div className="div-4">
                  <div className="text-wrapper">Заявки</div>
                  <FeatherIconArrowUpRight1 className="icon-instance-node" />
                </div>
                <Chips
                  className="chips-instance"
                  divClassName="design-component-instance-node"
                  phosphorIconNotepadColor="#55471D"
                  text="Заявка на розгляді"
                  type="publication"
                />
              </div>
            </div>
          </div>
        </div>
        <div className="div-5">
          <p className="p">Виклик майстра. Не працює ліфт</p>
          <FeatherIconMoreVertical className="icon-instance-node" />
        </div>
      </div>
    </div>
  );
};

```

Рисунок 3.9 – Приклад коду №1 (рисунок виконаний самостійно)

Рисунок 3.10 показує CSS-стилі для компонента Frame:

- .frame: визначає загальні стилі контейнера, включаючи вирівнювання, фон, бордери, радіус бордеру, відступи, і позиціонування.
- .frame .div-2: визначає стилі для дочірнього контейнера, встановлює вирівнювання, дисплей, гнучкість і позиціонування.
- .frame .frame-wrapper: визначає стилі для обгортки, включаючи бордер-радіус, дисплей, гнучкість, відступи, і ширину.

style.css 2.60 KiB

```css
.frame {
  align-items: flex-start;
  background-color: #ffffff;
  border: 1px solid;
  border-color: var(--neutral-30);
  border-radius: 24px;
  display: inline-flex;
  flex-direction: column;
  gap: var(--token-spacing-2-duplicate);
  padding: var(--token-spacing-5-duplicate);
  position: relative;
}

.frame .div-2 {
  align-items: flex-start;
  align-self: stretch;
  display: flex;
  flex: 0 0 auto;
  flex-direction: column;
  gap: var(--token-spacing-2-duplicate);
  position: relative;
  width: 100%;
}

.frame .frame-wrapper {
  align-items: center;
  border-radius: 24px;
  display: flex;
  gap: 68px;
  height: 77px;
  justify-content: space-around;
  position: relative;
  width: 376px;
}
```

Рисунок 3.10 – Приклад коду №2 (рисунок виконаний самостійно)

Рисунок 3.11 показує файл Chips.stories.js, який містить конфігурацію для Storybook, зокрема визначає компонент Chips з різними опціями для type та прикладом використання зі значеннями за замовчуванням.

Chips.stories.js 461 B Blame Edit

```
import { Chips } from ".";

export default {
  title: "Components/Chips",
  component: Chips,
  argTypes: {
    type: {
      options: ["type-7", "publication", "vote", "announcement", "type-6", "surveys", "immediately", "name"],
      control: { type: "select" },
    },
  },
};

export const Default = {
  args: {
    type: "type-7",
    className: {},
    phosphorIconNotepad1Color: "#091E42",
    divClassName: {},
    text: "Публікація",
  },
};

```

Рисунок 3.11 – Приклад коду №3 (рисунок виконаний самостійно)

Код на рисунку 3.12 визначає React-компонент MainMenu, який відображає головне меню з різними пунктами. Компонент використовує кілька іконок із зовнішніх компонентів (PhosphorIcon*) і рендерить дочірні компоненти меню (MenuChild) з різними іконками та текстом, залежно від поточного місця розташування (location).

Опис компонентів:

– імпортуються необхідні залежності з бібліотеки React

– імпортуються іконки з папки ../../icons/.

– імпортується дочірній компонент меню (MenuChild) і стилі (style.css).

Компонент MainMenu:

– location: поточне місце розташування (може бути "dashboard", "news" або інше);

– menuChildIcon, override, menuChildIcon1, ..., menuChildIcon6: іконки для кожного пункту меню (використовуються значення за замовчуванням).

Відображає меню:

– Створює контейнер з класом main-menu і класом, що відповідає location.

– Відображає кілька компонентів MenuChild, передаючи їм відповідні іконки, стани та текст.

```
import PropTypes from "prop-types";
import React from "react";
import { PhosphorIconChatsteardrop5 } from "../../icons/PhosphorIconChatsteardrop5";
import { PhosphorIconChecks } from "../../icons/PhosphorIconChecks";
import { PhosphorIconCrown6 } from "../../icons/PhosphorIconCrown6";
import { PhosphorIconDatabase4 } from "../../icons/PhosphorIconDatabase4";
import { PhosphorIconHouse8 } from "../../icons/PhosphorIconHouse8";
import { PhosphorIconMoney5 } from "../../icons/PhosphorIconMoney5";
import { PhosphorIconNewspaperclipping10 } from "../../icons/PhosphorIconNewspaperclipping10";
import { PhosphorIconRequest4 } from "../../icons/PhosphorIconRequest4";
import { MenuChild } from "../MenuChild";
import "./style.css";

export const MainMenu = ({
  location,
  menuChildIcon = <PhosphorIconHouse8 className="icon-instance-node" color="url(#paint0_linear_38_8469)" />,
  override = <PhosphorIconNewspaperclipping10 className="icon-instance-node" color="url(#paint0_linear_38_8472)" />,
  menuChildIcon1 = <PhosphorIconChatsteardrop5 className="icon-instance-node" color="url(#paint0_linear_38_8475)" />,
  menuChildIcon2 = <PhosphorIconMoney5 className="icon-instance-node" color="url(#paint0_linear_38_8478)" />,
  menuChildIcon3 = <PhosphorIconCrown6 className="icon-instance-node" color="url(#paint0_linear_38_8481)" />,
  menuChildIcon4 = <PhosphorIconRequest4 className="icon-instance-node" color="url(#paint0_linear_38_8484)" />,
  menuChildIcon5 = <PhosphorIconChecks className="icon-instance-node" color="url(#paint0_linear_38_8487)" />,
  menuChildIcon6 = <PhosphorIconDatabase4 className="icon-instance-node" color="url(#paint0_linear_38_8490)" />,
}) => {
  return (
    <div className={`main-menu ${location}`}>
      <MenuChild
        className="menu-child-instance"
        icon={menuChildIcon}
        stateProp="default"
        status={location === "dashboard" ? "current" : "idle"}
        text="Головна Сторінка"
      />
      <MenuChild
        className="menu-child-instance"
        icon={override}
        stateProp="default"
        status={location === "news" ? "current" : "idle"}
        text="Новини"
      />
```

Рисунок 3.12 – Приклад коду №4 (рисунок виконаний самостійно)

3.3 Проектування UI/UX

Після проведення низки досліджень, з пропонованими візуальними концепціями які представлені на рисунках 3.13 - 3.15, використовуючи різні групи людей котрі були розділені за різними віковими групами, статями та різними професіями було розроблено дизайн для системи який би міг задовільнити опитані групи та надати функціонал який би найбільше користувався попитом та був зручний для більшості опитаних респондентів (див. рис. 3.16).

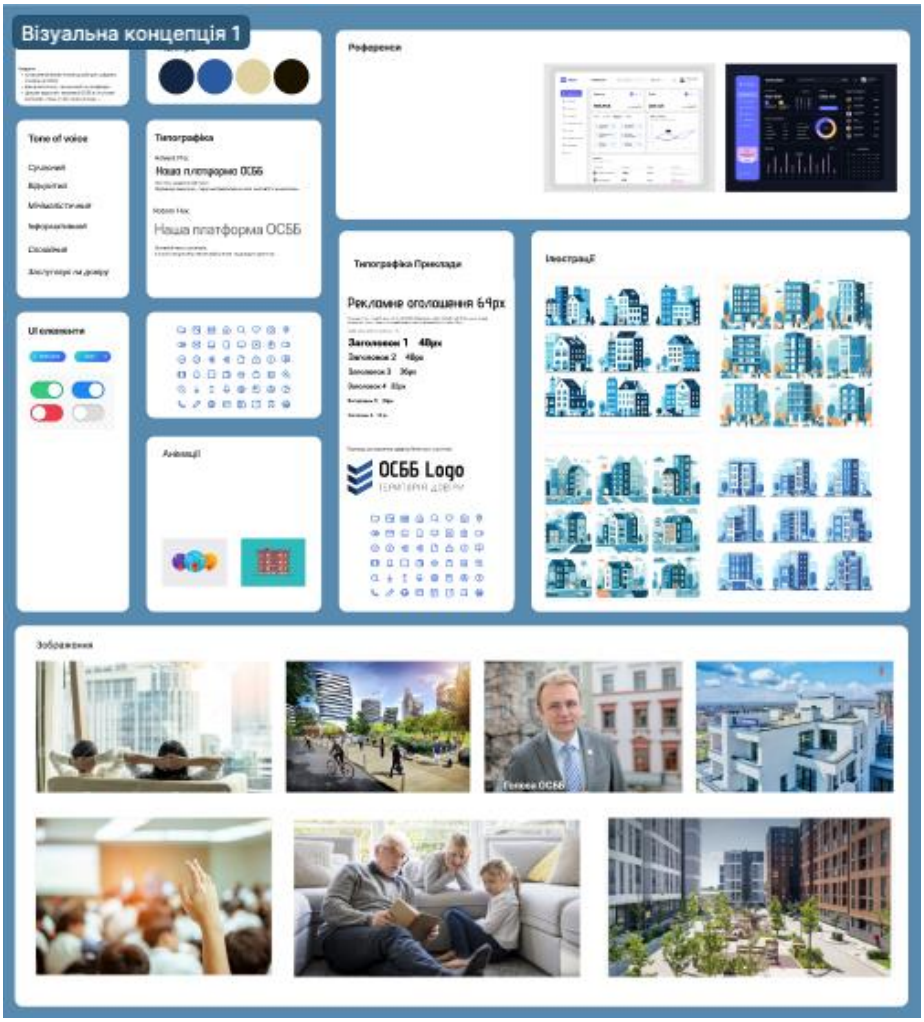


Рисунок 3.13 – Візуальна концепція №1 (рисунок виконаний самостійно)

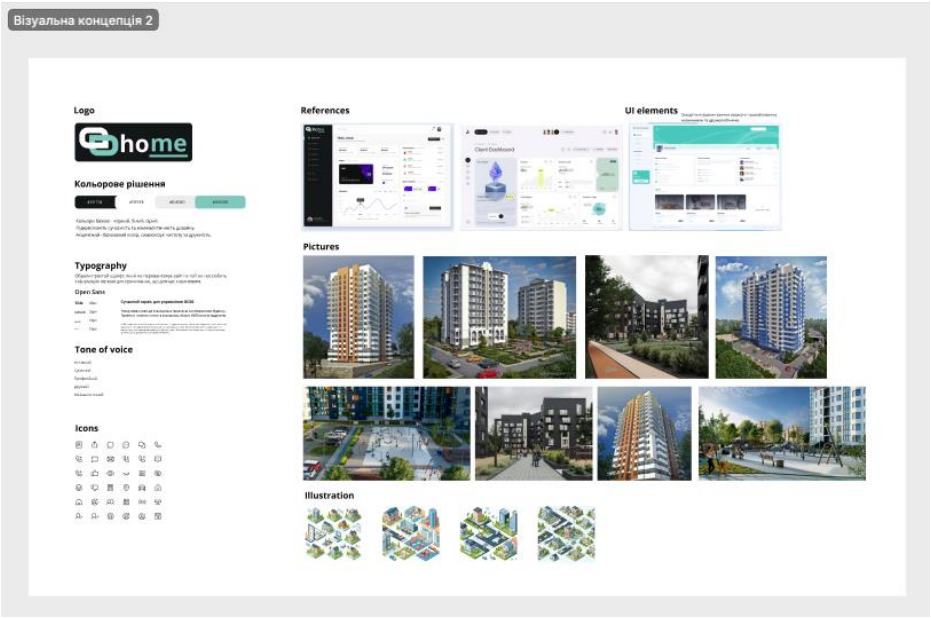


Рисунок 3.14 – Візуальна концепція №2 (рисунок виконаний самостійно)

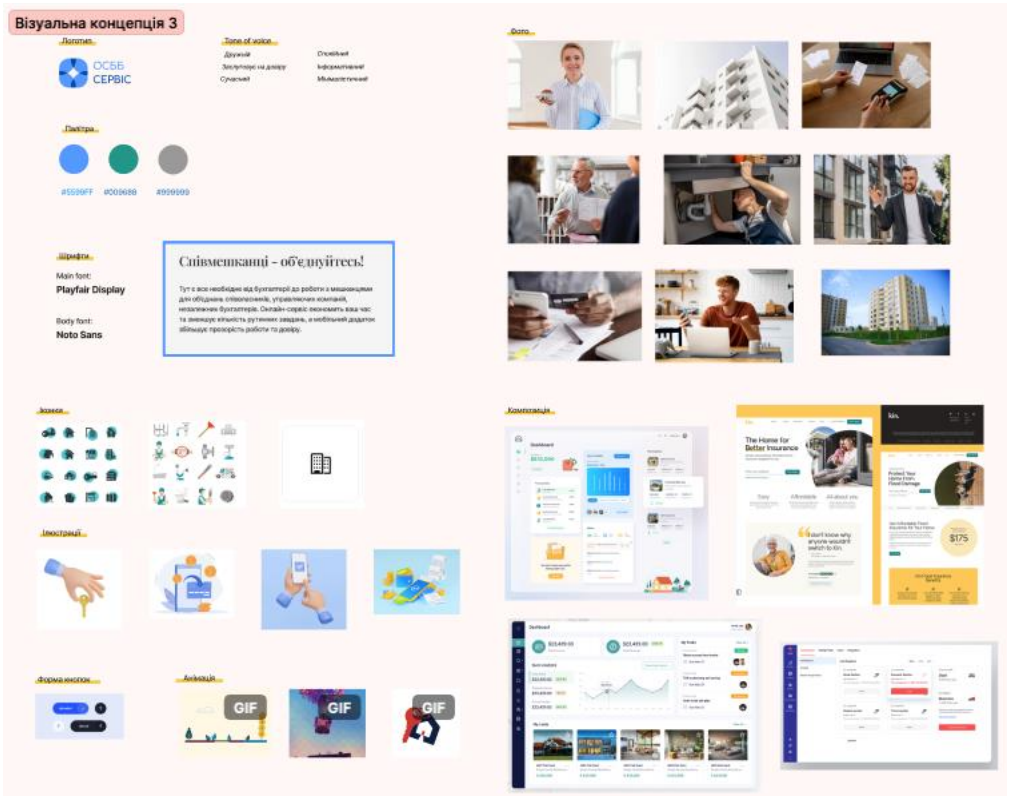


Рисунок 3.15 – Візуальна концепція №3 (рисунок виконаний самостійно)

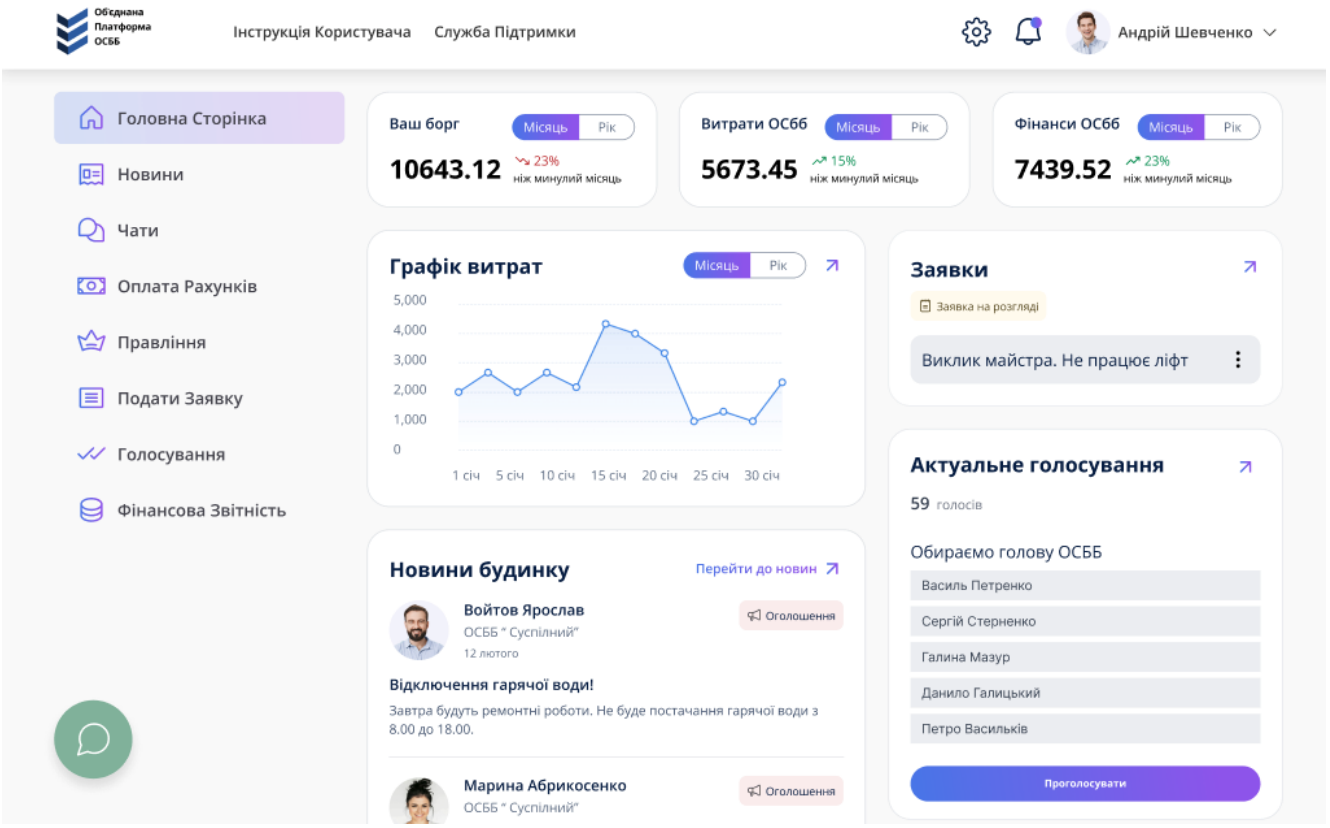


Рисунок 3.16 – Візуалізація дашборду (рисунок виконаний самостійно)

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Маршрутизація та навігація

Один із ключових файлів проекту - це компонент index.jsx, який відповідає за маршрутизацію та відображення різних сторінок на основі URL-адреси. Цей компонент містить елемент Routes, в якому знаходяться елементи Route, що визначають маршрути [10]. В атрибутах кожного елемента Route вказується адреса сторінки (path) та компонент, який повинен бути відображений (element). Спеціальна адреса «*» використовується для обробки будь-якого маршруту, який не співпадає з жодним із зазначених у інших маршрутах, тобто якщо адреса невірна і відповідна сторінка не існує, вона використовується для обробки помилок «404 page not found». Нижче наведено приклад коду компонента:

