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)

Програмна система моніторингу зарядних станцій електромобілів. Керування проєктом, формування вимог, мануальний тестувальник та тестування навантаженням
(тема)

Виконала:
студентка 4 курсу, групи ПЗПІ_20-10

Налєскіна Т.С.
(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. каф. ПІ Русакова Н.Є.
(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 Налєскіній Тетяні Сергіївні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система моніторингу зарядних станцій електромобілів. Керування проєктом, формування вимог, мануальний тестувальник та тестування навантаженням

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

2. Термін подання студентом роботи до екзаменаційної комісії 17.06.2024

3. Вихідні дані до роботи Розробити документацію, яка передбачена програмною системою та керуванням проектом. Провести мануальне тестування та тестовування навантаженням проекту.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог, архітектура проекту та проектування, опис прийнятих рішень, тестування програмного забезпечення, впровадження програмного забезпечення висновки, додатки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	22.04.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	28.05.2024	*виконано*
5	Тестування ПЗ	30.05.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	12.06.2024	*виконано*
9	Нормоконтроль, рецензування	12.06.2024	*виконано*
10	Здача роботи у електронний архів	14.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	16.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Налєскіна Т.С.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Русакова Н.Є.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 131 сторінка, 24 рисунка, 6 додатків, 20 джерел.

ВЕБ-ДОДАТОК, КЕРУВАННЯ ПРОЕКТОМ, МАНУАЛЬНЕ ТЕСТУВАННЯ, МОНІТОРИНГОВА СИСТЕМА, ТЕСТУВАННЯ НАВАНТАЖЕННЯМ, JIRA, SCRUM.

Об’єктом роботи є керування проектом, формування вимог, тест кейси, діаграми та документація до програмної системи моніторингу зарядних станцій.

Метою даної роботи є розробка програмної системи моніторингу зарядних станцій електромобілів з акцентом на ефективному менеджменті команди. Основні завдання включають керування проектом з визначенням ролей та відповідальностей, формування вимог з урахуванням потреб користувачів, розробку мануальних тестів для валідації функціональності та тестування навантаженням для перевірки масштабованості системи. Підкреслюється важливість ефективної співпраці та комунікації в команді для досягнення успішних результатів у розробці та впровадженні програмного продукту.

Середовища роботи - JIRA, як інструментальна програма для керування проектами та здійснення стеження за завданнями в реальному часі. Вона дозволяє створювати завдання, призначати їх учасникам, встановлювати терміни виконання, вести звітність і взаємодіяти з командою через зручний інтерфейс.

Використана ітеративна методологія розробки програмного забезпечення, яка ґрунтується на спільній роботі крос-функціональних команд я введення проекту - SCRUM .

JIRA, LOAD TESTING, MANUAL TESTING, MONITORING SYSTEM, PROJECT MANAGEMENT, SCRUM, WEB APPLICATION.

The object of work is project management, requirements formation, test cases, diagrams and documentation for the software monitoring system of charging stations.

The purpose of this work is to develop a software monitoring system for electric vehicle charging stations with an emphasis on effective team management. Key tasks include project management with roles and responsibilities defined, requirements tailored to user needs, development of manual tests for functionality validation, and load testing to verify system scalability. The importance of effective collaboration and communication in a team to achieve successful results in the development and implementation of a software product is emphasized.

Environment is JIRA, as a tool for managing projects and tracking tasks in real time. It allows you to create tasks, assign them to participants, set deadlines, keep records and interact with the team through a convenient interface.

An iterative methodology of software development has been used, which is based on the joint work of cross-functional teams and the introduction of the project is SCRUM.

Я, Налєскіна Тетяна Сергіївна, студентка гр. ПЗПІ-20-10, здобувачка вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему « Програмна система моніторингу зарядних станцій електромобілів. Керування проєктом, формування вимог, мануальний тестувальник та тестування навантаженням», 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 ознайомлена 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 сучасному світі питання екології та сталого розвитку стають все більш актуальними, наприклад, використання альтернативних джерел енергії, зокрема електромобілів, що надає зросту їх популярності, що в свою чергу, ставить перед собою нові виклики, зокрема, необхідність розвитку інфраструктури зарядних станцій.

Програмна система моніторингу зарядних станцій електромобілів є ключовим елементом цієї інфраструктури, оскільки вона забезпечує контроль, управління та оптимізацію процесу зарядки, що є критичним для забезпечення зручності та ефективності використання електромобілів.

У даній роботі досліджується процес проектування та розробки програмної системи моніторингу зарядних станцій електромобілів. Робота розглядає вимоги до такої системи, аналізує існуючі рішення, визначає принципи та архітектуру розроблюваної системи, а також проводиться розробка та тестування програмної системи.

Виходячи з актуальності проблеми та необхідності подальшого розвитку інфраструктури зарядних станцій для підтримки електромобілів, дана робота має важливе значення для подальшого розвитку та впровадження екологічно чистих видів транспорту.

Мета роботи полягає в керуванні проектом розробки програмної системи для моніторингу зарядних станцій електромобілів. Основні завдання включають планування, організацію, керування та контроль над процесом створення програмного продукту, а також проведення мануального тестування для перевірки його працездатності та відповідності вимогам, а також тестування навантаженням для оцінки його витривалості та ефективності під великими навантаженнями.

Результатом цієї роботи буде успішна розробка та впровадження програмної системи моніторингу зарядних станцій електромобілів, яка відповідає вимогам замовника та забезпечує ефективне управління процесом зарядки електромобілів. Крім того, результатами є створення документації з вимог до програмного продукту та опису його функціональності, успішна розробка програмного коду з

відповідними тестами для підтвердження його працездатності, проведення тестування навантаженням для визначення стійкості та продуктивності системи, впровадження програмного продукту в реальному середовищі та надання підтримки після впровадження, а також задоволення потреб замовника та підтримка розвитку інфраструктури зарядних станцій для електромобілів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Зарядні станції електромобілів — це спеціальні пристрої, які призначені для зарядки акумуляторів електромобілів. Вони забезпечують можливість підключення електромобіля до електричної мережі для зарядки його батареї. Зарядні станції можуть бути встановлені як у домашніх умовах, так і на громадських майданчиках, парковках, заправних станціях тощо. Вони можуть мати різні типи роз'ємів та стандартів зарядки, такі як CHAdeMO, CCS, Type 2 (Mennekes), а також можуть підтримувати різні потужності зарядки. Зарядні станції є ключовою складовою інфраструктури для розвитку та популяризації електромобілів.

На момент 2022 року в Україні було близько 1500 зарядних станцій, які були розподілені по всій країні. Більшість зарядних станцій знаходилися у великих містах, таких як Київ, Львів, Харків, Одеса та Дніпро, але є і станції у менших містах та на автодорогах між містами [1].

На сьогоднішній день, в Україні відсутні великі депо електричних зарядних станцій. Незважаючи на те, що інфраструктура зарядних станцій для електромобілів постійно розвивається, встановлення депо для зберігання та обслуговування зарядних станцій у великих масштабах ще не стало загальнопоширеною практикою.

Незважаючи на відсутність електричних зарядних депо в Україні на даний момент, з розвитком та зростанням популярності електромобілів можливе з'явлення таких структур у майбутньому, особливо у великих містах або на транспортних вузлах, де буде великий попит на електричний транспорт та відповідно інфраструктуру для його обслуговування.

Наприклад, у багатьох країнах світу, зокрема в розвинених країнах Європи, Північної Америки та Азії, спостерігається значний попит на великі електричні зарядні депо [2]. Це пояснюється швидким ростом ринку електромобілів та стрімким розвитком інфраструктури для їх зарядки. Великі депо електричних зарядних станцій дозволяють забезпечити ефективне обслуговування електромобілів у великих кількостях та забезпечити їх енергетичні потреби на

масштабному рівні. Такі депо можуть служити центрами обслуговування, місцями зберігання та ремонту зарядних станцій, а також виконувати функції моніторингу та управління мережею зарядних станцій.

Предметна область моніторингу зарядних станцій електромобілів включає в себе різноманітні аспекти, які можна розглядати з точки зору функціональних, технічних та організаційних питань.

Функціональні аспекти:

- моніторинг зарядних станцій. Розуміння стану та ефективності зарядних станцій, включаючи доступність, швидкість заряду, використання енергії і т. д.
- управління зарядкою. Можливість керувати процесом зарядки, планування заряду, встановлення пріоритетів та оптимізація розподілу ресурсів.
- відстеження споживання енергії. Оцінка ефективності та економічної доцільності використання зарядних станцій для електромобілів.

Технічні аспекти:

- системи збору даних. Використання датчиків, IoT-пристроїв та інших технологій для збору даних про зарядні станції та їх функціонування.
- аналіз та обробка даних. Використання аналітичних інструментів для обробки та аналізу великого обсягу даних про зарядні станції для отримання цінної інформації.

Організаційні аспекти:

- управління проектом. Планування, виконання та контроль проекту з розробки та впровадження програмної системи моніторингу.
- вимоги до системи. Визначення функціональних та нефункціональних вимог до програмної системи з урахуванням потреб користувачів та стандартів безпеки.
- тестування та якість. Розробка та виконання тестів для перевірки функціональності, надійності та продуктивності програмної системи моніторингу.

Ці аспекти взаємодіють між собою, утворюючи комплексну систему для ефективного моніторингу та управління зарядними станціями для електромобілів.

1.2 Виявлення проблем та актуалізація рішень

Аналізуючи предметну галузь, можемо виявити проблеми, які стосуються умов введення бізнесу в галузі програмної системи моніторингу зарядних станцій електромобілів [3]:

- фінансові витрати: розробка та впровадження програмної системи може потребувати значних фінансових витрат на дослідження, розробку програмного забезпечення, інтеграцію та тестування;
- конкуренція: у галузі електромобілів та інфраструктури зарядних станцій конкуренція може бути досить інтенсивною. Важливо мати конкурентоспроможні переваги, такі як ефективна та надійна програмна система, щоб зберігати та залучати клієнтів;
- зміни в законодавстві: законодавство щодо електромобілів та інфраструктури зарядних станцій може змінюватися. Бізнес повинен бути готовим до змін у регулюванні та адаптуватися до нових вимог;
- технологічні ризики: швидкий розвиток технологій може призвести до виникнення технологічних ризиків, таких як застарілість програмного забезпечення або нездатність до інтеграції з новими пристроями та стандартами;
- проблеми з безпекою та конфіденційністю даних: збір та обробка особистої інформації про користувачів, такої як інформація про їх зарядку електромобілів, може викликати проблеми з безпекою та конфіденційністю даних. Важливо забезпечити відповідні заходи захисту та дотримуватися вимог законодавства щодо захисту персональних даних.

Головні проблеми з програмною системою, які можна виявити:

- сумісність з різними типами зарядних станцій: різні виробники можуть використовувати різні протоколи зв'язку та різні формати даних для звітів про зарядку. Програмна система повинна бути здатна взаємодіяти з

різними типами зарядних станцій та інтегруватися з різноманітними пристроями;

- безпека даних: оскільки система буде збирати та обробляти дані про зарядку електромобілів, важливо забезпечити високий рівень безпеки для захисту цих даних від несанкціонованого доступу, втрати або зміни;
- надійність і стійкість системи: наявність програмної системи, яка контролює зарядку електромобілів, має бути надійною та стійкою до відмов та перерв у роботі. Незаплановані відмови можуть призвести до незручностей для користувачів та порушення нормального функціонування зарядних станцій;
- масштабованість: з ростом кількості електромобілів збільшується навантаження на програмну систему. Система повинна бути здатна масштабуватися для обробки зростаючої кількості даних та запитів від користувачів;
- взаємодія з іншими системами: система може взаємодіяти з іншими системами, такими як системи платежів, моніторингу трафіку електромережі тощо. Важливо забезпечити сумісність та надійну взаємодію з цими системами.