```
import "../global.css";
import "../styleguide.css";
import React from "react";
import ReactDOM from "react-dom/client";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import { LandingPage } from "./screens/LandingPage";
import { Dashboard } from "./screens/Dashboard";
import ProtectedRoute from './components/ProtectedRoute/ProtectedRoute.js';
import { Screen } from "./screens/Orders";

const router = createBrowserRouter([
 {
  path: "/",
  element: <LandingPage />,
 },
 {
  path: "/dashboard",
  element: (
     <ProtectedRoute>
       <Dashboard />
     </ProtectedRoute>
   ),
 },
 {
  path: "/orders",
```

Описаний вище код є прикладом React-додатку, що використовує React Router для маршрутизації між різними сторінками або компонентами. Нижче наведено детальне пояснення цього коду:

```
import "../global.css";
import "../styleguide.css";
```

У цьому місці імпортуються глобальні CSS стилі, які будуть застосовані до всього додатку.

Далі відбувається імморт бібліотек та компонентів:

```
import React from "react";
import ReactDOM from "react-dom/client";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import { LandingPage } from "./screens/LandingPage";
import { Dashboard } from "./screens/Dashboard";
import ProtectedRoute from './components/ProtectedRoute/ProtectedRoute.js';
import { Screen } from "./screens/Orders";
```

У цьому місці імпортуються необхідні бібліотеки та компоненти. React та ReactDOM використовуються для рендерингу компонентів. createBrowserRouter і RouterProvider з пакету react-router-dom забезпечують маршрутизацію. Імпортуються три основні компоненти: LandingPage, Dashboard та Screen (Orders), а також компонент ProtectedRoute, який, ймовірно, використовується для захисту маршруту.

Далі відбувається створення маршрутів:

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <LandingPage />,
  },
  {
    path: "/dashboard",
    element: (
      <ProtectedRoute>
        <Dashboard />
      </ProtectedRoute>
    ),
  },
  {
    path: "/orders",
    element: <Screen />,
  },
]);
```

У цьому місці створюється маршрутизатор з трьома маршрутами:

/ - маршрутизує на компонент LandingPage.

/dashboard - маршрутизує на компонент Dashboard, але цей маршрут захищений компонентом ProtectedRoute, що означає, що доступ до цього маршруту може бути обмежений (наприклад, вимагає аутентифікації).

/orders - маршрутизує на компонент Screen, що відображає замовлення.

```
const app = document.getElementById("app");
const root = ReactDOM.createRoot(app);
root.render(<RouterProvider router={router} />);
```

У цьому місці створюється кореневий елемент React у DOM з ідентифікатором app. Потім цей елемент використовується для рендерингу всього додатку за допомогою RouterProvider, який надає створений маршрутизатор router.

Загалом цей код реалізує базову структуру React-додатку з використанням маршрутизації для переходу між різними сторінками або компонентами.

Загальна структура реалізації веб частини зображена на рисунку 4.1.

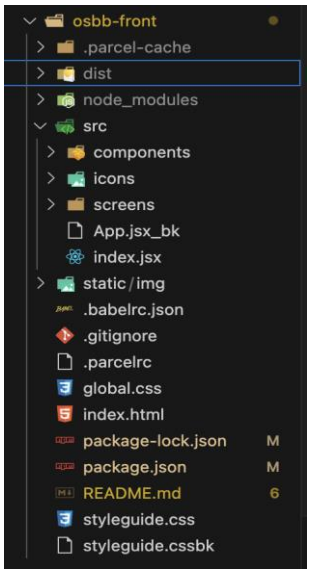


Рисунок 4.1 – Загальна структура веб застосунку (рисунок виконаний самостійно)

4.2 Сторінки та компоненти

Після налаштування маршрутизації було розпочато роботу над створенням основних сторінок та компонентів додатку. Спочатку були розроблені такі

сторінки: головна (лендинг), сторінка з умовами користування та сторінка помилки 404. У головну сторінку була додана функція логауту. На зображеннях нижче можна побачити приклади цих сторінок.

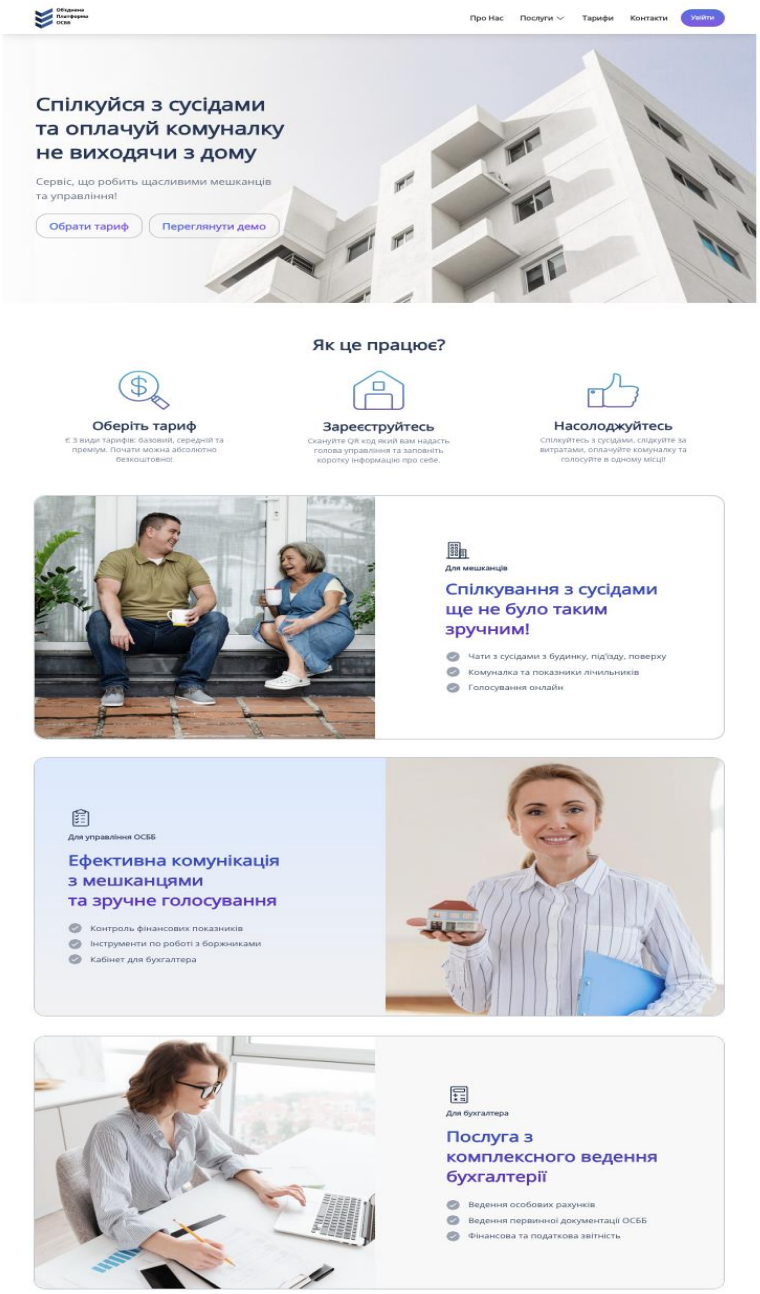


Рисунок 4.2 – Lending page частина 1 (рисунок виконаний самостійно)

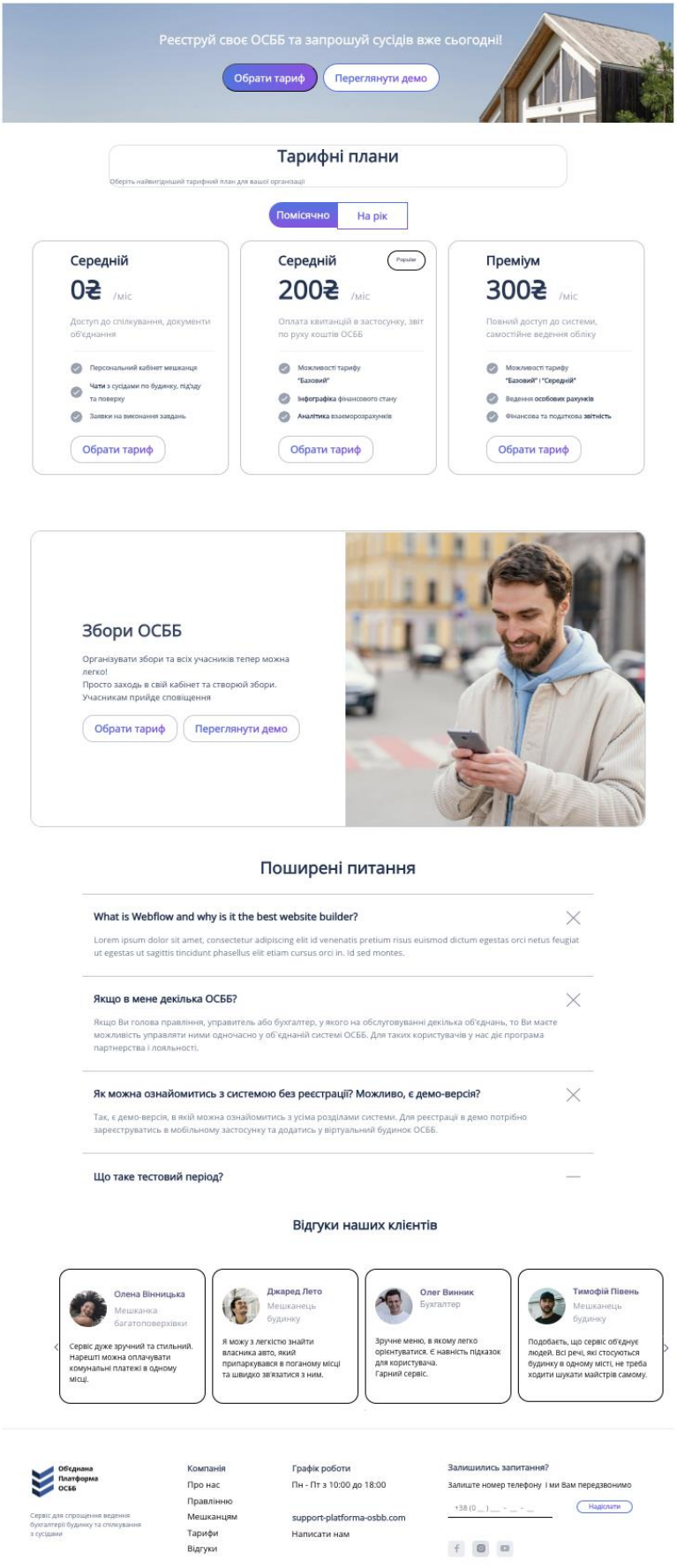


Рисунок 4.3 – Lending page частина 2 (рисунок виконаний самостійно)

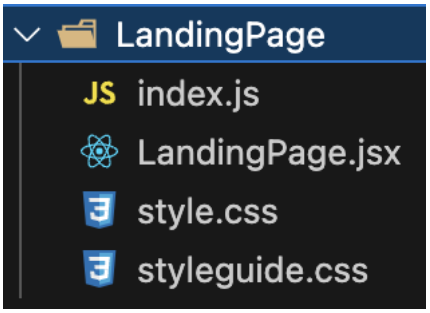


Рисунок 4.4 – Структура Lending page (рисунок виконаний самостійно)

А також частина коду також представлена нижче:

```
export const LandingPage = () => {
 return (
  <div className="landing-page">
   <div className="div-3">
    <div className="overlap">
     <div className="group-6">
      <img className="image-4" alt="Image" src="/img/image-1767.png" />
      <div className="overlap-group-2">
       <img className="image-5" alt="Image" src="/img/image-1769.png" />
       <img className="image-6" alt="Image" src="/img/image-1770.png" />
      </div>
     </div>
     <div className="call-to-action">
      <div className="block-all-to-action">
       <p className="heading">Реєструй своє ОСББ та запрошуй сусідів вже сьогодні!</p>
       <div className="button-set">
        <StateDefaultType
         className="state-default-type-2"
         divClassName=""
         text="Обрати тариф"
         visible={false}
         visible1={false}
         onButtonClick={() => {location.href="#tariffs"}}
        />
        <StateDefaultType
          className="state-default-type-3"
          divClassName=""
          text="Переглянути демо"
          visible={false}
          visible1={false}
        />
       </div>
      </div>
     </div>
```

Також наступним кроком було розроблено функціонал авторизації (логін та логаут з системи) до системи управління ОСББ для користувачів. Цей функціонал відображений на рисунку 4.5 – 4.6.