Для вирішення проблем, що стосуються умов введення бізнесу у галузі програмної системи моніторингу зарядних станцій електромобілів можна використовувати ряд системних рішень, такі як:

- ефективне фінансове планування: ретельне аналізування та планування фінансових витрат на розробку, впровадження та підтримку програмної системи. Розробка бюджету, що передбачає можливі ризики та непередбачені витрати;
- стратегічне управління конкуренцією: аналіз ринку та конкурентів для розробки стратегії, яка базується на унікальних перевагах вашої компанії. Це може включати розвиток нових функцій, покращення якості обслуговування або створення інноваційних пропозицій для клієнтів;

- активний моніторинг законодавства: постійне відслідковування та оновлення змін у законодавстві, що регулює галузь електромобілів та інфраструктури зарядних станцій. Забезпечення відповідності всіх бізнес-процесів вимогам законодавства;
- інвестування в дослідження та розвиток: активне інвестування в дослідження та розвиток для постійного вдосконалення програмної системи та її адаптації до нових технологій та вимог ринку;
- захист даних та безпека інформації: розробка та впровадження ефективних стратегій та заходів захисту даних, включаючи шифрування, механізми аутентифікації та авторизації, а також регулярні аудити та оновлення систем безпеки.

Для вирішення головних проблем можна використовувати ряд системних рішень, такі як:

- стандартизація протоколів та форматів даних: створення стандартів зв'язку та обміну даними між різними типами зарядних станцій може спростити процес інтеграції та забезпечити сумісність з різноманітними пристроями;
- використання шифрування та інших методів безпеки: впровадження сучасних методів шифрування та захисту даних може забезпечити безпеку особистої інформації про користувачів та запобігти несанкціонованому доступу до неї;
- розробка резервних систем і механізмів відновлення: реалізація систем аварійного відновлення та резервного збереження даних може забезпечити надійність та стійкість роботи програмної системи навіть у випадку відмов чи перерв у роботі;
- використання розподіленої архітектури: використання розподіленої архітектури дозволить розподілити навантаження на різні сервери та ресурси, що дозволить збільшити масштабованість системи;
- стандартизація та інтеграція з іншими системами: розробка та впровадження стандартів взаємодії та інтеграції з іншими системами

дозволить забезпечити сумісність та ефективну взаємодію з системами платежів, моніторингу тощо.

Конкуренція становить серйозну проблему для продукту з кількох причин [4].

По-перше, наявність конкурентів у галузі може призвести до зменшення ринкової частки продукту. Коли існують альтернативні рішення, користувачі мають вибір, і якщо ваш продукт не виявиться конкурентоспроможним, ви можете втратити своїх клієнтів.

По-друге, конкуренція може призвести до зниження цін на продукт. Якщо конкуренти пропонують аналогічні або подібні рішення за меншу ціну, вам може доводитися знижувати власні ціни, щоб привернути або зберегти клієнтів.

Третій аспект полягає в тому, що наявність конкурентів може спонукати їх до постійного вдосконалення та розвитку своїх продуктів. Це ставить перед вами завдання бути на кривій інновацій та постійно розвивати свій продукт, щоб відповідати вимогам ринку та зберегати свою конкурентоспроможність.

Нарешті, конкуренція може призвести до втрати унікальності вашого продукту. Якщо інші компанії випускають аналогічні або кращі рішення, це може підірвати унікальність вашого продукту та призвести до втрати цільової аудиторії.

Таким чином, конкуренція є головною проблемою для продукту, оскільки вона створює серйозний тиск на всі аспекти бізнесу, включаючи ринкову позицію, цінову стратегію, рівень інновацій та унікальність продукту.

Головними конкурентами для нашого продукту є Omega app (bp pulse) [5], JetApp [6], Ev.energy [7]. Перейдемо до детальнішого аналізу продуктів наших конкурентів та виявимо переваги та недоліки в порівнянні з нашим.

Серед переваг програми Omega app (bp pulse) варто відзначити широкий вибір позицій для придбання ПЗ компанії, доступність системи цілодобово та наявність мобільного додатку для Android та iOS. Крім того, вона успішно працює на ринку США, пропонує програми лояльності для бізнесу та має зручний, сучасний інтерфейс.

Однак, є деякі недоліки. Зокрема, програма отримала багато негативних відгуків від користувачів, що негативно впливає на її репутацію. Також вона має проблеми з функціоналом, такими як автоматичне планування часу заряджання та авторизація. Неоднозначне позиціонування на ринку Європи, яке також залежить від репутації, є ще однією проблемою. Часті проблеми з оплатою через застосунок або використання ApplePay для клієнтів банків з Великобританії та Німеччини також не дозволяють програмі працювати безперервно та ефективно.

Серед переваг програми JetApp можна виділити принцип віддаленого контролю для менеджерів та адміністраторів, наявність мобільного застосунку та зручний, сучасний інтерфейс.

Проте, варто зазначити деякі недоліки. Наприклад, програма має обмежений функціонал, до якого входить лише перезавантаження системи, оповіщення про помилки, реєстрація та авторизація, а також таблиці з персоналом. Також слід відзначити локальність ПЗ, оскільки воно доступне лише для користувачів з Швейцарії.

Програма Ev.energy має свої переваги та недоліки.

Між переваг можна відзначити можливість встановлення програмного забезпечення для домашнього використання, наявність мобільного застосунку та багатий функціонал продукту.

Проте, варто зазначити деякі недоліки. Наприклад, виникають проблеми з використанням мобільного застосунку користувачам Tesla. Також спостерігаються проблеми з використанням програмного забезпечення користувачам, у яких підключена альтернативна електроенергія. Крім того, існують проблеми зі службою підтримки, які також варто врахувати.

Аналізуючи програми Omega app (bp pulse), JatApp та Ev.energy, можна зробити висновок, що їхні переваги та недоліки мають свої особливості. Але ми маємо переваги над цими готовими програмними продуктами:

– розширений функціонал: забезпечення більш широкого спектру функцій, включаючи просте управління та ефективне використання електроенергії;

- гнучкість і сумісність: забезпечення можливості використання продукту з різними моделями автомобілів та системами електроживлення робить його більш привабливим для широкого кола користувачів;
- зручний інтерфейс: розробка інтуїтивно зрозумілого та простого у використанні інтерфейсу забезпечує зручність користування та позитивне враження від продукту;
- надійність та підтримка: забезпечення стабільної роботи системи та якісної служби підтримки для вирішення будь-яких проблем робить продукт як надійний та професійний.
- інтеграція з екосистемою: врахування можливості інтеграції продукту з іншими системами та сервісами, такими як програми моніторингу енергоспоживання та мобільні застосунки, робить його більш адаптивним та зручним у використанні.

Таким чином, у вирішенні проблеми конкуренції важливо зосередитися на створенні унікальності та цінності продукту для клієнтів. Також важливо будувати сильний бренд та ефективну маркетингову стратегію для привертання та утримання клієнтів. Активний моніторинг ринку та реагування на зміни можуть допомогти зберегти конкурентну перевагу.

1.3 Постановка задачі

Програмна система для моніторингу зарядних станцій електромобілів є комплексним інформаційним рішенням, яке об'єднує в собі різноманітні компоненти і функції для ефективного керування та контролю зарядними станціями. Основні складові програмної системи включають:

Моніторинг і збір даних. Система здатна отримувати дані з різних зарядних станцій, включаючи інформацію про стан заряду, використання енергії, доступність станцій та інші параметри.

Аналітика та звітність. Можливість аналізувати накопичені дані для отримання важливої інформації щодо ефективності зарядних станцій, споживання

енергії, роботи системи тощо. Генерація звітів та статистики для подальшого аналізу та прийняття управлінських рішень.

Управління зарядками. Функціонал для керування процесом зарядки, включаючи планування заряду, встановлення пріоритетів, управління через віддалений доступ тощо.

Система оповіщень та аварійного управління. Можливість автоматичного виявлення проблем та аварій, надсилання сповіщень та управління ситуаціями навіть у відсутності користувача.

Інтерфейс для користувачів. Зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє легко взаємодіяти з програмною системою, переглядати дані, встановлювати параметри та отримувати звіти.

Захист та безпека. Забезпечення захисту даних, використання шифрування, аутентифікації користувачів та інших заходів для забезпечення безпеки і конфіденційності інформації.

Ці складові дозволяють програмній системі моніторингу зарядних станцій електромобілів ефективно функціонувати, забезпечуючи якісний моніторинг, керування та аналіз усіх аспектів зарядних станцій та їх роботи.

Зважаючи на тему "Програмна система моніторингу зарядних станцій електромобілів. Керування проєктом, формування вимог, мануальний тестувальник та тестування навантаженням", можна сформулювати наступні постановки задач для кожного аспекту:

Керування проектом:

- забезпечити вчасну реалізацію проекту з розробки програмної системи моніторингу;
- створити графік робіт, визначити критичні шляхи та контрольні точки для відстеження прогресу;
- організувати комунікацію між учасниками проекту, забезпечити звітність та вирішення конфліктів.

Формування вимог:

– провести аналіз потреб користувачів та визначити функціональні та нефункціональні вимоги до програмної системи;
– створити документацію з вимогами, яка включатиме у себе опис функціональності, вимоги до інтерфейсу, умови експлуатації тощо;
– забезпечити взаємодію з користувачами для уточнення та затвердження вимог.

Мануальне тестування:

– розробити тестові скрипти та сценарії для проведення мануальних тестів функціональності програмної системи;
– виконати тестування на відповідність вимогам, виявити та зареєструвати дефекти та недоліки у роботі системи;
– провести регресійне тестування для перевірки виправлення виявлених помилок.

Тестування навантаженням:

– створити тестові сценарії для навантажувального тестування, які відображатимуть реальне використання системи;
– провести тестування для оцінки роботи системи при великому навантаженні, визначити межі її працездатності;
– здійснити оптимізацію та вдосконалення системи на основі результатів тестування навантаженням.

Ці постановки задач є ключовими для успішної реалізації програмної системи моніторингу зарядних станцій електромобілів та забезпечення високої якості та ефективності її функціонування.

2 ФОРМУВАННЯ ВИМОГ

Ідея проекту полягає в розробці комплексного рішення для моніторингу та керування зарядними станціями електромобілів.

Основна мета проекту - забезпечити зручність та ефективність процесу зарядки для користувачів електромобілів.

Мета цієї роботи: менеджмент проекту, мануальне тестування продукту та тестування навантаженням, детальніше:

- менеджмент проекту: основною метою є ефективне керування проектом розробки програмної системи для моніторингу зарядних станцій електромобілів. Це включає в себе планування, організацію, контроль та виконання проекту з урахуванням термінів, бюджету та ресурсів;
- мануальне тестування продукту: другою метою є проведення мануального тестування програмного продукту з метою перевірки його працездатності, відповідності вимогам та виявлення можливих дефектів або помилок;
- тестування навантаженням: третьою метою є тестування програмної системи навантаженням для визначення її продуктивності, стійкості та відповідності робочим навантаженням. Це дозволить оцінити реальні можливості системи та її здатність працювати під навантаженням в реальних умовах.

Для успішного менеджменту проекту було виділено конкретні критерії успіху:

- чіткі цілі та обумовлення завдань: ретельно визначення цілей проекту та розкриття завдання, які потрібно виконати для їх досягнення.
- ефективне планування: розробка докладного плану робіт, включаючи розподіл завдань, призначення термінів, а також визначення ресурсів та бюджету.
- ефективне керівництво командою: створення ефективного комунікаційного механізму, сприяння спільній спрямованості та мотивування учасників команди.

- управління ризиками: аналіз потенційних ризиків та розробка стратегій їх управління для забезпечення безперервного прогресу проекту.
- моніторинг та звітність: відстеження прогресу робіт, вчасно виявлення відхилення від плану та надавання звітності про стан проекту.
- фокус на якості: забезпечення виконання робіт у відповідності до вимог якості та стандартів, проводячи відповідні контрольні процедури.
- гнучкість та адаптивність: готовність до змін в процесі роботи та швидкої адаптації до нових умов або вимог.

Перший етап включає в собі чітко сформовані вимоги, що відповідають стандартам та вимогам потенційним користувачам продукту, до системи для подальшого контролю над командним проектом. Особлива увага на:

- цілі та потенційний результат проекту;
- вимоги до інтерфейсу;
- вимоги до продуктивності;
- вимоги до надійності даних;
- вимоги до функціональності.

Вимоги до продукту можуть бути зміненні під час роботи над проектом під впливом різних чинників.

Вимоги:

Вимоги до інтерфейсу:

а) адаптованість означає, що інтерфейс повинен бути:

1) сумісним з потребами та можливостями користувача;
2) забезпечувати простоту переходу від виконання однієї функції до іншої
3) забезпечувати користувача на високому рівні вказівками стосовно його можливих дій, а також генерувати належний зворотний зв’язок на його запити;
4) надавати користувачу можливість відчувати себе повноправним керівником ситуації при розв’язанні всіх типів задач, тобто, забезпечувати його всією необхідною інформацією; користувач повинен бути впевненим, що він сам розв’язує поставлену задачу;

5) забезпечувати користувача різними, взаємно доповнюючими формами представлення результатів в залежності від типу запиту або від характеру отриманого рішення;

б) достатність інтерфейса;

в) дружність інтерфейсу;

г) гнучкість інтерфейсу.

Програмний інтерфейс. Програмне забезпечення для перегляду веб-сторінок має використовуватися користувачем для доступу до веб-системи. Він повинен підтримувати Javascript, HTML5, CSS3.

Комунікаційний протокол. Використання протоколу прикладного рівня для зв'язку між зарядними станціями електротранспорту та центральною системою управління OCPP.

Обмеження пам’яті: Веб-сайт не може безпосередньо контролювати кількість доступної пам'яті на комп'ютері користувача. Прямого обмеження пам'яті комп'ютера для користування веб-сайтом немає.

Операції. Для розробки веб-додатку використовується архітектурний стиль REST, що базується на принципах HTTP-протоколу. Основні операції, які можна виконувати з використанням REST, включають наступні:

Функції продукту.

а) Централізоване управління:

1) можливість власникам та операторам централізовано керувати всіма зарядними станціями через веб-платформу або мобільні додатки.
2) забезпечення ефективного та зручного управління роботою всіх станцій, включаючи регулювання параметрів та віддалене відключення чи включення.

б) Реальний час та геолокація:

1) надання інформації про доступність та стан кожної зарядної станції в режимі реального часу з використанням технологій геолокації.

2) забезпечення користувачам точну та актуальну інформацію, допомагаючи знаходити доступні станції найближче до їхнього розташування.

в) Аналітика та звітність:

1) збір та аналіз статистики використання, ефективності та попиту на послуги кожної зарядної станції.
2) надання власникам та операторам деталізовану інформацію для стратегічного планування та оптимізації інфраструктури.

г) Система оплати:

1) інтеграція з системами оплати для зручності користувачів та надання фінансової прозорості власникам.

д) Авторизація та аутентифікація:

1) забезпечення ідентифікацію користувачів та гарантією, що лише авторизовані особи мають доступ до різних рівнів функціоналу системи, зменшуючи ризик несанкціонованого використання.

Керування проектом за допомогою методологією Scrum. Scrum - це гнучка методологія управління проектами, яка базується на ітеративному та інкрементальному підході до розробки програмного забезпечення. У Scrum проект розбивається на короткі ітерації, які називаються спринтами [8].

У методології Scrum передбачено ряд задач, які повинні виконуватися:

- планування спринту: команда разом обговорює та визначає обсяг робіт на наступний спринт.
- проведення регулярних стендапів: короткі зустрічі команди, під час яких кожен розповідає про свої досягнення, плани та проблеми.
- реалізація робіт згідно обумовленого плану спринту: кожен учасник команди працює над виконанням визначених завдань протягом спринту.
- проведення регулярних ретроспектив: аналіз досягнень та проблем на завершенні кожного спринту з метою пошуку шляхів покращення робочого процесу.

- оцінювання спроможностей команди: визначення швидкості команди для ефективного планування робіт.
- оцінка інкременту продукту: перевірка результатів роботи команди на кінець кожного спринту з метою підтвердження їх відповідності вимогам та очікуванням.

Для оптимізації деяких задач буде застосована система керування проектами JIRA, для зустрічей - Google Meet, для підтримки зв’язку Telegram спільнота.

Jira підходить для керування проектом з кількох причин:

- гнучкість;
- інтеграція з іншими інструментами;
- управління завданнями та задачами;
- планування та призначення ресурсів;
- звітність та аналітика;
- спільна робота команди;

Також другої метою моєї роботи є мануальне тестування. Мануальне тестування - це процес вручну перевірки програмного продукту з метою виявлення помилок, дефектів та невідповідностей до вимог. Під час мануального тестування тестувальник виконує різні дії, перевіряє різні властивості та функціональність програми, та реєструє виявлені проблеми для подальшого виправлення розробниками. Цей вид тестування використовується для оцінки якості продукту перед його випуском на ринок, а також для забезпечення правильності його роботи після внесення змін або вдосконалень.

В обов’язки мануального тестувальника входить:

- аналіз вимог;
- створення тест-кейсів;
- виконання тест-кейсів;
- виявлення та документування дефектів;
- взаємодія з розробниками;
- регулярне звітування.

Третьою метою роботи є тестування навантаженням. Тестування навантаженням - це процес випробування програмного продукту під навантаженням, щоб оцінити його продуктивність та стійкість за різних умов навантаження. Під час цього виду тестування програму піддають дії, які відтворюють реальні умови використання, збільшуючи обсяг даних, кількість користувачів або навантаження на сервер. Мета - визначити, як програма поводиться за межами нормальних умов, та виявити можливі проблеми з продуктивністю, масштабуванням або стійкістю.

В обов’язки тестувальника навантаженням входить:

- планування тестування;
- створення сценаріїв навантаження;
- виконання тестів;
- аналіз результатів;
- внесення звітів;
- взаємодія з розробниками.

Тестування - це процес перевірки програмного продукту з метою виявлення дефектів, помилок або невідповідностей вимогам. Це важлива складова розробки програмного забезпечення, що допомагає забезпечити якість, надійність та відповідність продукту специфікаціям. Важливою частиною тестування є аналіз результатів та звітність для підтримки прийняття рішень у процесі розробки.

3 АРХІТЕКТУРА ПРОЕКТУ ТА ПРОЕКТУВАННЯ

3.1 UML проектування ПЗ

Функції, які планується реалізувати у першому релізі, було розділено на групи користувачів для полегшення їх представлення. Відомо, що система буде взаємодіяти з п'ятьма типами користувачів: власники, адміністратори від компаній, моніторингова система, водії та працівники (див. рис. 3.1).

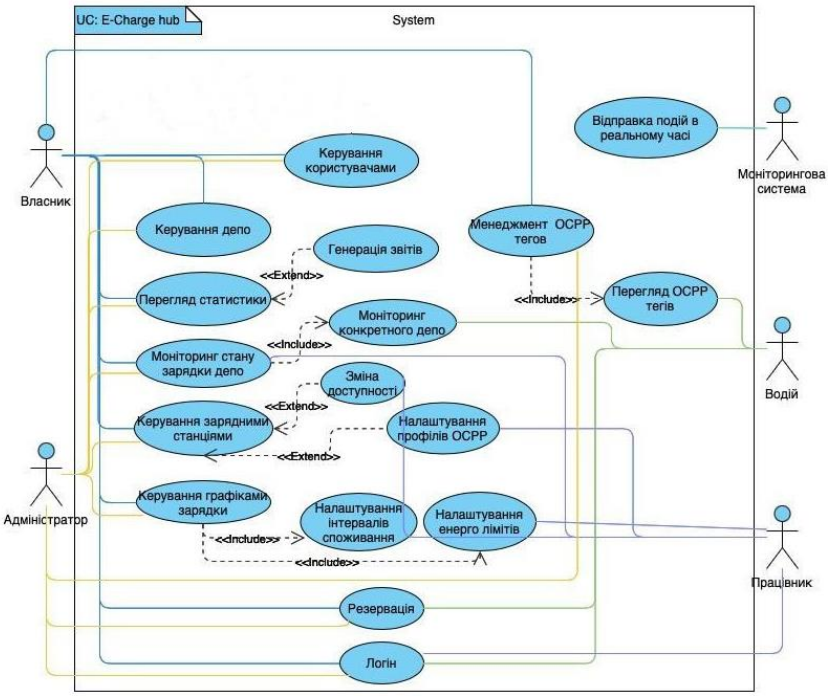


Рисунок 3.1 - Діаграма прецедентів програмної системи (рисунок виконано самостійно)

Власник (Owner) може займатись керуванням, користувачами та депо; переглядати статистику, яка включає в себе генерацію звітів Також може моніторити стан зарядки депо, що передбачає і моніторинг конкретного депо. Керувати зарядними станціями, що включають в собі зміну доступності та налаштування профілів OCPP. Керувати графіками зарядки, що передбачає налаштування інтервалів споживання та налаштування енергетичних лімітів. Має доступ до менеджменту OCPP тегів, що включає в собі перегляд OCPP тегів, та до резервації та авторизації.

Адміністратор (Administrator) може займатись керуванням користувачами та депо; переглядати статистику, що включає генерацію звітів. Також може моніторити стан зарядки депо, що передбачає і моніторинг конкретного депо. Керувати зарядними станціями, що включають в собі зміну доступності та налаштування профілів OCPP. Керувати графіками зарядки, що передбачає налаштування інтервалів споживання та налаштування енергетичних лімітів. Має доступ до менеджменту OCPP тегів, що включає в собі перегляд OCPP тегів, та до резервації та авторизації.

Моніторингова система (Monitoring system) відправляє події в реальному часі.

Водій (Driver) може моніторити конкретний роз'єм, переглядати OCPP теги. Має доступ до резервації та авторизації.

Працівник (Employee) може моніторити стан зарядки депо, займатись налаштуванням профілів OCPP та енергетичних лімітів. Має доступ до авторизації.

3.2 Обмеження та вимоги

Перед розробкою програмної системи потрібно правильно обрати технології розробки для кожної частини проекту, для того щоб не було технічних помилок та несумісностей, які вплинуть на саму розробку та подальше використання продукту.