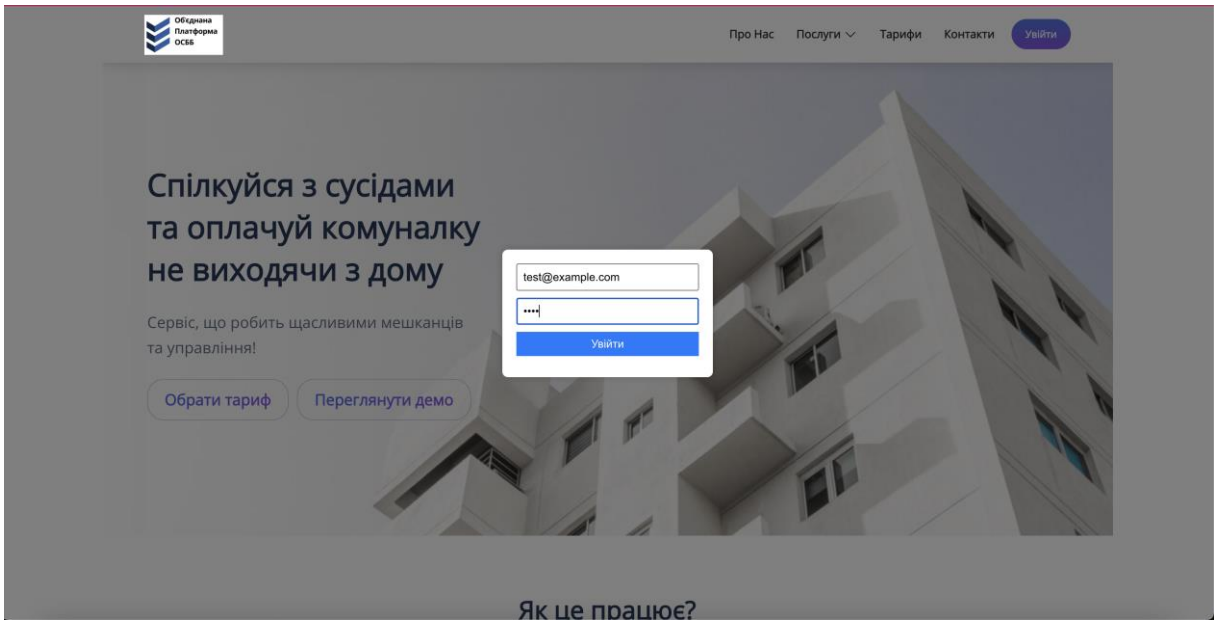


Рисунок 4.5 – Сторінка логіну (рисунок виконаний самостійно)

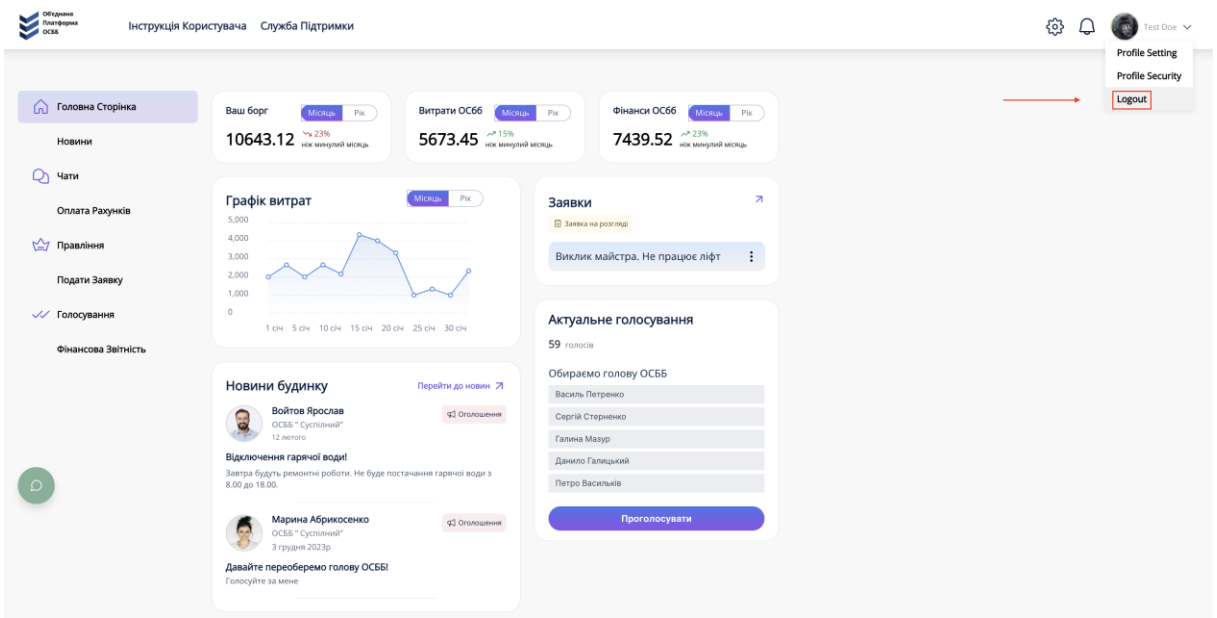


Рисунок 4.6 – Сторінка логауту (рисунок виконаний самостійно)

Вся інформація про юзера зберігається на бекенд частині в базі даних. Під час реєстрації нового користувача данні про нього зберізаються у базі даних.

На рисунку 4.7 зображена цей функціонал адміністратора програми.

Рисунок 4.7 – Сторінка адміністратора (рисунок виконаний самостійно)

А сам процес реєстрації складається з декількох кроків в яких користувачу запропонавано внести деяку інформацію про себе. По закінченню цього процесу реєстрауції створюється новий користувач (див. рис.4.8 – 4.12).

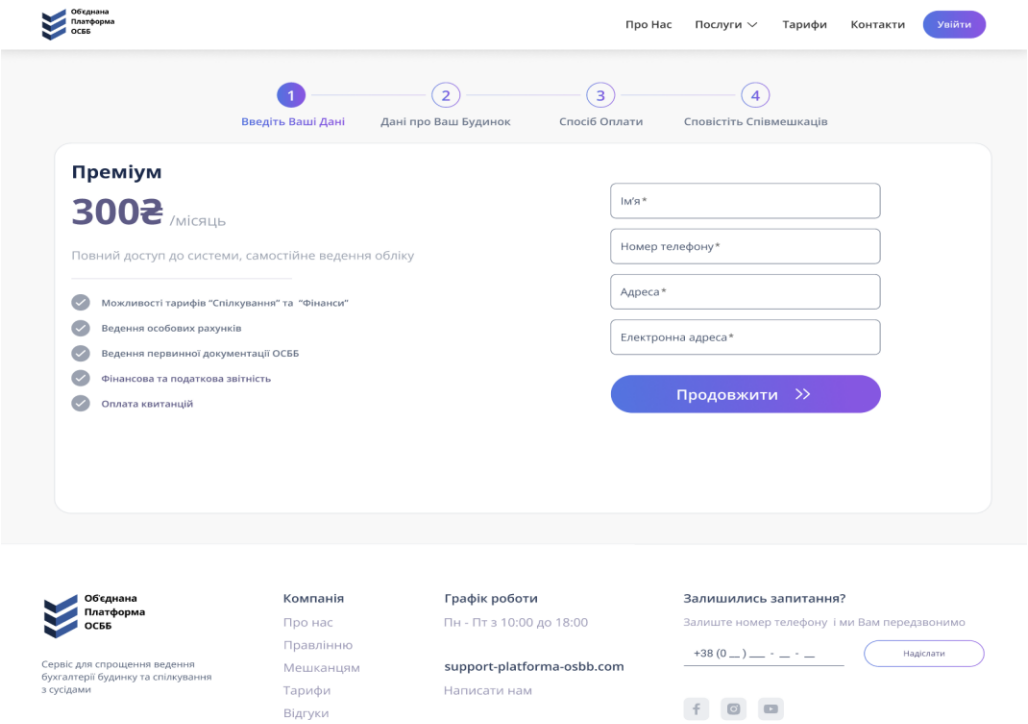


Рисунок 4.8 – Процес реєстрації сторінка 1 (рисунок виконаний самостійно)

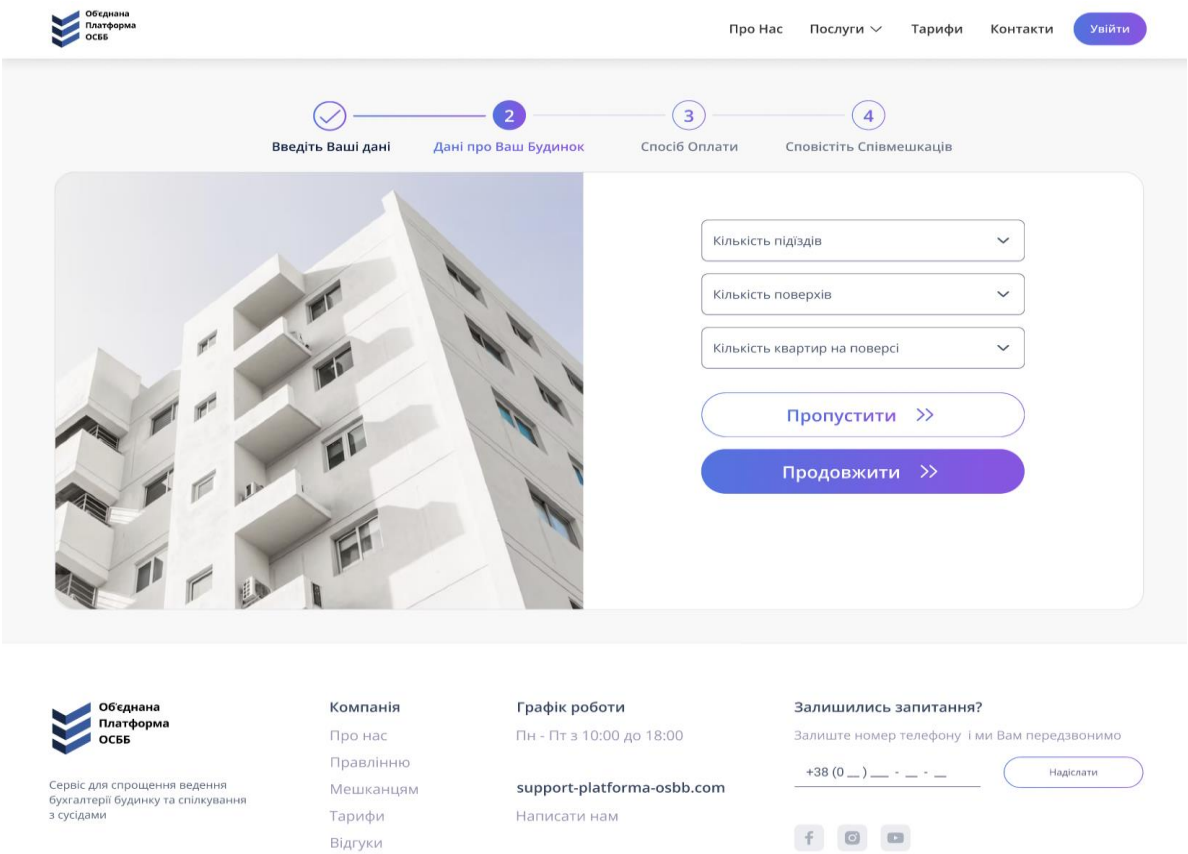


Рисунок 4.9 – Процес реєстрації сторінка 2 (рисунок виконаний самостійно)

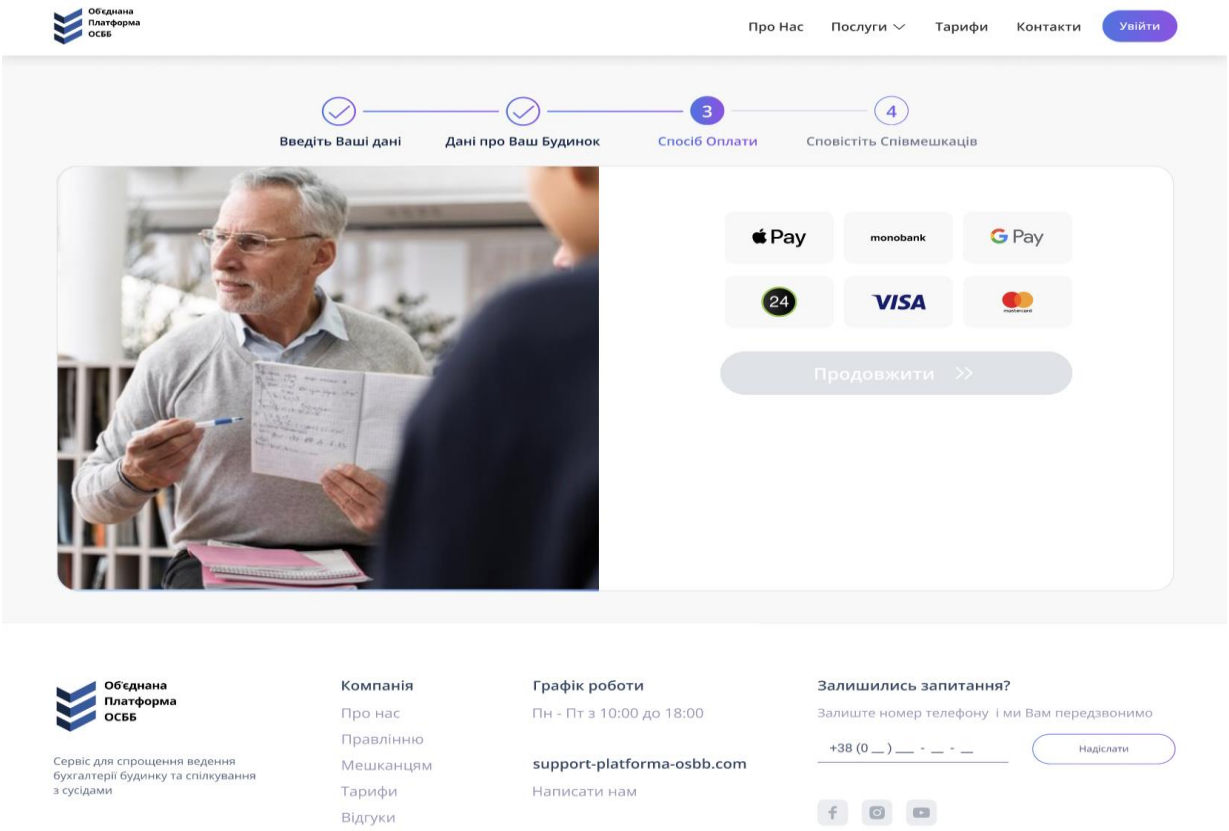


Рисунок 4.10 – Процес реєстрації сторінка 3 (рисунок виконаний самостійно)