Серверна частина буде реалізована за допомогою мікросервісної архітектури. Концепція мікросервісної архітектури полягає у дизайні архітектури програмної системи таким чином, що функціональність розподіляється між сервісами. Кожен із сервісів виконується незалежно і у власному потоці. Тож мікросервіси не повинні мати ніяких залежностей від платформ, інструментів або мов програмування і повинні бути розроблені за допомогою інструментів, які найкраще підходять для конкретної цілі. Найбільш імовірно, що кожен з сервісів використовується декількома різними клієнтами, серед яких інші мікросервіси системи, що розроблені і працюють на різних платформах. Тому важливо використовувати комунікаційні протоколи та засоби, що можуть ефективно використовуватися як мобільним додатком чи web застосунком, так і компонентами системи.

У мікросервісній архітектурі сервіси взаємодіють між собою за допомогою передачі повідомлень мережею. [9].

Розробка мікросервісів для системи буде проводитися на мові програмування C# з використанням платформи .NET 8 та фреймворку ASP.NET Core. Обрання цих конкретних технологій пояснюється їхньою здатністю підтримувати широкий спектр бібліотек, зокрема ті, які використовуються для роботи з хмарними технологіями, gRPC і RabbitMQ.

У зазначеній схемі взаємодії можна побачити, що зарядна станція взаємодіє через протокол OCPP та веб-сокети з відповідним сервісом обробки підключень, який називається "chargepoint websocket service". Цей сервіс відповідає за обмін повідомленнями між зарядними станціями та іншими сервісами системи.

Взаємодія між сервісом обробки підключень станцій та іншою частиною системи здійснюється асинхронно за допомогою брокера повідомлень RabbitMQ. Цей брокер дозволяє ефективно передавати та обробляти повідомлення між різними компонентами системи безпосередньо та надійно. Така архітектурна модель сприяє швидкому та надійному обміну даними між різними частинами системи та її сервісами.

Технологічний стек для розгортання програмної системи моніторингу зарядних станцій електромобілів використовує сучасні інструменти та підходи для ефективного управління та функціонування системи. У цьому контексті, використання Kubernetes та Docker є важливим елементом для забезпечення ізольованості, масштабованості та автоматизації процесів. Дані технології дозволяють ефективно керувати процесами розгортання, моніторингу та управління додатками в хмарних середовищах та власних дата-центрах.

У цьому підпункті детально розглянуті технології, які використовуються у проекті з розробки програмної системи моніторингу зарядних станцій електромобілів, з фокусом на використання Kubernetes та Docker для оркестрації та контейнеризації додатків.

Kubernetes є основною технологією для оркестрації та управління контейнерами. Він дозволяє ефективно розгортати, масштабувати та керувати

контейнеризованими додатками в середовищі хмарних обчислень або власного дата-центру.

Використання Kubernetes дозволяє автоматизувати процеси управління ресурсами, моніторингу стану додатків, автоматичного відновлення в разі відмов та багато іншого.

Docker є відомим інструментом для контейнеризації додатків. Він дозволяє запаковувати додатки та їх залежності у контейнери, що забезпечує ізольоване та переносне середовище для виконання програм.

Використання Docker дозволяє швидко розгортати та тестувати додатки в будь-якому середовищі, зменшуючи проблеми зі сумісністю та конфліктами середовищ.

Контейнеризація додатків:

– для кожного компонента програмної системи, такого як веб-сервер, база даних, служба моніторингу тощо, створюються відповідні Docker контейнери. Це забезпечує ізольоване та незалежне виконання кожного компонента;
– Dockerfile використовується для визначення налаштувань контейнера, таких як базовий образ, необхідні пакети, команди для запуску додатків тощо.

Kubernetes конфігурація:

– Для управління та розгортання контейнерів з використанням Kubernetes створюються файли конфігурації, такі як Deployment, Service, ConfigMap тощо.
– Ці файли визначають параметри масштабування, кількість реплік, мережеві політики та інші налаштування для кожного компонента системи.

Такий стек технологій дозволяє ефективно розгортати, масштабувати та управляти програмною системою, забезпечуючи гнучкість, надійність та високу продуктивність у виконанні завдань.

Для управління базами даних у проекті обрано систему Microsoft SQL Server (MS SQL Server). Вибір цієї конкретної СУБД обумовлений рядом факторів. По-перше, MS SQL Server володіє розширеними можливостями для оптимізації запитів, моніторингу та забезпечення безпеки даних. Крім того, враховуючи використання мови програмування C# та платформи .NET, MS SQL Server є оптимальним вибором, оскільки він добре інтегрується з цими технологіями та забезпечує високий рівень сумісності. Також важливим аспектом є масштабованість MS SQL Server, яка дозволяє масштабувати бази даних від невеликих проектів до великих корпоративних систем. Офіційна підтримка та постійні оновлення забезпечують безпеку, стабільність та продуктивність роботи з базою даних.

Архітектура клієнтської частини "Onion" є однією з варіацій архітектурного шаблону "Clean Architecture" (Чиста архітектура). Вона спроектована для створення високоякісних, розширюваних і підтримуваних клієнтських додатків. Основна ідея Onion архітектури полягає в розділенні додатку на логічні шари, що забезпечує збереження чистоти коду, зниження залежностей та полегшення тестування.

Клієнтська частина розробляється за допомогою декількох технологій.

Angular є сучасним фреймворком для розробки веб-додатків, який використовується для створення клієнтської частини системи.

NGRX є бібліотекою для керування станом додатків, яка ґрунтується на патерні Redux. Вона дозволяє ефективно організовувати стан додатку, управляти даними та забезпечувати їхню синхронізацію між компонентами.

RxJS - це бібліотека для реактивного програмування в JavaScript, яка забезпечує можливість легкої роботи з асинхронними подіями та потоками даних.

Socket.IO - це бібліотека для реалізації багатоканального зв'язку в реальному часі між клієнтом та сервером. Вона використовує WebSocket та інші протоколи для забезпечення швидкої та надійної комунікації.

Cypress - це інструмент для автоматизованого тестування клієнтської частини веб-додатків. Він дозволяє створювати та виконувати тести для перевірки функціональності, інтерфейсу користувача та інших аспектів додатку.

Jasmine/Karma - це комбінація фреймворку для тестування Jasmine та інструменту для запуску тестів Karma. Вони використовуються для написання та виконання юніт-тестів та інтеграційних тестів Angular додатків.

Bootstrap 5 - це популярний CSS фреймворк, який використовується для створення стильного та адаптивного дизайну веб-додатків. Він надає готові компоненти, сітку та стилізацію елементів для швидкого розроблення інтерфейсів.

При проектуванні та розробці клієнтської частини потрібно врахувати обмеженість користувацького інтерфейсу та дотримуватись стандартам розробки. UI розробка та проектування - це комплексний процес, що вимагає уваги до деталей, досліджень та співпраці з командою для створення високоякісного та зручного для використання користувацького інтерфейсу.

Користувацький інтерфейс повинен бути зручним та інтуїтивно зрозумілим для користувачів. При розробці слід уникати перевантаження інтерфейсу зайвими елементами та функціоналом, а також забезпечити його адаптивність для різних пристроїв та розмірів екранів.

Важливо також враховувати потреби та вимоги цільової аудиторії при розробці інтерфейсу. Це може включати в себе дизайн-методології, такі як Material Design або Bootstrap (який і було обрано), для створення сучасного та приємного для використання інтерфейсу.

У системі необхідно забезпечити час відгуку на перехід між вкладками веб-додатку менше 1 секунди для забезпечення зручного та швидкого користувацького досвіду. Для цього важливо оптимізувати швидкість завантаження та відображення контенту на кожній вкладці, уникати зайвих запитів до сервера та використовувати кешування для прискорення доступу до даних.

Надійність системи є також важливим аспектом. У випадку виникнення помилки система повинна продовжувати працювати, а користувач повинен отримати зрозуміле повідомлення про проблему без виведення коду помилки або

іншої технічної інформації, яку може зрозуміти тільки розробник. Це допомагає зберегти позитивне враження від користування системою та дозволяє користувачам ефективно вирішувати проблеми без необґрунтованої турботи про технічні деталі.

Веб-додаток повинен бути розроблений з урахуванням сумісності з усіма сучасними браузерами, такими як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо. Використання технологій HTML5, CSS3 та JavaScript ES6 допоможе забезпечити сумісність та підтримку різних браузерів.

Адаптивний дизайн забезпечить коректне відображення веб-додатка на пристроях з різними розмірами екрану, включаючи мобільні пристрої, планшети та настільні комп'ютери.

Веб-додаток повинен підтримувати мови інтерфейсу українська та англійська для забезпечення доступності користувачам з різних країн та мовних груп.

При використанні зображень, фото та інших матеріалів на сайті слід дотримуватись авторських прав та використовувати лише ліцензований або вільний від авторських прав контент.

Система повинна бути стабільною та надійною. При виникненні помилок користувачам повинні відображатися зрозумілі повідомлення про проблему без технічної інформації.

3.3 Забезпечення якості

Забезпечення якості продукту з точки зору менеджменту є ключовим аспектом у процесі розробки та випуску програмного продукту. Формулювання стратегії якості:

- менеджмент повинен визначити стратегію якості продукту, яка відображає вимоги до якості, цілі та плани щодо забезпечення якості на різних етапах розробки;

– методика SMART (Specific, Measurable, Achievable, Relevant, Time-bound) для постановки конкретних, вимірюваних, досяжних, зв'язаних із завданням та маючих обмеження в часі цілей щодо якості продукту.

Визначення стандартів якості:

– розробка та утримання стандартів якості, які включають в себе вимоги до функціональності, відмовостійкості, ефективності, безпеки та інших аспектів продукту;
– використання стандартів якості ISO 9001:2015 або інших відомих нормативних документів для розробки та встановлення системи управління якістю в організації.

Планування та виконання тестування:

– розробка плану тестування, який включає в себе різноманітні види тестів, такі як модульні тести, інтеграційні тести, системні тести, відмовостійкість, безпеку тощо. Важливо також визначити критичні сценарії тестування та покриття коду тестами.

Внутрішній аудит якості:

– проведення регулярних внутрішніх аудитів якості продукту, які дозволяють ідентифікувати потенційні проблеми та вдосконалювати процеси розробки та контролю якості;
– використання інструментів для проведення код-рев'ю та аналізу якості коду.

Стеження за метриками якості:

– встановлення ключових метрик якості, таких як частота виявлення дефектів, час відгуку на проблеми, задоволеність користувачів, виконання термінів тощо. Моніторинг цих метрик допомагає вчасно виявляти проблеми та приймати необхідні заходи для покращення якості.
– використання інструментів для моніторингу якості продукту, таких як Jira для відстеження дефектів та завдань;
– впровадження методології Agile з щоденними стендапами та регулярними ретроспективами для швидкого реагування на виникаючі ризики та зміни.

Управління ризиками:

– визначення потенційних ризиків для якості продукту та розробка стратегій їх управління. Це може включати в себе регулярні оцінки ризиків, розробку планів запобігання та вирішення проблем;
– використання методів ризик-менеджменту, таких як SWOT-аналіз (Strengths, Weaknesses, Opportunities, Threats) для виявлення потенційних ризиків та розробки стратегій їх управління.

Неперервне вдосконалення:

– впровадження процесу неперервного вдосконалення якості, такого як методика PDCA (Plan-Do-Check-Act), що дозволяє систематично вдосконалювати процеси розробки та забезпечення якості;
– використання інструментів для управління проектами та завданнями

Ці підходи спрямовані на забезпечення того, щоб програмний продукт відповідав вимогам якості, був надійним та забезпечу.

3.4 Керування проектом

Керування проектом - це комплексний процес планування, організації, виконання та контролю за реалізацією проекту з метою досягнення його цілей та завдань в умовах обмеженого бюджету, ресурсів і часу. Основні аспекти керування проектом включають у себе:

– Планування. Визначення мети проекту, визначення завдань, ресурсів, термінів виконання, а також встановлення критеріїв успішності.
– Організація. Створення структури команди проекту, розподіл обов'язків, управління ресурсами та розвиток графіка виконання завдань.
– Виконання. Реалізація плану, виконання завдань, координація робіт, контроль за витратами та виконанням графіка.
– Контроль. Моніторинг і аналіз прогресу проекту, виявлення відхилень від плану, прийняття відповідних корективних заходів.
– Звітність. Підготовка звітів про прогрес та результати проекту для стейкхолдерів.

– Комунікації. Встановлення ефективного зв'язку між учасниками проекту, обмін інформацією та рішеннями.

Керування проектом базується на різних методологіях та підходах, таких як Waterfall, Agile, Scrum тощо, які використовуються в залежності від специфіки проекту, потреб замовника та умов виконання.

Основна мета керування проектом - досягнення успішного завершення проекту з відповідними результатами та задоволення вимог стейкхолдерів.

В керуванні проектом буде використовуватись гнучка філософія Agile. А метод Scrum є однією із найпоширеніших реалізацій Agile-підходу [10].

Скрам базується на головних ідеях Agile :

– Учасники процесу та їх комунікація важливіші за інструменти і сам процес;
– Працюючий продукт важливіший за докладну документацію;
– Постійна взаємодія із замовником важливіша за узгодження договору;
– Готовність змінювати і модифікувати продукт і робочий процес важливіший за строге дотримання плану.

Головна мета цих принципів — забезпечити на виході продукт, який максимально відповідатиме потребам та очікуванням замовника. Без гнучкої методології управління проєктом це завдання дуже складно вирішити.

Scrum став революційним методом, який, з одного боку, спрощує координацію роботи та оцінку ефективності команди, а з іншого — дає можливість швидко вносити зміни та адаптувати процес до нових актуальних завдань [11].

Базова часова одиниця Scrum – спринт. В рамках цього робочого відрізка проводиться ціла низка зустрічей-нарад (мітинги, від англ. meeting):

– Sprint Planning (планування спринту);
– Daily Scrum (щоденний звіт);
– Sprint Review (презентація/обговорення результатів спринту);
– Sprint Retrospective (оцінка підсумків спринту та робочого процесу, висновки) (див. рис. 4.1).

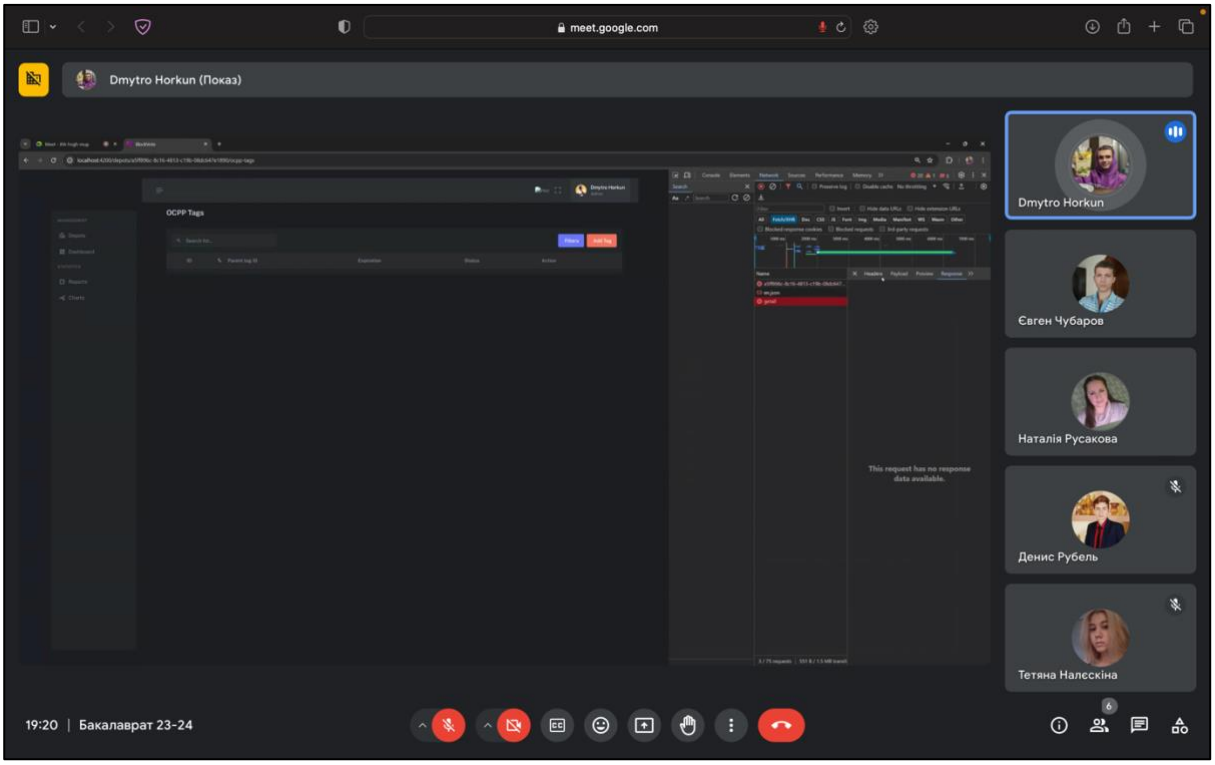


Рисунок 3.1 - Sprint Retrospective у Google Meet (рисунок виконано самостійно)

Як середовище керування та моніторингу було вирішено використовувати Jira. Jira - це інструмент управління проектами та задачами, розроблений для підтримки гнучких методологій, таких як SCRUM та Agile. З точки зору керування проектом, Jira надає широкий набір можливостей для планування, виконання, відстеження прогресу та аналізу результатів проектів.

Створення проектів. Jira дозволяє створювати різні типи проектів, такі як програмні проекти і налаштовувати їх під конкретні потреби.

Створення задач. Користувачі можуть створювати різні типи задач (user stories, bugs, tasks тощо), визначати їх пріоритетність, встановлювати терміни виконання та призначати відповідальних.

Планування та виконання завдань. Jira надає можливість планувати роботу на основі спринтів (для SCRUM), створювати розклади виконання задач та відстежувати прогрес команди.

Відстеження прогресу. Завдяки графікам, діаграмам та звітам, Jira дозволяє відстежувати прогрес виконання проекту, оцінювати часові рамки та витрати ресурсів.

Керування завданнями. Jira надає можливість призначати, перезначати та розподіляти завдання між учасниками команди, а також відстежувати їх статуси та взаємозв'язки.

Комунікація та співпраця. У Jira є можливість коментувати задачі, спільно працювати над документацією, взаємодіяти з колегами та стейкхолдерами через систему коментарів та сповіщень.

Аналіз результатів. За допомогою аналітичних інструментів, Jira дозволяє аналізувати ефективність роботи команди, виявляти проблемні зони та знаходити шляхи для покращення процесів.

Jira допомагає командам ефективно керувати проектами, забезпечуючи чітку структуру, організацію робочих процесів та сприяючи комунікації в команді.

Дошка програмної системи моніторингу зарядних станцій електромобілів нашої команди (див. рис. 4.2)

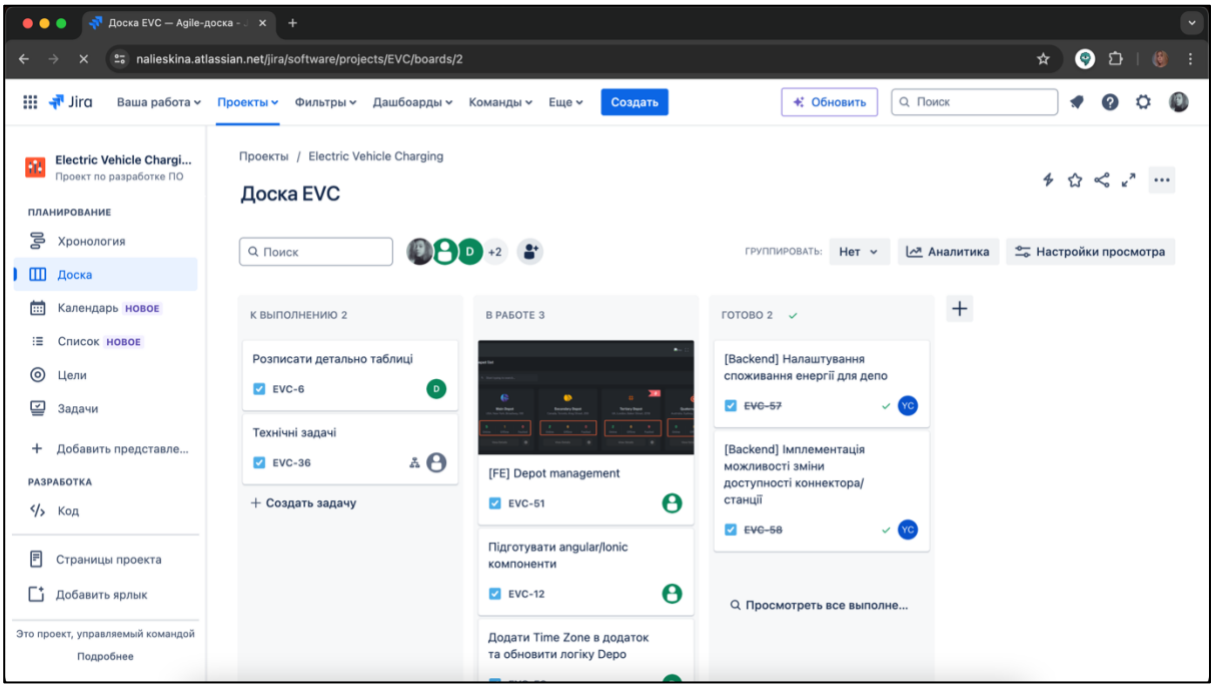


Рисунок 3.2 - Дошка у Jira (рисунок виконано самостійно)

У Jira дошка (board) є інструментом, який допомагає візуалізувати та організувати задачі та робочі процеси у рамках проекту. Дошки у Jira можуть бути

сконфігуровані під різні методології розробки, такі як SCRUM, та відображати поточний статус завдань, їх прогрес та взаємодію команди.

3.5 Тестування

Програмна система моніторингу зарядних станцій електромобілей проходить етап мануального тестування та тестування навантаженням.

Ручне тестування – це тип тестування програмного забезпечення, при якому тестовий кейс виконується тестувальником вручну без допомоги будь-яких автоматизованих інструментів [12].

Існує кілька етапів, на яких використовуються ручне тестування, перший – на етапі розробки базового функціоналу.

Коли основна функціональність програмного забезпечення знаходиться в стадії розробки, тестують роботу кожної частини програми вручну, оскільки це швидше, ніж створювати тестові кейси для досить простих частин коду.

Ручне тестування також переважає на останніх етапах розробки, коли програма має створений інтерфейс користувача. Тестування інтерфейсу передбачає перевірку того, як реальний користувач реагує на те, як розроблено меню і як працює система.