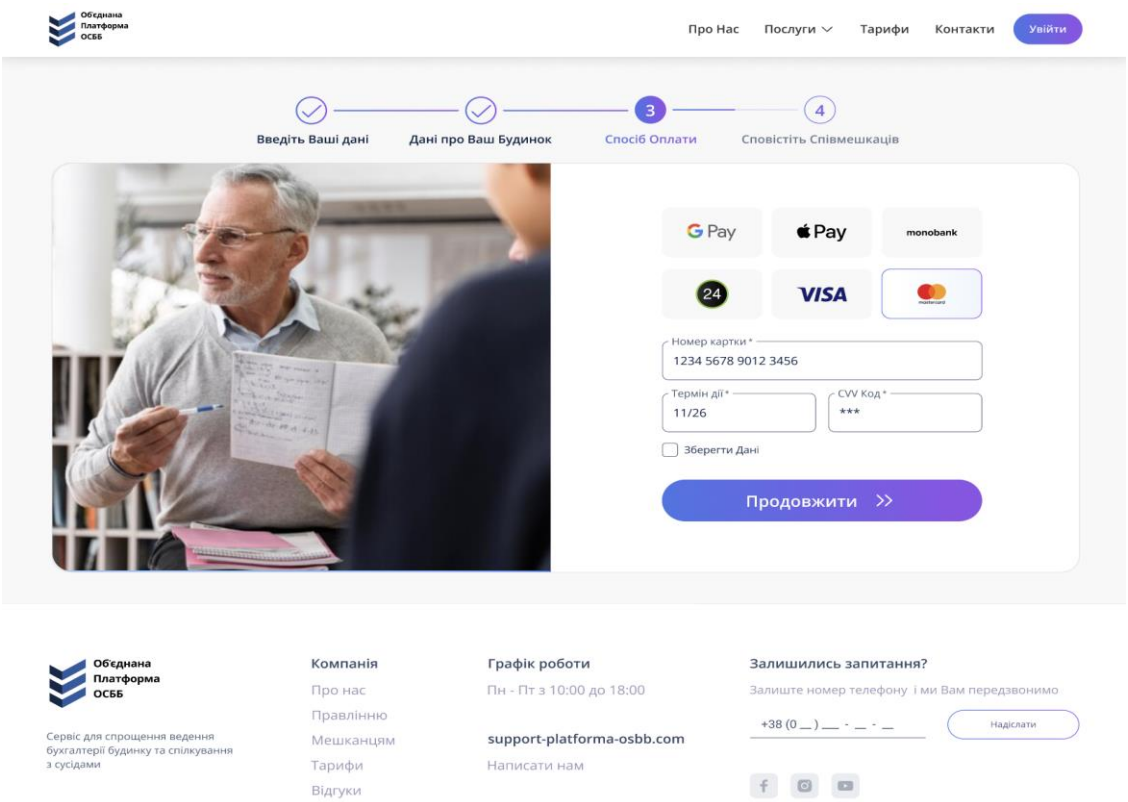


Рисунок 4.11 – Процес реєстрації сторінка 3.1 – вибір методу оплати (рисунок виконаний самостійно)

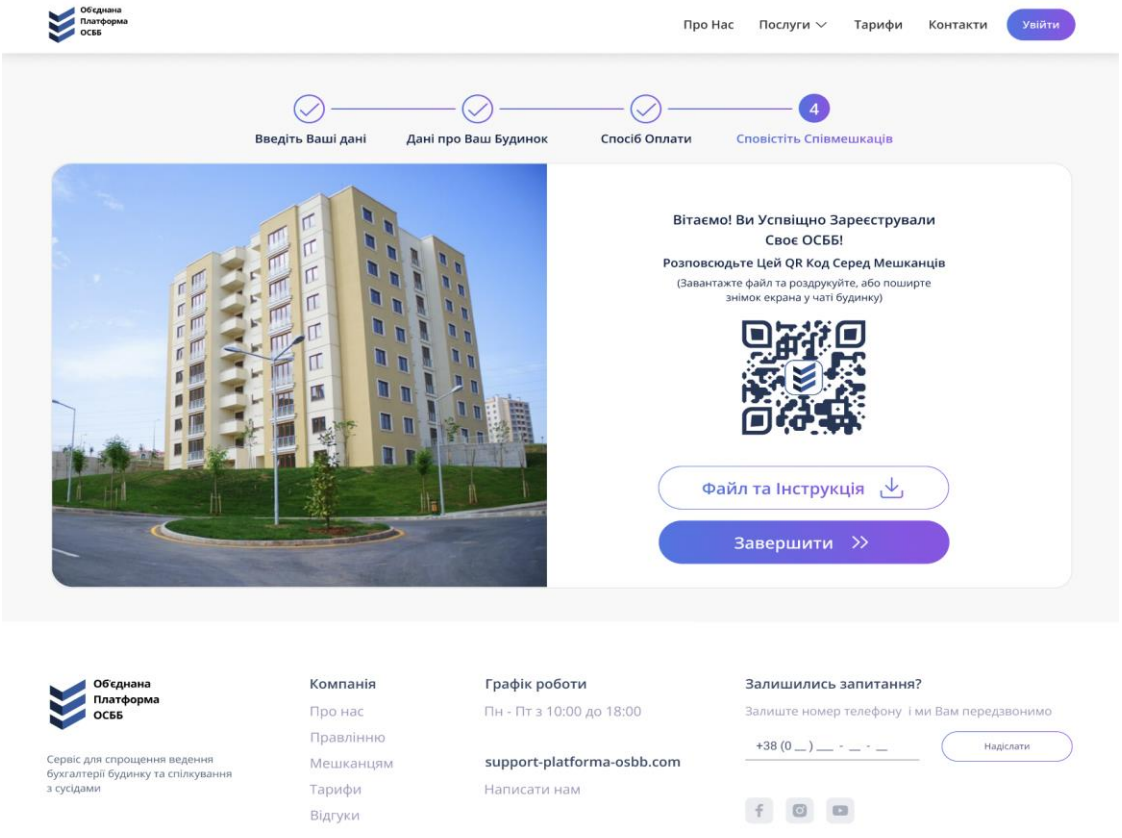


Рисунок 4.12 – Процес реєстрації сторінка 4 (рисунок виконаний самостійно)

Після того як користувач потрапин нв головну сторінку у нього з'являється доступ до інформації про заборгованість якщо така є та статистичну різницб у відсоткаж у порівнянні з минулим місяцем або попереднім роком, доступ до витрат ОСББ також за місяць або рік та статистикою для цієї інформації у порівнянні з попереднім періодом, інформація про акумульовані ресерси на рахунку ОСББ також з вибором періоду та статистикою з порівнянням минулих періодів. Також відображається графік витрат по квартирі, меню з заявками, актуальним голосуванням, новинами будинку а також сторінками з новинами, чатом мешканців, оплатою рахунків, інформацією про правління, подачею заявки на обслуговуваввня або ремонт, голосування та фінансова звітність – рисунок 4.13.

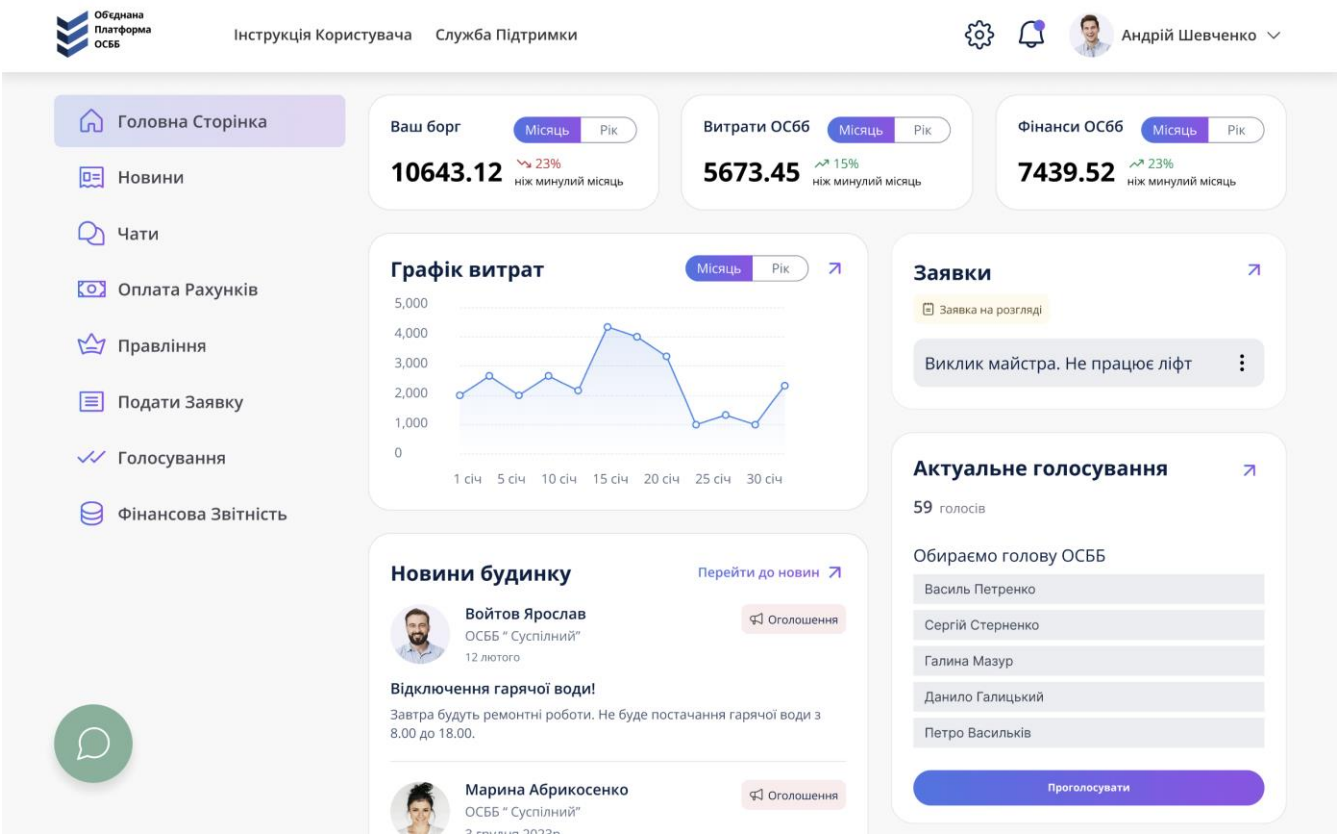


Рисунок 4.13 – Вигляд головної сторінки (рисунок виконаний самостійно)

Розглянемо кожну вкладку окремо та опишемо основні елементи меню на сторінках. Вкладка «Новини» містить інформацію про новини у будинку які автоматично відображаються в хронологічному порядку – найновіші зверху,

найстаріші знизу, також є можливість пошуку серед новин та фільтрація по дататах та типах публікацій,

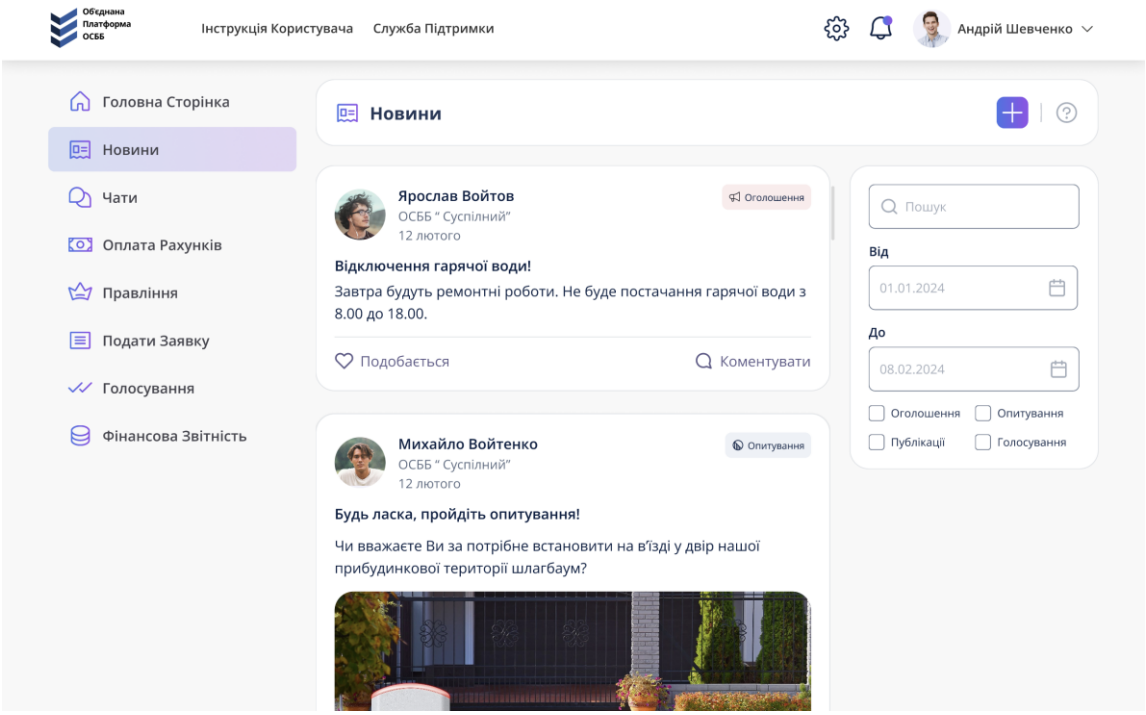


Рисунок 4.14 – Вигляд сторінки «Новини» (рисунок виконаний самостійно)

Наступною сторінкою є сторінка «Чати» яка дозволяє спілкуватися мешканцям між собою (див.рис. 4.15 – 4.16)

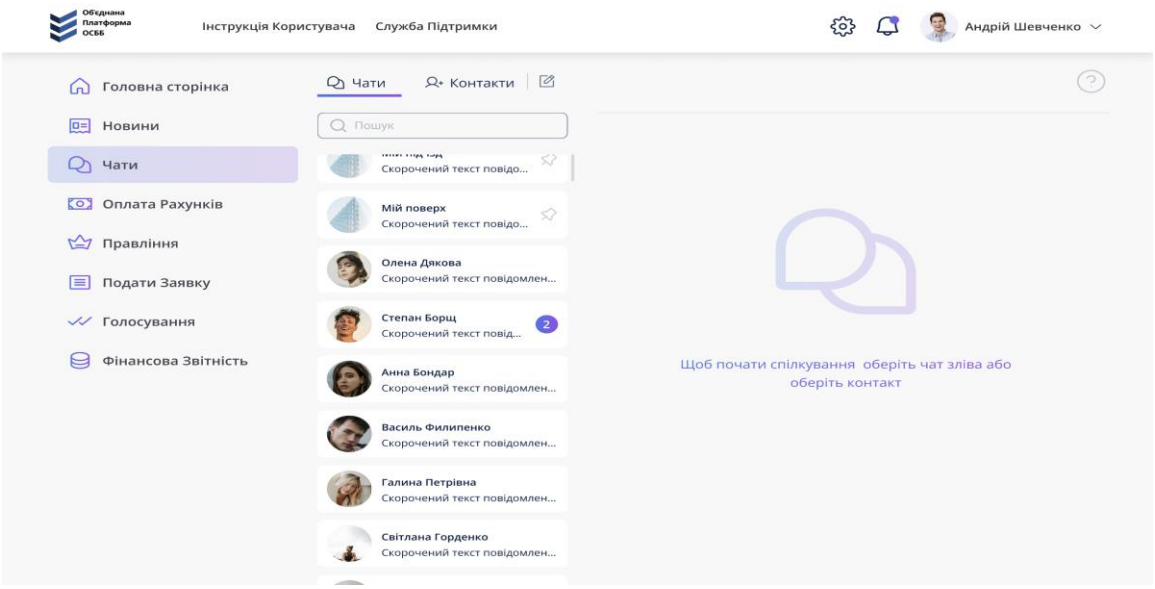


Рисунок 4.15 – Вигляд сторінки «Чати» - 1 (рисунок виконаний самостійно)

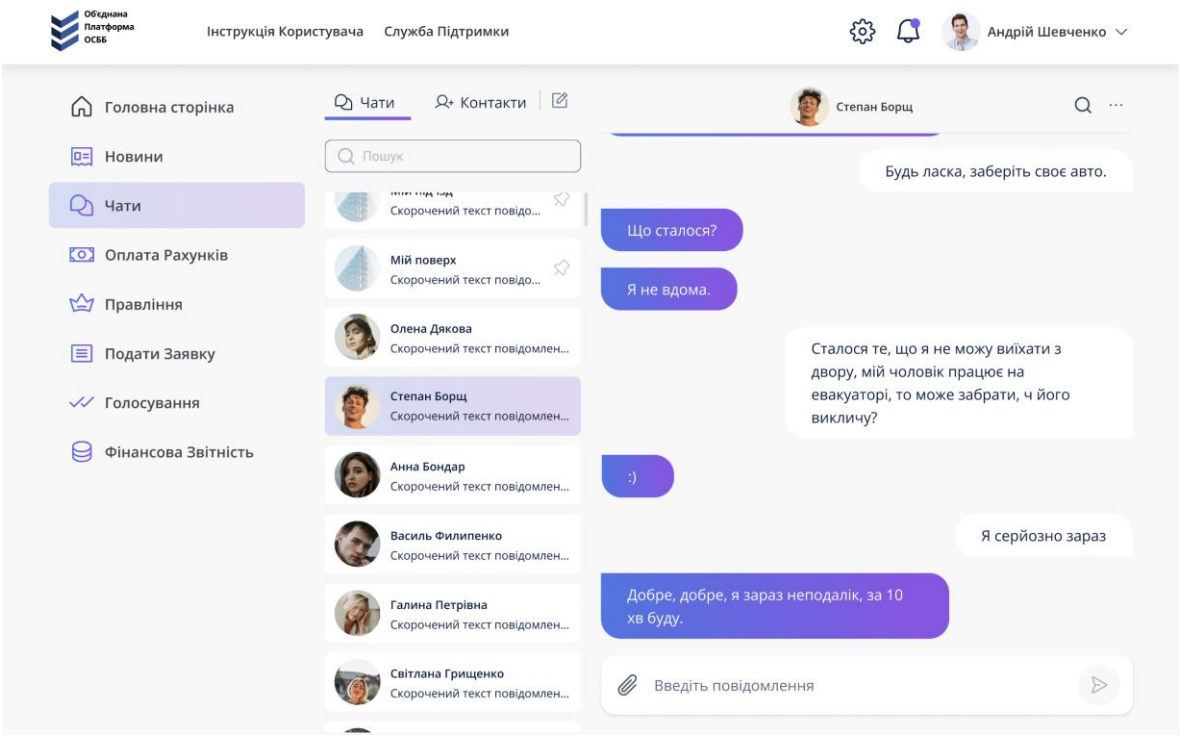


Рисунок 4.16 – Вигляд сторінки «Чати» - 2 (рисунок виконаний самостійно)

Наступним функчіоналом платформи являється владка сплати рахунків. Вона дозволяє сплатити рахунки як окремо по послугам так і одразу зробити загальну проплату, а також вибір способу оплати. Функціонал сторінки коказаний на рисунках 4.17 – 4.19.