Оскільки це пов’язано з великою кількістю якісних даних і особистої думки, а не з чистими кількісними показниками, ручне тестування є ідеальним варіантом для отримання більш глибокого розуміння продукту.

Життєвий цикл мануального тестування складається з декількох етапів, що передбачають певні дії та процеси для впевненості в якості програмного продукту.

- планування тестування;
- аналіз вимог та документування тест-кейсів;
- підготовка тестового середовища;
- виконання тест-кейсів згідно з планом тестування та документацією, виявлення та реєстрація дефектів;
- аналіз результатів тестування;
- виправлення та перевірка дефектів;

– підготовка до повторного тестування;

Цикл мануального тестування може бути повтореним декілька разів для підтвердження якості програмного продукту перед його випуском.

Тест-кейс – це такий план дій, який допомагає нам перевірити, що наше програмне забезпечення працює правильно. Це як інструкція, якої ми дотримуємося, щоб переконатися, що все йде добре [13].

Тест-план (Test plan) – це документ, який описує весь обсяг робіт з тестування, починаючи з опису об'єкта тестування, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення [14].

Тестування буде проводить на основі методів білого і чорного ящику.

Тестування методом «чорного ящика», також відоме як тестування, засноване на специфікації або тестування поведінки – техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами тестованої системи.

Тестування методом білого ящика (також: прозорого, відкритого, скляного ящика; засноване на коді або структурне тестування) – метод тестування програмного забезпечення, який передбачає, що внутрішня структура/пристрій/реалізація системи відомі тестувальнику. Ми вибираємо вхідні значення, грунтуючись на знанні коду, який буде їх обробляти. Точно так само ми знаємо, яким повинен бути результат цієї обробки. Знання всіх особливостей тестованої програми та її реалізації – обов’язкові для цієї техніки. Тестування білого ящика – поглиблення у код системи, за межі її зовнішніх інтерфейсів [15].

Також система буде проходите через тестування навантаженням.

Тестування навантаження (Load Testing) – тестування часу відповіді програми на запити різних типів, з метою переконатися, що додаток працює відповідно до вимог при звичайному користувацькому навантаженні [16].

Під час тестування навантаженням проводяться такі дії:

– створення навантаження. Створюються умови, що спричиняють підвищене навантаження на систему, наприклад, велика кількість запитів, одночасні виклики, великі обсяги даних.

- вимірювання продуктивності. Вимірюються показники швидкості, реакції системи на навантаження, час відгуку, завантаженість ресурсів (пам'ять, процесор, мережа) тощо.
- аналіз результатів. Оцінюється реакція системи на підвищене навантаження, виявляються можливі проблеми з продуктивністю, швидкість реакції та інші аспекти, які потребують оптимізації.
- оптимізація та вдосконалення. На основі результатів тестування приймаються рішення щодо оптимізації системи, усунення проблем з продуктивністю та підвищення її надійності та швидкості реакції.

Таке тестування допомагає забезпечити, що система залишиться стабільною та ефективною при реальному використанні користувачами або при великому обсязі даних.

4 ОПИС ПРИЙНЯТИХ РІШЕНЬ

У сучасному світі електромобілі набирають все більшої популярності завдяки своїй екологічності та економічній вигідності. Одним з ключових аспектів успішного впровадження та використання електромобілів є наявність розвиненої інфраструктури зарядних станцій. Ефективне управління та моніторинг таких станцій дозволяє забезпечити безперервність роботи, оптимізацію ресурсів та підвищення задоволеності користувачів.

Метою даної дипломної роботи є розробка програмної системи моніторингу зарядних станцій електромобілів. Ця система буде включати функції управління проектом, формування вимог, а також проведення мануального та навантажувального тестування. У розділі буде детально розглянуто PEST та SWOT аналіз продукту, що дозволить визначити ключові фактори, які впливають на розвиток та впровадження системи.

Методика PEST аналізу часто використовується для оцінки ключових ринкових тенденцій галузі, а результати PEST аналізу можна використовувати для визначення списку загроз і можливостей при складанні SWOT аналізу компанії.

Зазвичай виконується у форматі таблиці, що складає 4 квадрати, кожен з яких відповідає за окремий напрямок: політичний, економічний, соціальний, технологічний.

Проводячи PEST аналіз, рекомендовано описувати не лише поточний стан кожного фактора, а прогнозувати його зміни на найближчі 3-5 років. Саме оцінка впливу фактора в довгостроковій перспективі на прибуток компанії, дозволяє застосовувати отримані дані для формування стратегії.

Якщо компанія реалізує свої товари на різних ринках і функціонує в різних галузях - рекомендується проводити PEST аналіз для кожної галузі та для кожного ринку [17].

Політичні (Political). Вивченню підлягає:

- національна та світова політична ситуація;
- передумови до змін;
- втручання держави у сферу діяльності компанії;

– податкова політика;

– рівень бюрократії та корупції;

– тенденції до виникнення політичної кризи.

Економічні (Economic) аналізують:

– показники доходів громадян;

– рівень інфляції;

– стан ринку праці, зокрема й рівень безробіття;

– динаміку внутрішнього валового продукту;

– зміни в курсі валют;

– зміни на ринках, від яких залежить компанія;

– динаміку економічного розвитку;

– вартість енергоносіїв та сировини.

Соціально-культурні (Social). Розгляду підлягають:

– демографічні показники та їхня динаміка;

– розвиток та зміна споживчих вподобань;

– специфіка менталітету та низка цінностей, притаманних середовищу;

– ступінь освіченості населення (зокрема й кваліфікація кадрів).

Технологічні (Technological). Аналіз стосується:

– політики держави в галузі інноваційних розробок;

– рівня стану технологій в індустрії;

– впливу нововведень на бізнес;

– появи відкриттів, патентів, продуктів тощо;

– доступу компанії до технологій;

– розвитку інтернету, мобільних пристроїв, їхній вплив на компанію.

SWOT — це ефективний інструмент бізнес-планування, який використовується в бізнесі для формування стратегій. Цей інструмент допомагає проаналізувати внутрішні фактори (сильні та слабкі сторони), які впливають, і зовнішні фактори (можливості та загрози), які можуть мати вплив на організацію [18].

Абревіатура SWOT складається зі скорочень слів і означає наступне:

- S (Strengths) – сильні сторони. Унікальні характеристики та переваги, які дають можливість виділятися бізнесу на тлі конкурентів. Завдяки їм підприємство може збільшити свій прибуток. Наприклад, досить великий вибір товару, хороший сервіс, більш доступні ціни, сучасне обладнання, лідер галузі.
- W (Weaknesses) – слабкі сторони. Недоліки, які не дають належною мірою розвиватися компанії, гальмують зростання її прибутку і роблять більш вразливою по відношенню до конкурентів. Наприклад, недостатня кількість співробітників, зрив термінів доставки, невеликий асортимент, низька якість товару, старе обладнання.
- (Opportunities) – можливості. Компоненти оточення, які можуть поліпшити становище і стан бізнесу за умови їх використання. Наприклад, кваліфіковані співробітники, правильне розміщення виробництва, відсутність сильних конкурентів, спонсорство.
- T (Threats) – загрози. Компоненти оточення, через які компанія може якось постраждати, втратити клієнтів або прибуток. Наприклад, велика конкуренція, фінансова криза, нестабільна ситуація в країні, поява сильного конкурента.

PEST аналіз включає вивчення політичних, економічних, соціальних та технологічних аспектів, що впливають на проект. Цей аналіз допоможе зрозуміти зовнішнє середовище та ідентифікувати можливі загрози та можливості. SWOT аналіз, у свою чергу, дозволить оцінити внутрішні сильні та слабкі сторони проекту, а також виявити зовнішні можливості та загрози. Комплексний аналіз цих факторів сприятиме прийняттю обґрунтованих рішень та підвищенню ефективності реалізації проекту.

PEST аналіз для нашого продукту (див. рис. 4.1.)

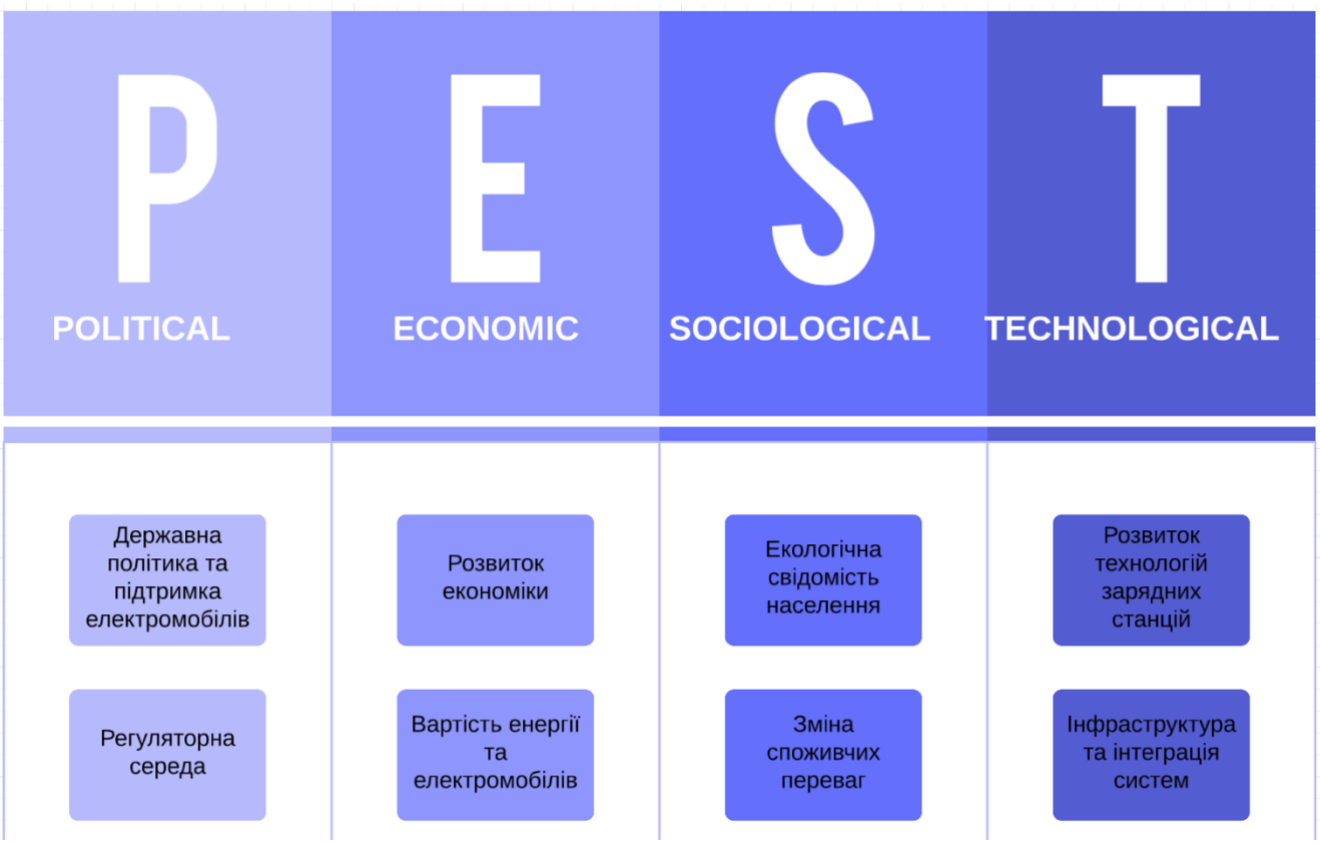


Рисунок 4.1 - PEST аналіз продукту (рисунок виконано самостійно)

PEST аналіз для реалізації продукту в Україні:

Політичні фактори (Political).