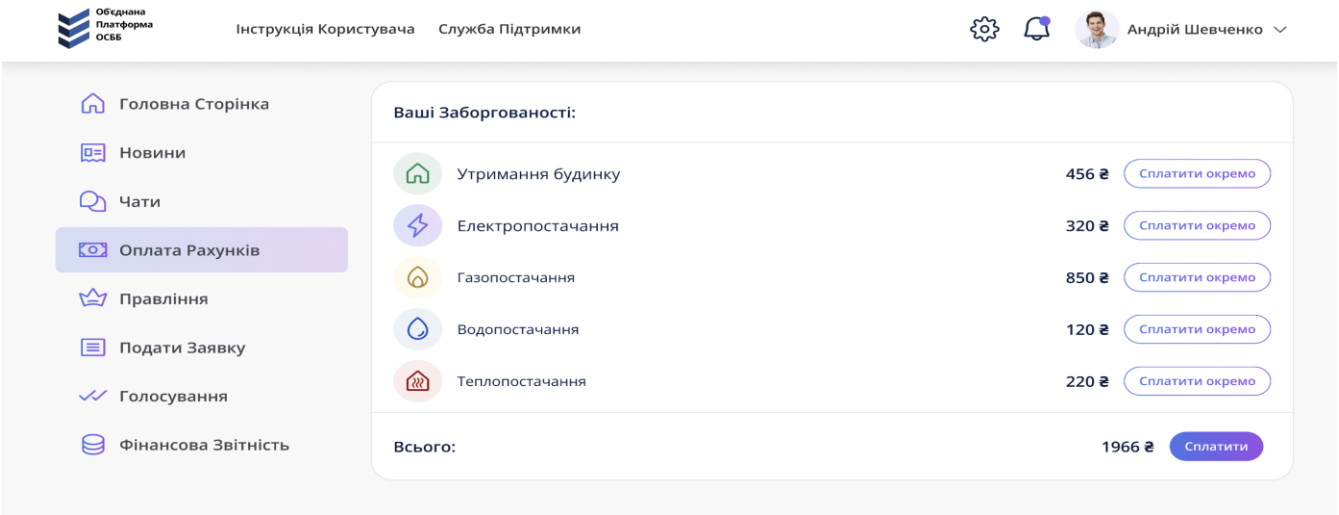


Рисунок 4.17 – Вигляд сторінки «Сплати рахунків» - 1 (рисунок виконаний самостійно)

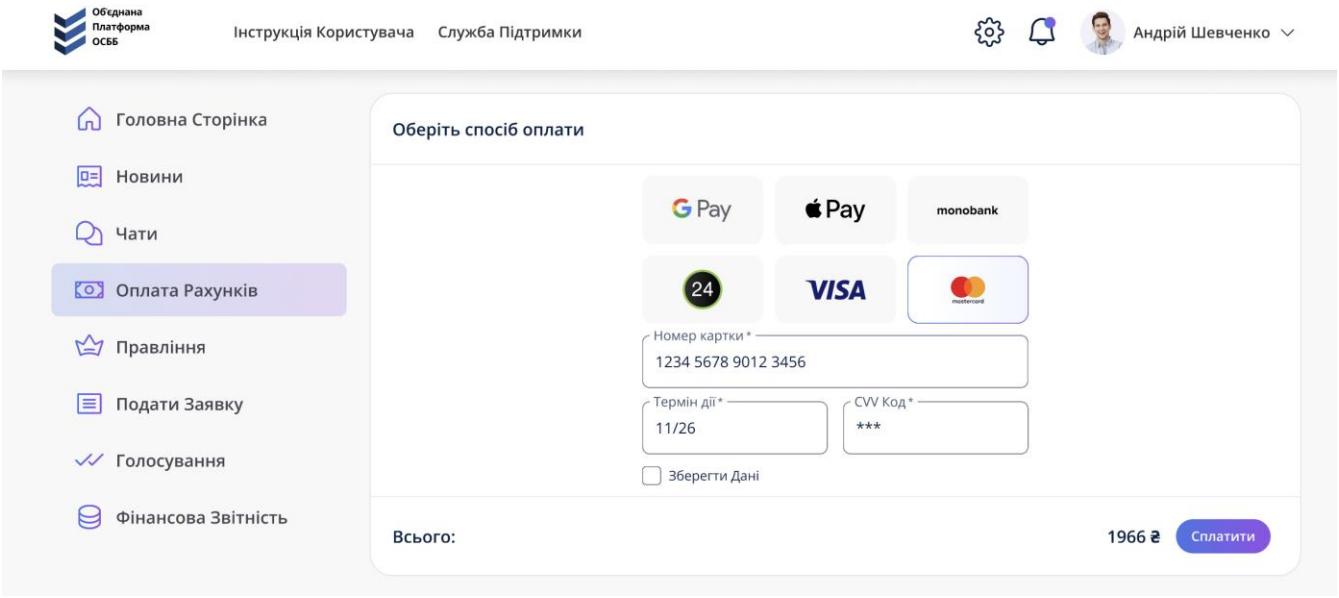


Рисунок 4.18 – Вигляд сторінки «Сплати рахунків» - 2 (рисунок виконаний самостійно)

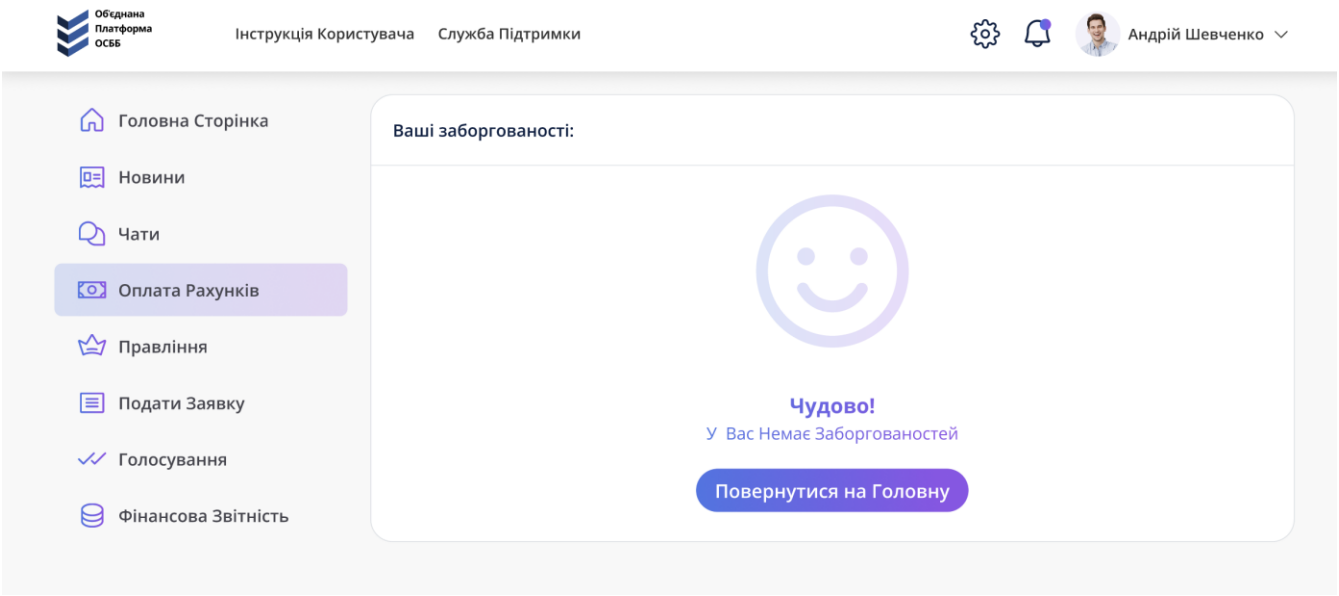


Рисунок 4.19 – Вигляд сторінки «Сплати рахунків» - 3 (рисунок виконаний самостійно)

Далі сторінка яка називається «Правління» і відображає інформацію про правління ОСББ. На цій сторнінці користувач може отримати інформацію а також зв'язатися з людьми які обрані до правління ОСББ. Цей функціональність представлена на рисунку 4.20.

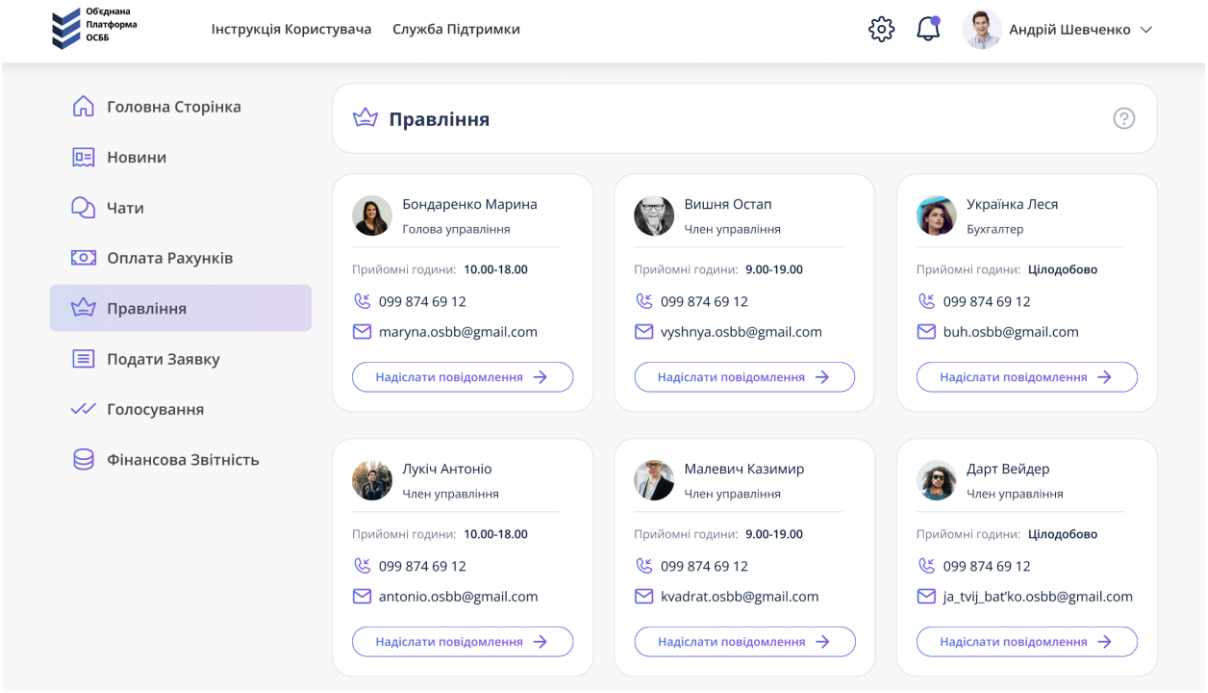


Рисунок 4.20 – Вигляд сторінки «Правління» (рисунок виконаний самостійно)

Одним із основних функціоналів є можливість подавати заявки до ОСББ, цей функціонал реалізований у вкладці «Подати заявку». Після того як заявка створена вона починає відображатися на цій сторінці до якої має доступ будь-хто з мешканців і будьхто може відслідковувати статуси по ній. Функціонал створення заявки та її відображення показаний на рикунках 4.21 – 4.24.

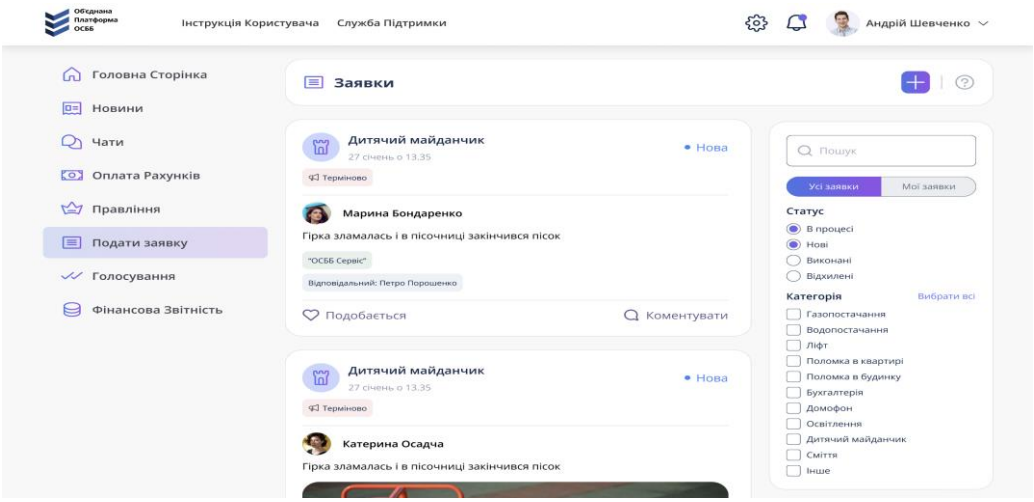


Рисунок 4.21 – Вигляд сторінки «Подати заявку» - 1 (рисунок виконаний самостійно)

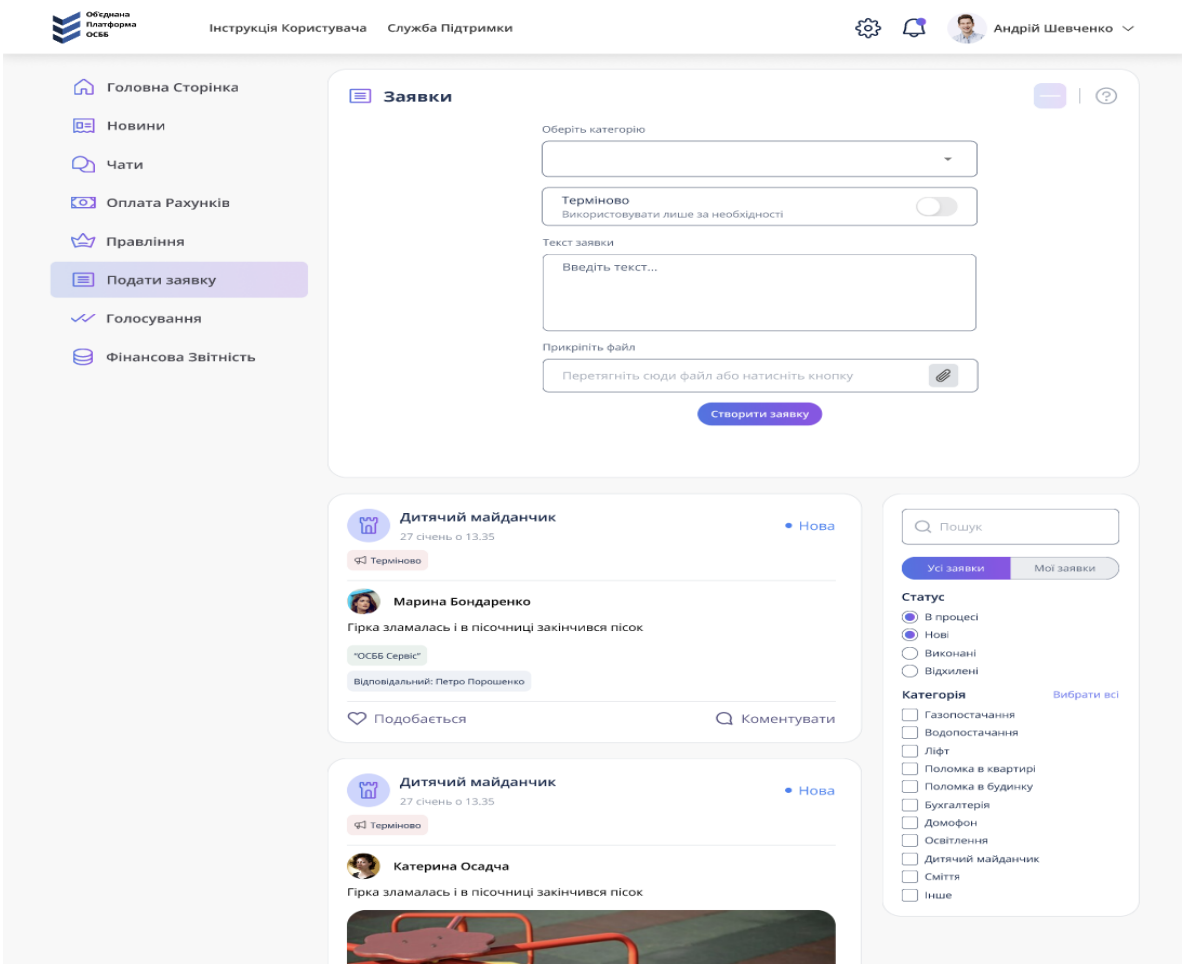


Рисунок 4.22 – Вигляд сторінки «Подати заявку» - 2 (рисунок виконаний самостійно)

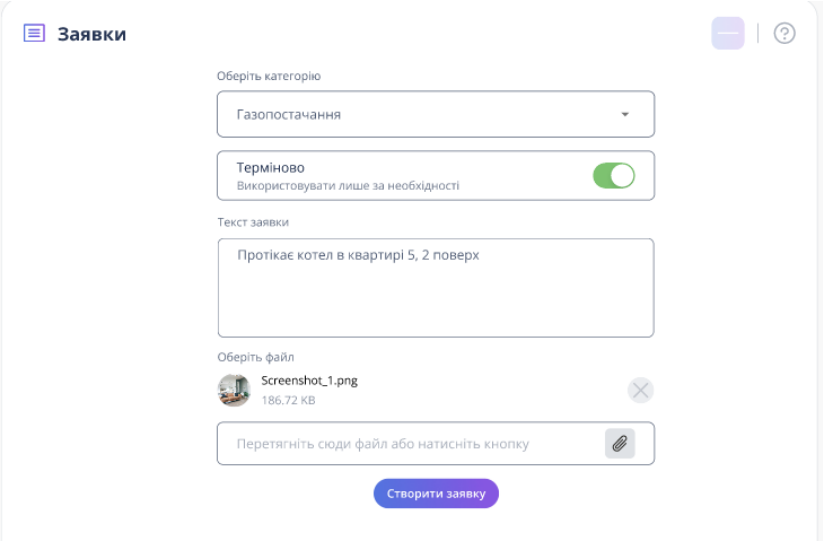


Рисунок 4.23 – Вигляд сторінки «Подати заявку» - 3 (рисунок виконаний самостійно)

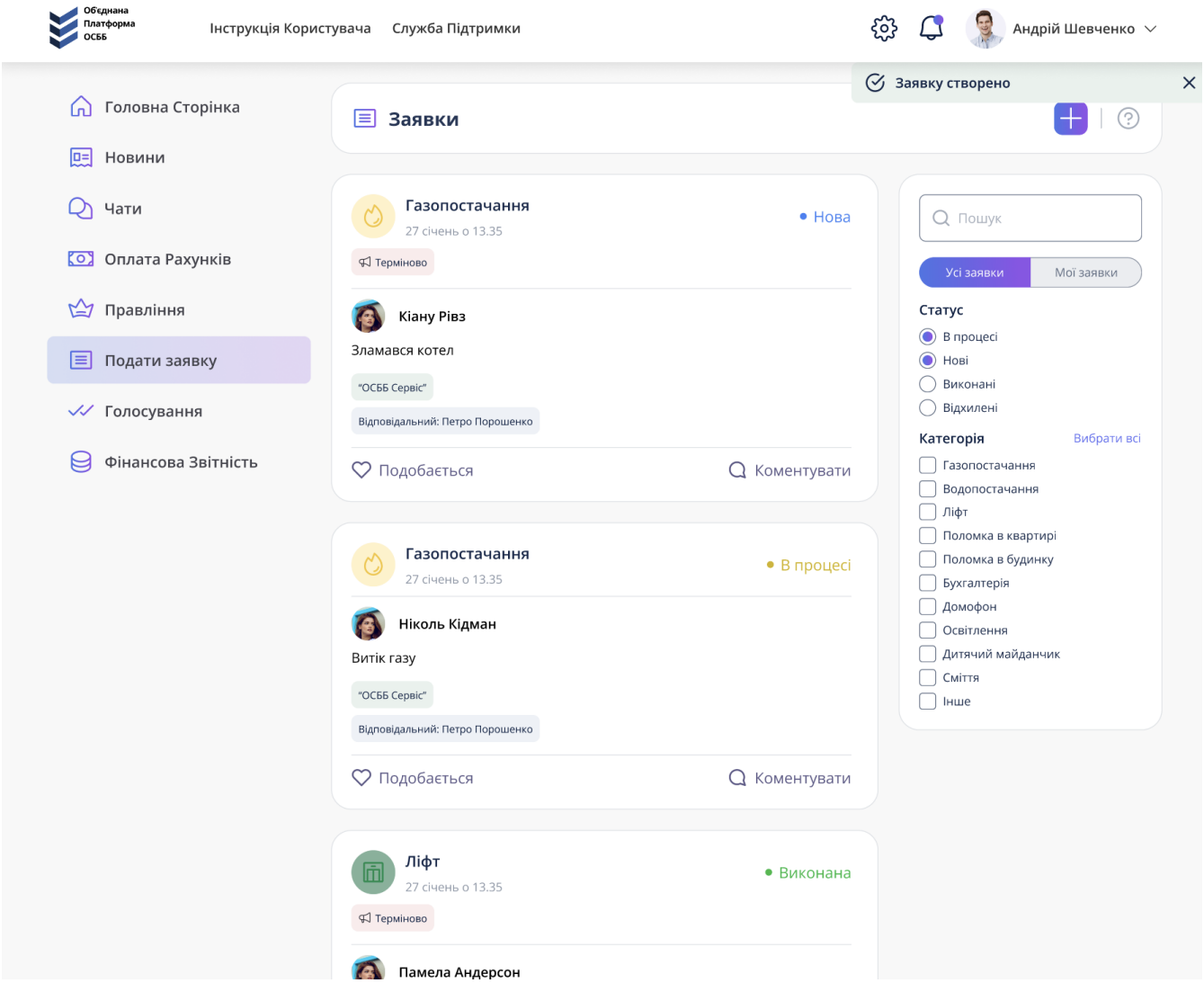


Рисунок 4.24 – Вигляд сторінки «Подати заявку» - 4 (рисунок виконаний самостійно)

Також є функціонал голосування з замими голосуваннями та інформацією з часом про закінчення голосувань. Такий функціонал є надзвичайно важливим, адже під час голосування мешканями приймаються рішення які напряму будеть впривати на майбутнє ОСББ а також проживання самих мешканців, цей функціонал ще не реалізований але плануться бути реалізованим за допомогою використання цифрового підпису адже кожне голосування в ОСББ це нормативний акт який керує майбутні процесу у ОСББ і до цього потрібно ставитися максимально відповідально. Функціонал голосування зображено на рисунку 4.25.

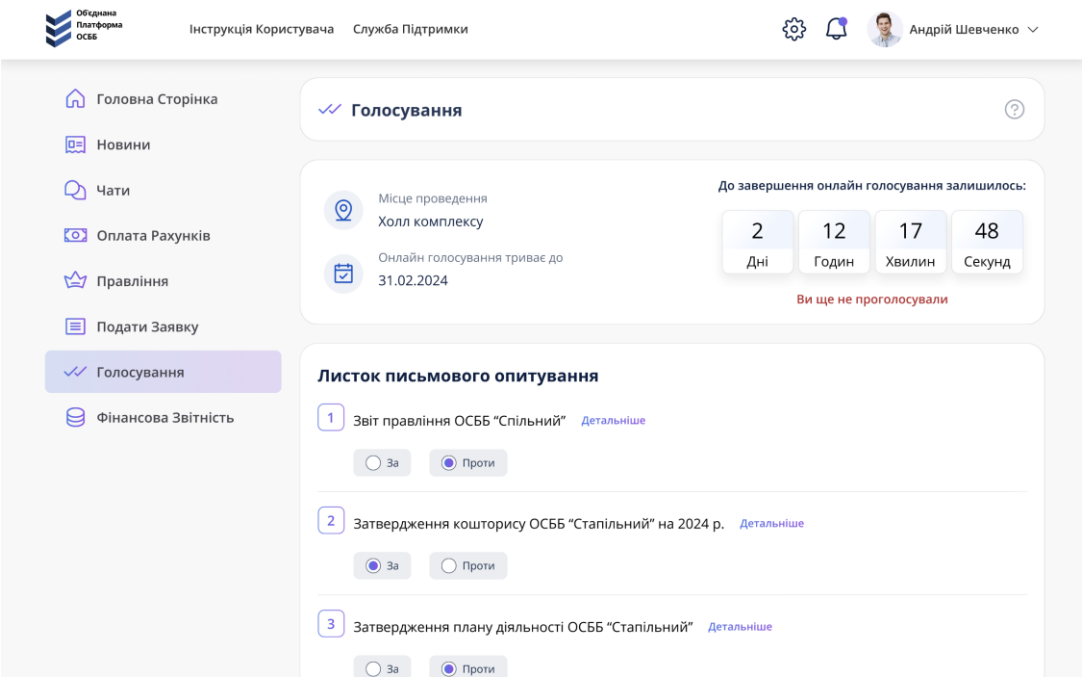


Рисунок 4.25 – Вигляд сторінки «Голосування» (рисунок виконаний самостійно)

Також важливим функціоналом є звітність ОСББ перед мешканцями будинку, такий функціонал відображається на вкладці «Фінансова звітність». На цій сторінці мешканець може отримувати статистику за обрані періоди а також обіг коштів та заборгованість (рисунки 4.26 – 4.27).

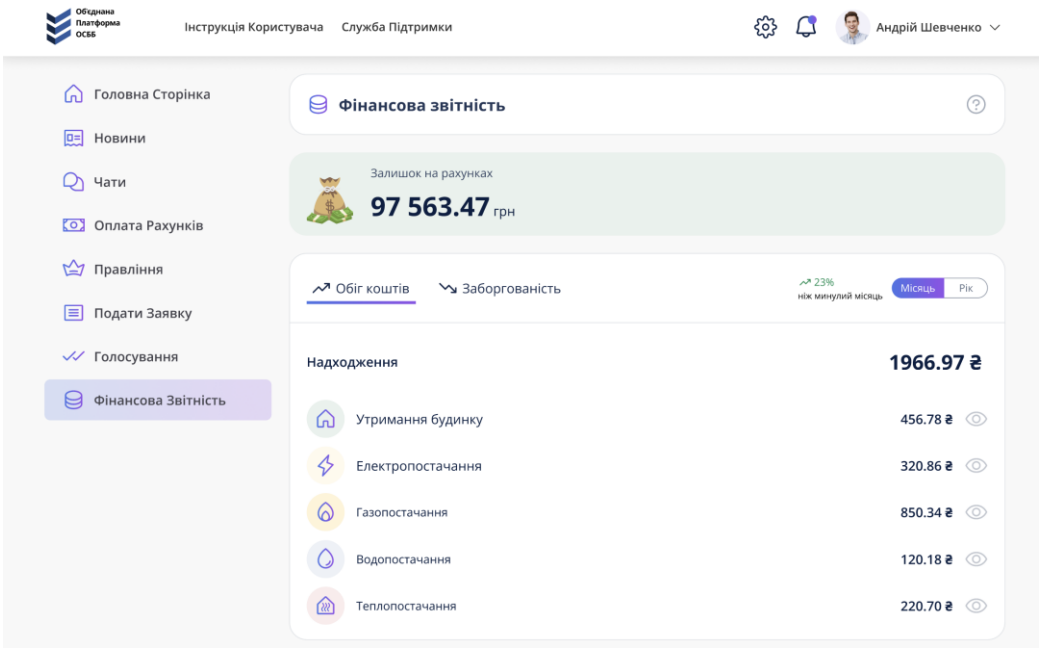


Рисунок 4.26 – Вигляд сторінки «Фінансова знітність» -1 (рисунок виконаний самостійно)

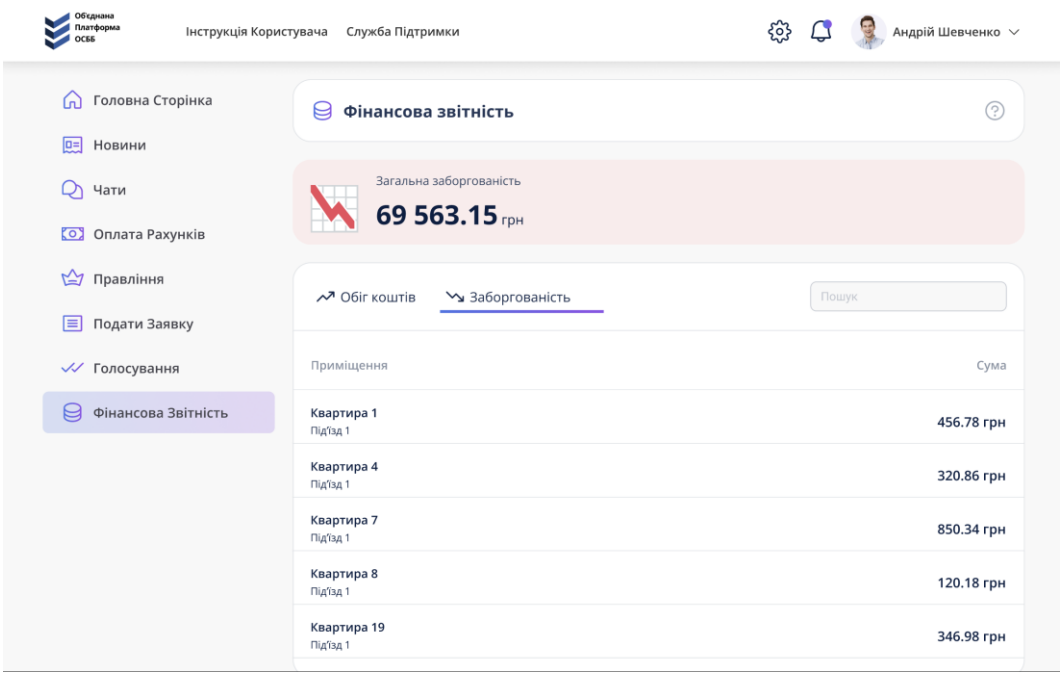


Рисунок 4.27– Вигляд сторінки «Фінансова знітність» -2 (рисунок виконаний самостійно)

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування функціоналу

З метою перевірки функціоналу програми було обрано метод ручного тестування. Ручне тестування передбачає перевірку програмного забезпечення на наявність дефектів вручну, без застосування засобів автоматизації та скриптів. Під час тестування тестувальник взаємодіє з системою як користувач, вводячи дані та перевіряючи результати з метою виявлення помилок, дефектів та невідповідностей вимогам, які були висунуті до програми [11].