а) Державна політика та підтримка електромобілів:

1) уряд України активно підтримує розвиток електромобільного транспорту через різні стимули, включаючи зниження податків на імпорт електромобілів та надання субсидій для встановлення зарядних станцій;

2) впровадження нормативно-правових актів, спрямованих на розширення інфраструктури для електромобілів, що сприяє збільшенню кількості зарядних станцій.

б) Регуляторна середа:

1) законодавчі вимоги щодо стандартів безпеки та технічних норм для зарядних станцій, які можуть впливати на проектування та впровадження системи моніторингу.

2) можливість змін у законодавстві, що стосуються використання електромобілів та зарядних станцій, може вплинути на стратегію розвитку проекту.

Економічні фактори (Economic).

а) Розвиток економіки:

1) стан економіки України та інвестиційний клімат впливають на фінансування проектів, пов'язаних з електромобільною інфраструктурою.
2) доступність фінансування та кредитування для встановлення зарядних станцій може стимулювати попит на програмну систему моніторингу.

б) Вартість енергії та електромобілів:

1) ціни на електроенергію можуть впливати на економічну доцільність використання електромобілів та зарядних станцій.
2) зниження вартості електромобілів сприяє їх більшому поширенню, що, у свою чергу, збільшує попит на інфраструктуру зарядних станцій.

Соціальні фактори (Social).

а) Екологічна свідомість населення:

1) зростаюча екологічна свідомість українців підвищує інтерес до електромобілів як екологічно чистого транспорту.
2) підтримка громадськості щодо зменшення викидів вуглецю та використання відновлюваних джерел енергії сприяє розвитку інфраструктури для електромобілів.

б) Зміна споживчих переваг:

1) збільшення попиту на електромобілі серед населення, що спричиняє зростання потреби в доступних та надійних зарядних станціях.
2) вплив демографічних змін та урбанізації на споживчі переваги, що може вплинути на розташування та використання зарядних станцій.

Технологічні фактори (Technological).

а) Розвиток технологій зарядних станцій:

1) швидкий розвиток технологій зарядки, включаючи швидкі зарядні станції та бездротову зарядку, які потребують сучасних систем моніторингу.
2) інновації у галузі програмного забезпечення для моніторингу та управління зарядними станціями.

б) Інфраструктура та інтеграція систем:

1) наявність розвиненої телекомунікаційної інфраструктури для забезпечення надійного зв'язку між зарядними станціями та центральною системою моніторингу.
2) можливості інтеграції з іншими системами, такими як розумні мережі, для оптимізації використання енергії.

PEST аналіз показав, що проект розробки програмної системи моніторингу зарядних станцій електромобілів в Україні має значний потенціал для успішного впровадження завдяки сприятливим політичним, економічним, соціальним та технологічним факторам. Однак необхідно уважно стежити за можливими змінами у зовнішньому середовищі та бути готовими адаптувати стратегію проекту для забезпечення його успішної реалізації та довготривалого розвитку.

Pest аналіз для реалізації продукту для реалізації в країні Євросоюзу, для прикладу взято Німеччину.

Політичні фактори (Political)

а) державна політика та підтримка електромобілів:

1) уряд Німеччини активно підтримує розвиток електромобільного транспорту, надаючи субсидії та податкові пільги на покупку електромобілів та встановлення зарядних станцій;
2) чітка стратегія Німеччини щодо зниження викидів парникових газів і досягнення кліматичних цілей стимулює розвиток електромобільної інфраструктури.

б) Регуляторна середа:

1) строгі нормативні вимоги щодо безпеки та технічних стандартів для зарядних станцій, що можуть впливати на проектування та впровадження системи моніторингу;
2) політична стабільність і сприятливе законодавче середовище сприяють залученню інвестицій у цей сектор.

Економічні фактори (Economic).

а) Розвиток економіки:

1) Німеччина має сильну економіку, яка підтримує інноваційні проекти та технологічні стартапи;
2) високий рівень інвестицій в екологічно чисті технології та проекти, пов'язані з розвитком електромобільної інфраструктури.

б) Вартість енергії та електромобілів:

1) ціни на електроенергію та доступність поновлюваних джерел енергії впливають на економічну доцільність використання електромобілів.
2) зниження вартості виробництва та збільшення доступності електромобілів сприяють їх більшому поширенню, що підвищує попит на інфраструктуру зарядних станцій.

Соціальні фактори (Social).

а) Екологічна свідомість населення:

1) високий рівень екологічної свідомості серед населення Німеччини стимулює попит на електромобілі та розвиток зарядних станцій;
2) підтримка громадськості щодо зниження викидів вуглецю та перехід на більш екологічно чисті види транспорту.

б) зміна споживчих переваг:

1) збільшення інтересу до електромобілів серед населення, що призводить до зростання потреби в доступних і надійних зарядних станціях;
2) вплив демографічних змін та урбанізації на попит на електромобілі та зарядні станції.

Технологічні фактори (Technological);

а) Розвиток технологій зарядних станцій:

1) Німеччина є лідером у сфері інновацій та технологій, включаючи розробку сучасних зарядних станцій та програмного забезпечення для їх моніторингу;
2) Високий рівень досліджень і розробок у галузі швидкісної зарядки та бездротових технологій зарядки.

б) Інфраструктура та інтеграція систем:

1) розвинена телекомунікаційна інфраструктура забезпечує надійний зв'язок між зарядними станціями та центральною системою моніторингу;
2) можливості інтеграції з іншими системами для оптимізації використання енергії та покращення ефективності зарядних станцій.

Проведений PEST аналіз програмної системи моніторингу зарядних станцій електромобілів у Німеччині виявив сприятливі умови для розвитку та впровадження цього продукту. Сильна державна підтримка та сприятливе законодавче середовище створюють основу для інвестицій та інновацій у цьому секторі. Стійка економіка та високий рівень екологічної свідомості населення стимулюють попит на електромобілі та зарядні станції. Розвинена технологічна та телекомунікаційна інфраструктура забезпечує надійну базу для реалізації проекту.

Порівняльний аналіз показує, що обидві країни мають сприятливі умови для розвитку програмної системи моніторингу зарядних станцій електромобілів, але кожна з них має свої особливості:

– політична підтримка. В обох країнах існує державна підтримка електромобілів, але Німеччина має більш розвинуту та стабільну політичну систему, що створює більш передбачуване середовище для інвесторів.
– економічні умови. Німеччина має сильнішу економіку та вищий рівень інвестицій в інноваційні проекти, тоді як економіка України ще розвивається, що може впливати на доступність фінансування.

– соціальні фактори. В обох країнах зростає екологічна свідомість, але в Німеччині цей процес більш виражений і підтримується активною громадською підтримкою.
– технологічні можливості. Німеччина є лідером у сфері технологій, що забезпечує впровадження найсучасніших рішень, тоді як в Україні технологічна інфраструктура ще розвивається.

Таким чином, обидві країни мають значний потенціал для розвитку програмної системи моніторингу зарядних станцій електромобілів, але для успішної реалізації проекту в Україні необхідно враховувати можливі виклики, пов'язані з економічною та технологічною складовими. Німеччина, зі своєю стабільною економікою та високим рівнем інновацій, надає більш сприятливі умови для впровадження та розвитку таких проектів.

SWOT аналіз для нашого продукту (див. рис. 4.2)

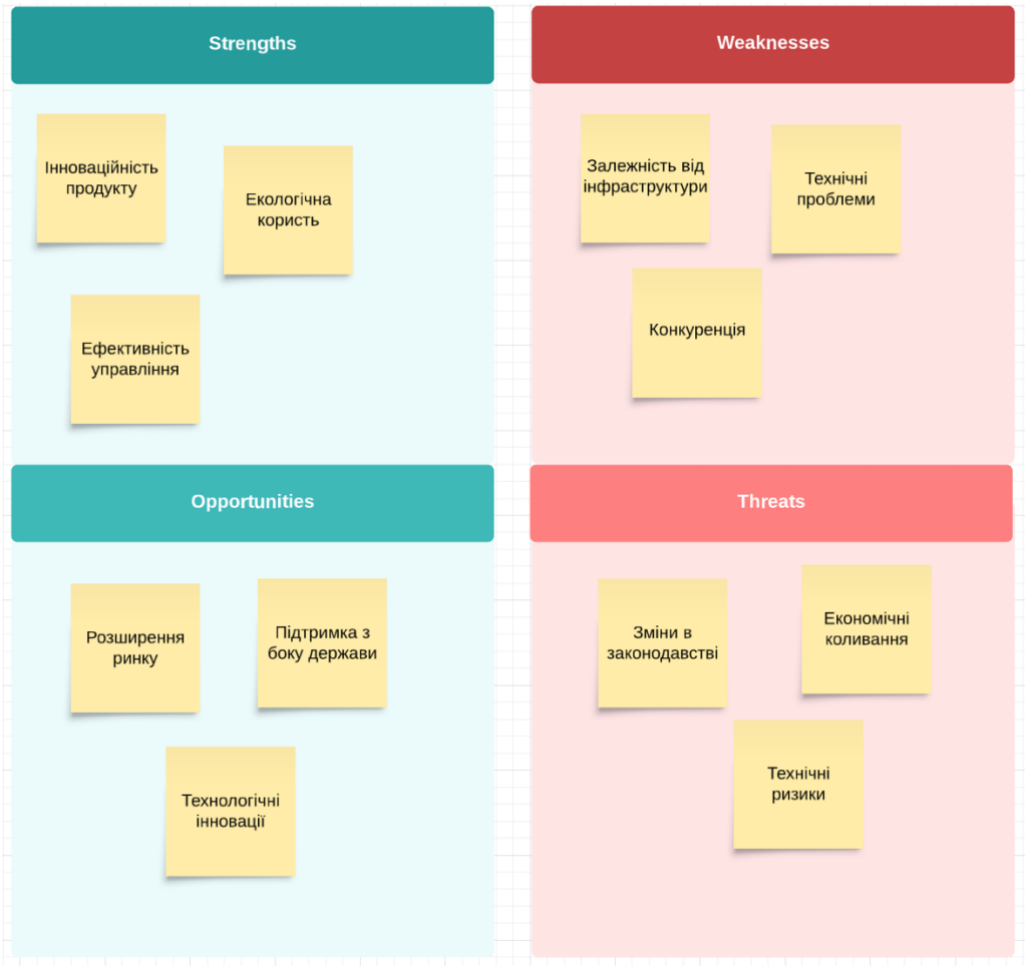


Рисунок 4.2 - SWOT аналіз для продукту (рисунок виконано самостійно)

SWOT аналіз для «Програмної системи моніторингу зарядних станцій електромобілів»

Сильні сторони (Strengths).

а) Інноваційність продукту:

1) сучасне програмне забезпечення, що включає передові технології моніторингу та управління зарядними станціями;
2) можливість інтеграції з іншими системами, такими як розумні мережі (smart grids), для оптимізації використання енергії.

б) Екологічна користь:

1) сприяння зниженню викидів парникових газів за рахунок підтримки інфраструктури для електромобілів;
2) підвищення екологічної свідомості користувачів.

в) Ефективність управління:

1) зниження витрат на обслуговування зарядних станцій через автоматизацію моніторингу та виявлення проблем;
2) покращення користувацького досвіду завдяки своєчасному оновленню інформації про стан зарядних станцій.