Цей метод було обрано, оскільки він найбільш підходить для тестування продукту на ранніх етапах розробки, наприклад, у стадії MVP. У такій системі часто відбуваються зміни та вдосконалення, що вимагає частого оновлення тестів і значних витрат часу. Це особливо важливо при впровадженні нового функціоналу. Ручне тестування дозволяє швидко перевірити систему при будь-яких значних або незначних змінах у програмі.

Крім того, ручне тестування дозволяє використовувати досвід та інтуїцію тестувальника для виявлення проблем, які важко виявити автоматизованими засобами тестування. Тестувальник може мислити нестандартно, знаходити непередбачувані проблеми та дефекти, тестувати нетипові сценарії використання додатку.

Ще одним фактором, що вплинув на вибір ручного тестування, стали труднощі у використанні бібліотек для тестування React:

- додаток часто використовує асинхронні операції, які викликають труднощі при тестуванні (наприклад, запити до серверу, виклики API, обробка даних під час завантаження);
- якщо компонент використовує сторонні бібліотеки та має залежності від них, можуть виникати проблеми, що вимагають пошуку обхідних шляхів або використання мокування для підміни залежностей, за допомогою таких інструментів як jest.mock();

– складнощі тестування взаємозв'язків та передачі даних між деякими сторінками, наприклад, при передачі даних через стан при переході користувача на іншу сторінку.

Для проведення тестування був складений та використаний чек-ліст, який представляє собою структурований список усіх компонентів, які необхідно перевірити. На рисунку 5.1 наведено фрагмент чек-листу для перевірки сторінки реєстрації.

Сторінка підписки та реєстрації		
	Вибір тарифного плану	
	Шкала етапів (4 етапи)	
	Поле вводу даних	
		Поле "Ім’я"
		Поле "Номер телефону"
		Поле "Адреса"
		Поле "Електронна адреса"
		Кнопка «Продовжити»
	Екран "Дані про ваш будинок"	
		Дропдаун меню "Кількість під’їздів"
		Дропдаун меню "Кількість поверхів"
		Дропдаун меню "Кількість квартир на поверсі"
		Кнопка "Пропустити"
		Кнопка "Продовжити"
	Екран "Спосіб оплати"	
		Кнопка вибору способу оплати
		Додаткові поля методу оплати у разі вибору одного з: “GPay”, “ApplePay”, “MonoPay”, “Privat 24”, “Visa”, “MasterCard”
		«Зберегти дані» чекбокс
		Кнопка "Продовжити"
	Екран «Сповістіть мешканців»	
		Згенерований QR код з лінкою на інструкцію користування для мешканців
		Кнопка «Файл та Інструкція» яка завантажує згенерований документ по натисканню
		Кнопка «Завершити»

Рисунок 5.1 – Чекліст сторінки реєстрації (рисонок виконаний самостійно)

Використання чек-листів дозволяє скоротити час тестування, усуваючи необхідність підготовки тест-кейсів та тест-планів. Повний чек-ліст наведено у додатку Б.

Застосунок було протестовано у найбільш поширених браузерах, таких як Google Chrome, Safari, Mozilla Firefox, Microsoft Edge та Opera.

Під час тестування не було виявлено жодних блокуючих, критичних та значних помилок.

5.2 Тестування інтерфейсу

Щоб ретельно перевірити інтерфейс додатку, було вирішено використовувати ручне тестування. Цей підхід дозволяє вручну оцінити візуальні аспекти та загальне сприйняття інтерфейсу, такі як розташування елементів, кольори, шрифти, анімації та інші дизайнерські елементи. Хоча бібліотеки для тестування React можуть перевіряти наявність конкретних елементів на сторінці, вони не здатні повністю оцінити взаємодію користувача з інтерфейсом та його візуальне враження від нього. Також автоматизовані тести не можуть виявити ситуації, коли елемент присутній у структурі сторінки, але відображається некоректно (наприклад, перекритий іншими елементами або знаходиться поза межами видимого блоку).

Ручне тестування дозволяє оцінити реальний досвід користувача та інтуїтивність інтерфейсу у реальних умовах. Автоматизовані тести не здатні адекватно оцінити ці аспекти, оскільки вони зосереджені лише на функціональності, а не на користувацькому досвіді.

Для перевірки інтерфейсу використовувалися структуровані чек-листи, які охоплювали різні аспекти перевірки елементів інтерфейсу:

- відповідність іконок їх значенню;
- уніфікація дизайну;
- ясність повідомлень;

- відповідність виду елемента його призначенню;
- естетичність оформлення;
- вирівнювання та розташування елементів;
- зручність навігації;
- адаптивність сторінки;
- шрифти;
- довгі назви;
- анімації;
- зміна курсору при наведенні на елемент введення;
- відповідність стандартам;
- ефекти при наведенні на елемент введення;
- правопис;
- скролінг при переповненні;
- повідомлення про підтвердження дії;
- підказки.

Ручне тестування забезпечує комплексний підхід до оцінки інтерфейсу, дозволяючи виявляти та усувати проблеми, які можуть залишитися непоміченими під час автоматизованого тестування.

В результаті тестування інтерфейсу серйозних помилок виявлено не було.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено клієнтську частину програмної системи для роботи ОСББ в Україні.

В ході роботи було проведено аналіз предметної області, виявлено проблеми з якими стикаються ОСББ, проведено порівняльний аналіз конкуруючих продуктів, і на основі проведеної попередньої роботи була поставлена задача на розробку програмної системи для управління ОСББ, та сформульовано функціональні та нефункціональні вимоги до системи.

Перед початком розробки програмної системи було проведено моделювання та створено різні види UML діаграм, для кращого уявлення про майбутню архітектуру системи, та те, як вона буде функціонувати. Також було спроектовано користувацький інтерфейс застосунку з урахуванням загальноприйнятих стандартів розробки UI.

Для розробки програми була використана мова програмування JavaScript з розширенням JSX, бібліотека React.js, платформа Node.js, бібліотека i18next, бібліотека React Leaflet.

Робота над проектом дозволила не лише застосувати набуті раніше теоретичні знання на практиці, а й додатково їх поглибити та систематизувати, а також дозволила отримати цінні практичні навички у розробці веб-застосунків та використанні різних програмних інструментів. Отримано досвід з формування вимог до системи, проектування архітектури, розробки користувацького інтерфейсу.

Результатом кваліфікаційної роботи стала клієнтська частина програмної системи для підтримки взаємодії керівництвом ОСББ та мешканцями цього об'єднання, яка відповідає поставленим задачам, та виконує усі висунуті вимоги. Програма дозволяє мешканцям багатоквартирного будинку контролювати роботу ОСББ, мати доступи до різноманітних звітностей, голосувать та сплати рахунків. Також частиною додатку є можливість спілкуватися мешканцям між собою а керівництву ОСББ тримати руку на життевому поульсі будинку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт MySQL. URL: https://dev.mysql.com/doc/ (дата звернення 20.05.2024)

2. Figma – онлайн сервіс для розробки інтерфейсів та прототипування. URL: https://www.figma.com /(дата звернення 29.05.2024)

3. Python – високорівнева мова програмування. URL: https://www.python.org/ (дата звернення 27.05.2024)

4. “Мій Дім” – онлайн сервіс для ОСББ. URL: https://miydimonline.com.ua/ (дата звернення 22.05.2024)

5. “Дах” - онлайн сервіс для ОСББ. URL: https://dah-online.com/ (дата звернення 30.05.2024)

6. “ОМО” - онлайн сервіс для ОСББ. URL: https://omo.systems/ua/ (дата звернення 30.05.2024)

7. «Моє ОСББ» - онлайн сервіс для ОСББ. URL: https://moeosbb.com/(дата звернення 30.05.2024)

8. PyCharm – офіційний сайт середовища розробки. URL: https://www.jetbrains.com/pycharm/ (дата звернення 20.06.2024)

9. Referencing values with refs. React. URL: https://react.dev/learn/referencingvalues-with-refs (дата звернення: 18.07.2024).

10. Map creation and interactions. React. URL: https://reactrouter.com/en/main/route/route (дата звернення 20.05.2024)

11. Ручне та автоматизоване тестування. URL: https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/ (дата звернення: 18.07.2024).

ДОДАТОК А

Чек-ліст для тестування клієнтскьої частини

Таблиця А.1 Чек-ліст для тестування (таблиця виконана самостійно)

Сторінка підписки та реєстрації		
	Вибір тарифного плану	
	Шкала етапів (4 етапи)	
	Поле вводу даних	
		Поле "Ім’я"
		Поле "Номер телефону"
		Поле "Адреса"
		Поле "Електронна адреса"
		Кнопка «Продовжити»
	Екран "Дані про ваш будинок"	
		Дропдаун меню "Кількість під’їздів"
		Дропдаун меню "Кількість поверхів"
		Дропдаун меню "Кількість квартир на поверсі"
		Кнопка "Пропустити"
		Кнопка "Продовжити"
	Екран "Спосіб оплати"	
		Кнопка вибору способу оплати
		Додаткові поля методу оплати у разі вибору одного з: “GPay”, “ApplePay”, “MonoPay”, “Privat 24”, “Visa”, “MasterCard”

Продовження таблиці А.1

		«Зберегти дані» чекбокс
		Кнопка "Продовжити"
	Екран «Сповістіть мешканців»	
		Згенерований QR код з лінкою на інструкцію користування для мешканців
		Кнопка «Файл та Інструкція» яка завантажує згенерований документ по натисканню
		Кнопка «Завершити»
Головна сторінка		
	Хедер	
		Логотип Платформф ОСББ
		Кнопка «Інструкція користувача»
		Кнопка «Служба підтримки»
		Кнопка «Налаштування»
		Дзвіночок індикуючий про повідомлення
		Кнопка з іменем користувача
	Ліва частина сторінки з табами	
		Кнопка «Головна сторінка»
		Кнопка «Новини»
		Кнопка «Чати»
		Кнопка «Оплата рахунків»

Продовження таблиці А.1

		Кнопка «Правління»
		Кнопка «Подати заявку»
		Кнопка «Голосування»
		Кнопка «Фінансова звітність»
	Верхня частина головної сторінки	
		Інформація про борг за рік або місяць
		Інформація про витрати ОСББ за рік або місяць
		Інформація про надходження фінансів на рахунок ОСББ за рік або місяць
	Середня частина головної сторінки	
		Графік витрат користувача за рік або місяць
		Новини будвнку в хронологічному порядку
	Права частина головної стоорінки	
		Поле заявки зі статусом та короткою інформацією по заявці
		Поле актуального голосування з інформацією про голосування і можливістю проголосувати
Новини		
	Верхня частина	
		Назва сторінки «Новини»
		Кнопка для подачі новини
	Центральна частина	

Продовження таблиці А.1

		Опис новин з типом новини, датою публікації та автором публікації, з можливістю «полайкати» новину та можлфвістю прокоментувати ноавину
	Права частина	
		Пошук новини за ключовими словами або фільтрацією за датою
Сторінка «Чат»		
	Середня частина містить 2 вкладки – «Чат» та «Контакти»	
	Вкладка «Чат» центральна частина	
		Срисок груп та контактів та кількість повідомлень від користувачів якщо вам хтось написав
	Вкладка «Чат» права частина	
		Сама переписка з обраним контактом
	Вкладка контакти центральна частина	
		Список контактів, по натисненню на контакт відкривається повна інформація про нього з номером телефону, аватаркою, адресою та кнопкою «Написати повідомлення»
Оплата Рахунків		
	Верхня частина сторінки	
		Назва таблиці: «Ваші заборгованості»
	Вид заборгованості	

Продовження таблиці А.1

		«Утримання будинку» - баланс та кнопка «Сплатити окремо»
		«Електропостачання» - баланс та кнопка «Сплатити окремо»
		«Газопостачання» - баланс та кнопка «Сплатити окремо»
		«Водопостачання» - баланс та кнопка «Сплатити окремо»
		«Теплопостачання» - баланс та кнопка «Сплатити окремо»
		«Всього» інформація по загальній заборгованості і кнопка «Сплатити»
Правління		
	Верхня частина сторінки	
		Назва сторінки «Правління»
	Центральна частина сторінки	
		Плитка з іменами, посадою, годинами прийому, номеном телефону, імейлом та «Надіслати повідомлення» кнопкою (3 плитки в рядку)
Подати заявку вкладка		
	Верхня частина сторінки	
		Назва вкладки «Заявки» та кнопка «+» для подачі заявок
	Центральна частина сторінки	
		Назва заявки, статуст, дата публікації, автором публікації, коротким описом проблеми, відповідальною особою, моливістю полайкати та прокоментувати заявку
		Заявки відображаються у хронологічній послідовності

Продовження таблиці А.1

	Права частина сторінки	
		Пошук заявок по ключовим словам
		Фільтрація заявок: «Мої»/«Усі»
		Фільтрація по статусу заявки: Нові, В процесі, Виконані, Відхилені
		Вибір по категоріям: - газовостачання; - водопостачання; - ліфт; - поломка в квартирі; - поломка в будинку; - бухгалтерія; - домофон; - освітлення; - дитячий майданчик; - сміття; - інше.
Голосування		
	Верхня частина сторінки	
		Назва сторінки «Голосування»
		Відображення інформації про голосувавання: Місце проведення голосування, дата до якої буде тривати онлайн голосування, таймером зворотнього відліку до кінця голосувавання та статусом вашого голосувавання (проголосовано чи ще ні)
	Центральна частина сторінки	
		Листок зі списком письмового опитування та з інформацією про голосування і кнопкою «детальніше»
Фінансова звітність		

Кінець таблиці А.1

	Верхня частина сторінки	
		Назва сторінки «Фінансова звітність»
		Залишок на рахунку ОСББ
	Центральна частина сторінки	
	Обіг коштів вкладка	
		Надходження коштів на утримання будинку з сумою
		Надходження коштів на електропостачання будинку з сумою
		Надходження коштів на газопостачання будинку з сумою
		Надходження коштів на водопостачання будинку з сумою
		Надходження коштів на теплопостачання будинку з сумою
		Кнопка з можливістю обрати періоз за рік та місяць
	Заборгованість вкладка	
		Інформація по боржникам з номером квартирв та сумою борку
		Надходження коштів на електропостачання будинку з сумоюожливістю пошуку заборгованості по квартирі

Додаток Б
Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

StrikePlagiarism.com

Дата звіту 7/22/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп-22-2_Грабар_О_М

Автор
Грабар Олександр

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	16

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

6.71% 6.71% КП 1

3.19% 3.19% КЦ

25
Довжина фрази для коефіцієнта подібності 2

6151
Кількість слів

49376
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://www.osbb.online/	43	0.70 %
2	https://present5.com/disciplina-bazi-danix-praktichne-zanyattya/	28	0.46 %
3	https://miydimonline.com.ua/uk/companyregistration/create?from=index	23	0.37 %
4	https://present5.com/disciplina-bazi-danix-praktichne-zanyattya/	23	0.37 %
5	ПЗПІп-22-2_Махлай_К_О [redacted] 7/16/2024 Kharkiv National University of Radio Electronics (Харківський національний університет радіоелектроніки)	20	0.33 %

Додаток В

Слайди презентації

КВАЛІФІКАЦІЙНА РОБОТА

Програмне забезпечення
підтримки діяльності ОСББ.
Веб застосунок.

1

Виконав: ст. ПЗПІп-22-2 Грабар О.М.
Керівник: доц. Кириченко І.В.

2

МЕТА РОБОТИ

- Мета розробки – створити зручну та ефективну систему, яка зможе полегшити життя мешканцям ОСББ, керівництву ОСББ та допоможе акумулювати додаткові кошти на обслуговування об’єкту і тим самим зменшити квартплату для кожного мешканця.

3

ПРОБЛЕМИ ПРЕДМЕТНОЇ ГАЛУЗІ

- Після створення неприбуткової юридичної організації ОСББ стикається з проблемою бухгалтерського обліку
- Відсуктність коммунікації між мешканцями будинку між собою, а також між мешканцями будинку та правляінням ОСББ
- Відсутність платформи для голосування на установчих зборах онлайн
- Дуже часто відсутня інформація у мешканців про стан справ у ОСББ, а також легкого зв'язку з мешканців та правління
- Неможливість прозорого моніторингу обігових коштів ОСББ та боротьби з боржниками
- Багато мешканців ОСББ скаржаться на важкість комунікації з правлінням а також створення поточних заявок на ремонт чи покращення життя ОСББ
- Відсутність зручної оплати комунальних послуг у одному місці
- Одна з проблематик – це новини ОСББ де можно було б оперативно моніторити ситуацію у будинку
- Відсутність єдиної платформи яка б об'єднувала в собі загальний функціонал який потрібен мешканцям

4

АНАЛІЗ КОНКУРЕНТІВ

Аналог	Бухгалтерський облік	Звітність	Облік за лічильниками	Інтеграція з платіжними системами	Комунікація всередині платформи	Інтеграція з розумним будинком	Мобільний додаток
ОМО	-	+	+	+	-	+	+
ДАХ	+	+	-	+	+	-	+
Моє ОСББ	+	+	+	+	+	-	-
Мій дім	+	+	+	+	+	-	+
Моя оселя	-	+	-	-	+	-	-

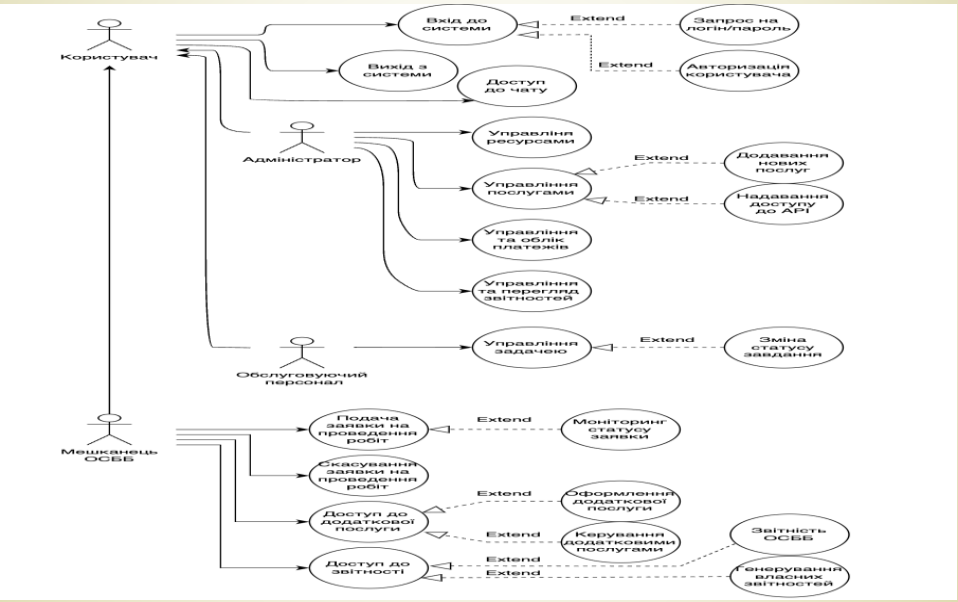
5

ПОСТАНОВКА ЗАДАЧІ

- Необхідно спроектувати та програмно реалізувати веб-застосунок для систем підтримки діяльності ОСББ, який буде забезпечувати наступні функції.
  - містити всю інформацію про ОСББ, його керівництво, всіх проживаючих у будинку в цілому, а також у квартирі, інформацію про прописаних мешканців квартири (це дозволить робити документи про склад родини наприклад), наявність автівки, марку, номер квартири і ім'я її власника.
  - Розсилка нагадувань про сплату квартплати, про заборгованість на початок місяця, про послуги якими користується та чи інша квартира, інформування про події у житті будинку
  - Створення звітів по витратах ОСББ.
  - Можливість оформлювати заявки на виконання сервісних чи ремонтних робіт для мешканців, наприклад: виклик сантехніка, виклик електрика, виклик майстра по вікнах та дверях, заявка інтернет провайдеру, виклик завгоспа, який зможе щось полагодити у будинку чи ліквідувати аварію.
- Провести тестування веб-застосунку

6

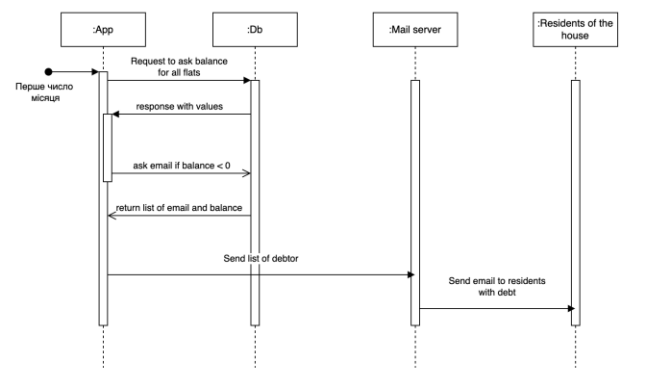
Use case діаграма



7

Діаграми послідовності.

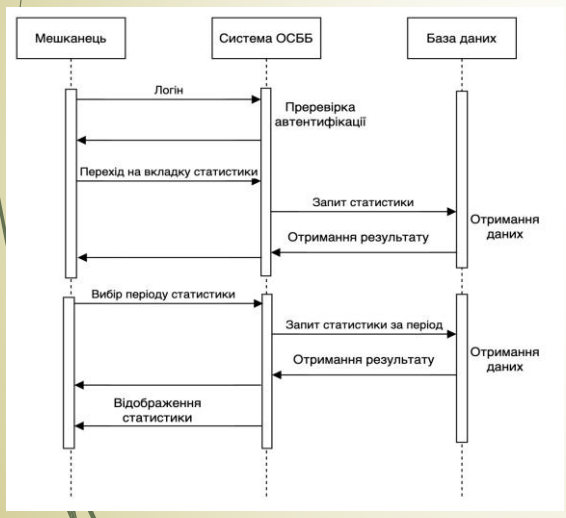
Щомісячна розсилка електронних листів на пошту боржників



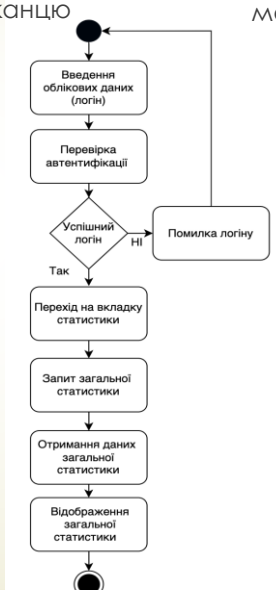
8

Отримання статистики по мешканцю

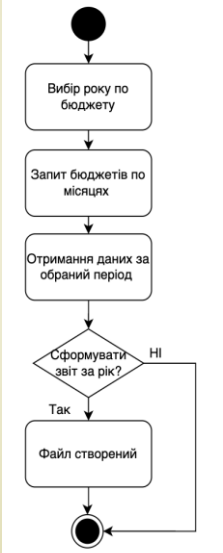
Діаграма послідовностей запиту до бази даних



Запиту загальної статистики по мешканцю



Запиту статистики за обраний період по мешканцю



9

ВИКОРИСТАНІ ТЕХНОЛОГІЇ

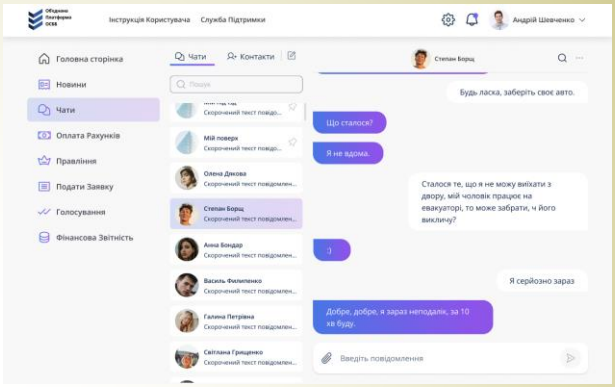
10

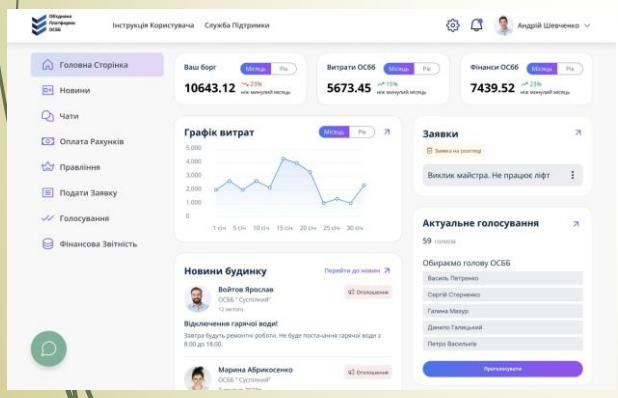
Фрагмент коду

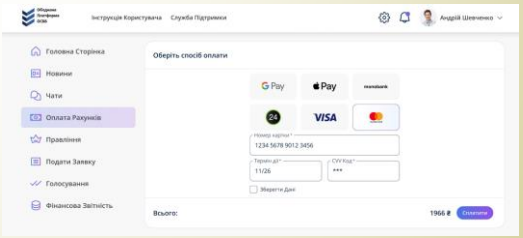
```
import "../global.css";
import "../styleguide.css";
import React from "react";
import ReactDOM from "react-dom/client";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import { LandingPage } from "./screens/LandingPage";
import { Dashboard } from "./screens/Dashboard";
import ProtectedRoute from './components/ProtectedRoute/ProtectedRoute.js';
import { Screen } from "./screens/Orders";

const router = createBrowserRouter([
  {
    path: "/",
    element: <LandingPage />,
  },
  {
    path: "/dashboard",
    element: (
      <ProtectedRoute>
        <Dashboard />
      </ProtectedRoute>
    ),
  },
  {
    path: "/orders",


import "../global.css";
import "../styleguide.css";

import React from "react";
import ReactDOM from "react-dom/client";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import { LandingPage } from "./screens/LandingPage";
import { Dashboard } from "./screens/Dashboard";
import ProtectedRoute from './components/ProtectedRoute/ProtectedRoute.js';
import { Screen } from "./screens/Orders";
```

11

Інтерфейс додатку

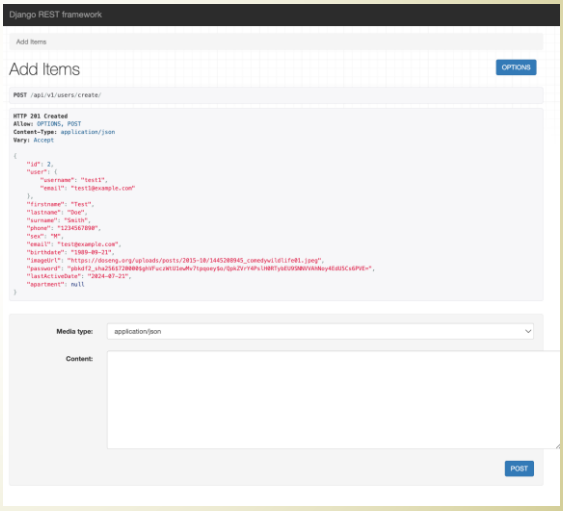






12

Адмін меню



13

ТЕСТУВАННЯ

- З метою перевірки функціоналу програми було обрано метод ручного тестування.

Сторінка підписки та реєстрації
Вибір тарифного плану
Шкала етапів (4 етапи)
Поле вводу даних
Поле "Ім'я"
Поле "Номер телефону"
Поле "Адреса"
Поле "Електронна адреса"
Кнопка «Продовжити»
Екран "Дані про ваш будинок"
Дропдаун меню "Кількість під'їздів"
Дропдаун меню "Кількість поверхів"
Дропдаун меню "Кількість квартир на поверсі"
Кнопка "Пропустити"
Кнопка "Продовжити"
Екран "Спосіб оплати"
Кнопка вибору способу оплати
Додаткові поля методу оплати у разі вибору одного з: "GPay", "ApplePay", "MonoPay", "Privat 24", "Visa", "MasterCard"
«Зберегти дані» чекбокс
Кнопка "Продовжити"
Екран «Сповістіть мешканців»
Згенерований QR код з лінкою на інструкцію користування для мешканців
Кнопка «Файл та Інструкція» яка завантажує згенерований документ по натисканню
Кнопка «Завершити»

Тестування Інтерфейсу

14

Для перевірки інтерфейсу використовувалися структуровані чек-листи, які охоплювали різні аспекти перевірки елементів інтерфейсу:

- відповідність іконок їх значенню;
- уніфікація дизайну;
- ясність повідомлень;
- відповідність виду елемента його призначенню;
- естетичність оформлення;
- вирівнювання та розташування елементів;
- зручність навігації;
- адаптивність сторінки;
- шрифти;
- довгі назви;
- анімації;
- зміна курсору при наведенні на елемент введення;
- відповідність стандартам;
- ефекти при наведенні на елемент введення;
- скролінг при переповненні;
- повідомлення про підтвердження дії;

ВИСНОВКИ

15

- Під час виконання кваліфікаційної роботи було розроблено клієнтську частину програмної системи для роботи ОСББ в Україні.
  - В ході роботи було проведено аналіз предметної області, виявлено проблеми з якими стикаються ОСББ, проведено порівняльний аналіз конкуруючих продуктів, і на основі проведеної попередньої роботи була поставлена задача на розробку програмної системи для управління ОСББ, та сформульовано функціональні та нефункціональні вимоги до системи.
  - Проведено моделювання та створено UML діаграми.
  - Спроектовано користувацький інтерфейс застосунку з урахуванням загальноприйнятих стандартів розробки UI.
  - Для розробки програми була використана мова програмування JavaScript з розширенням JSX, бібліотека React.js, платформа Node.js, бібліотека i18next, бібліотека React Leaflet.
- Результатом кваліфікаційної роботи стала клієнтська частина програмної системи для підтримки взаємодії керівництвом ОСББ та мешканцями цього об'єднання, яка відповідає поставленим задачам, та виконує усі висунуті вимоги. Програма дозволяє мешканцям багатоквартирного будинку контролювати роботу ОСББ, мати доступи до різноманітних звітностей, голосувать та сплати рахунків. Також частиною додатку є можливість спілкуватися мешканцям між собою а керівництву ОСББ тримати руку на життевому поульсі будинку.