Слабкі сторони (Weaknesses).

а) Залежність від інфраструктури:

1) відсутність або недостатність телекомунікаційної інфраструктури в деяких регіонах може обмежувати функціональність системи;
2) високі початкові витрати на впровадження та встановлення системи.

б) Технічні проблеми:

1) можливі технічні збої та помилки в роботі програмного забезпечення;
2) необхідність постійного технічного обслуговування та оновлення системи.

в) конкуренція:

1) наявність інших подібних рішень на ринку, що може знизити конкурентоспроможність продукту;

2) високий рівень конкуренції серед постачальників технологій та послуг для електромобільної інфраструктури.

Можливості (Opportunities).

а) Розширення ринку:

1) зростання попиту на електромобілі в усьому світі стимулює розвиток інфраструктури зарядних станцій;

2) вихід на нові ринки та розширення географії присутності.

б) Підтримка з боку держави:

1) отримання субсидій та грантів від урядів для підтримки екологічних ініціатив;

2) співпраця з державними установами для розвитку національних програм з розвитку електромобільної інфраструктури.

в) Технологічні інновації:

1) впровадження нових технологій зарядки, таких як швидкісна та бездротова зарядка;

2) розвиток програмного забезпечення для більш ефективного моніторингу та управління зарядними станціями.

Загрози (Threats).

а) Зміни в законодавстві:

1) можливі зміни у законодавстві та регуляторних вимогах можуть вплинути на роботу системи та вимагати адаптації продукту.

2) невизначеність у регуляторному середовищі може стримувати інвестиції.

б) Економічні коливання:

1) економічна нестабільність або рецесія можуть знизити рівень інвестицій в інфраструктуру для електромобілів;

2) зміни в цінах на електроенергію можуть вплинути на економічну доцільність використання електромобілів.

в) Технічні ризики:

1) кіберзагрози та можливість хакерських атак на програмне забезпечення.

2) швидкий розвиток технологій, що може зробити існуючі рішення застарілими.

SWOT аналіз показує, що програмна система моніторингу зарядних станцій електромобілів має значний потенціал для розвитку завдяки своїм сильним сторонам та можливостям, таким як інноваційність продукту, екологічна користь, та підтримка з боку держави. Водночас, проекту слід враховувати наявні слабкі сторони, такі як залежність від інфраструктури та конкуренція, а також загрози у вигляді змін у законодавстві та економічних коливань. Ефективне використання можливостей та мінімізація ризиків допоможе забезпечити успішну реалізацію та довгостроковий розвиток проекту.

Роблячи висновок з PEST та SWOT аналізів, програмна система моніторингу зарядних станцій електромобілів має значний потенціал для успішного розвитку та впровадження як в Україні, так і в Німеччині. Сильні сторони та можливості перевищують слабкі сторони та загрози, що дозволяє оптимістично оцінювати перспективи цього продукту. Для забезпечення успіху проекту необхідно постійно адаптуватися до змін у зовнішньому середовищі, використовувати державну підтримку та інвестувати у нові технології для підтримання конкурентоспроможності.

Для менеджементу та керуванням проектом було вирішено використовувати спринти на базі беклогу.

Спринт — це короткий, зазвичай тривалістю від однієї до чотирьох тижнів, період, протягом якого команда виконує конкретний набір завдань, спрямованих на досягнення певної мети. Кожен спринт завершується створенням робочої версії продукту, яку можна продемонструвати зацікавленим сторонам.

Спринти, як правило, тривають від одного до чотирьох тижнів. Тривалість спринту визначається командою на початку проекту і може бути налаштована відповідно до потреб і специфіки проекту. Однак, наша команда залишилась на спринті тривалістю в 3 тижні.

Початок роботи команди - 17 січня 2024 року.

Перший спринт: Сбір і аналіз інформації, формування вимог, написання тез для конференцій, створення діаграм.

Збір і аналіз інформації. На початку першого спринту команда займається збором і аналізом необхідної інформації щодо проекту. Це включає вивчення вимог клієнта, дослідження ринку та аналіз конкурентів. Ключові завдання в цьому етапі:

- опитування клієнта і стейкхолдерів. В нашому випадку зустрічі та обговорення остаточної концепції дипломного продукту.
- аналіз ринку та конкурентів. Вивчення сучасних тенденцій у галузі, аналіз конкурентів, виявлення сильних та слабких сторін.
- аналіз технічних можливостей. Оцінка технічних аспектів розробки продукту, вибір технологій та інструментів.

Формування вимог. Після збору і аналізу інформації команда переходить до формування вимог для проекту. Це включає визначення функціональних, нефункціональних, технічних та інших вимог, які повинен відповідати продукт. Ключові етапи цього процесу:

- Складання вимог: Оформлення вимог у вигляді документації, яка містить чітко сформульовані та вимірювані критерії.
- Погодження вимог між членами команди. Проведення переговорів та обговорень для підтвердження та уточнення вимог.

Написання тез для конференцій. Одним із завдань першого спринту може бути написання тез для участі у конференціях або презентацій проекту перед зацікавленими сторонами. Це допомагає привернути увагу до проекту, залучити потенційних інвесторів та партнерів, а також отримати зворотний зв'язок від експертів та колег з галузі.

Створення діаграм. Іншим важливим елементом першого спринту може бути створення діаграм, таких як діаграми потоків роботи, архітектурні схеми, діаграми баз даних тощо. Це допомагає візуалізувати структуру та логіку продукту, а також вирішувати технічні питання щодо його реалізації.

Перший спринт визначається сбором і аналізом інформації, формуванням вимог, написанням тез для конференцій та створенням діаграм. Цей етап є

критичним для успішного початку проекту, оскільки він дозволяє команді зрозуміти вимоги, визначити напрямки розвитку та відзначитися перед широкою громадськістю чи зацікавленими сторонами.

Другий спринт був спрямований на початок програмної реалізації продукту.

Створення мікросервісів. Мікросервіси — це архітектурний підхід, який передбачає розбиття програмного додатку на невеликі, автономні сервіси, що працюють окремо один від одного. У другому спринті команда приступила до створення мікросервісів, які будуть відповідати за різні функціональні можливості продукту.

Створення UI. Разом з програмними сервісами команда працювала над створенням користувацького інтерфейсу (UI), який забезпечить зручний доступ до функціональності продукту для кінцевих користувачів. Це включало в себе розробку інтерактивних елементів, дизайн і реалізацію користувацьких взаємодій.

Додавання Docker та Ocelot. Крім того, у другому спринті було додано підтримку Docker для контейнеризації програмних сервісів. Docker дозволяє запускати, встановлювати та керувати контейнерами з програмним забезпеченням незалежно від операційної системи. Також був доданий Ocelot — це бібліотека, яка дозволяє налаштовувати та управляти шлюзами API, що полегшує реалізацію мікросервісної архітектури.

Імплементація контролерів та шлюзів. Команда приступила до імплементації контролерів для обробки запитів від користувачів у мікросервісах. Також був реалізований шлюз API за допомогою Ocelot, який забезпечує централізований доступ до різних сервісів та маршрутизацію запитів.

Другий спринт був насиченим етапом у програмній реалізації продукту. Команда успішно розпочала створювати мікросервіси, розробляла користувацький інтерфейс для зручного взаємодії з продуктом, впроваджувала Docker для контейнеризації сервісів та використовувала Ocelot для керування API шлюзами. Це становить важливий крок у напрямку реалізації функціональності та підготовки продукту до подальшого розвитку та випробувань (див. рис. 4.3).

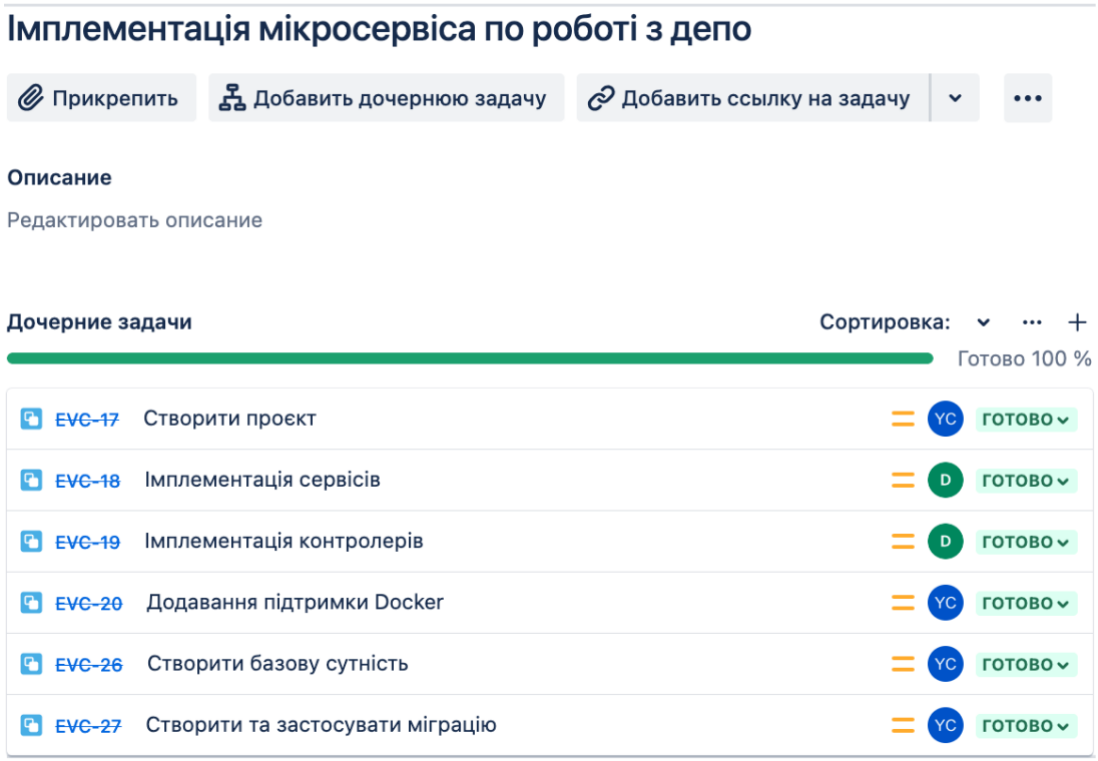


Рисунок 4.3 - Приклад виконаної задачі разом з дочірними задачами (рисунок виконано самостійно)

Третій спринт був присвячений декільком ключовим аспектам розробки продукту.

Обробка OCPP. Один з головних завдань у цьому спринті — це розробка обробки OCPP (Open Charge Point Protocol). Це протокол, який використовується для взаємодії зарядних станцій для електромобілів. Команда працювала над імплементацією цього протоколу для забезпечення функціональності зарядних станцій.

Обробка повідомлень з різних мікросервісів. Також у цьому спринті команда працювала над обробкою повідомлень, які надходять з різних мікросервісів. Це включало в себе розробку механізмів обробки та реагування на повідомлення з різних частин системи.

Додавання Azure. У третьому спринті також було додано підтримку Azure — хмарної платформи для розгортання, керування та моніторингу програмних додатків. Це дозволило команді використовувати різноманітні сервіси та інструменти Azure для покращення продукту.

Створення сторінок на клієнтській частині .Команда також працювала над розробкою клієнтської частини продукту, створюючи сторінки, інтерфейси та взаємодії для кінцевих користувачів.

Створення SRS документу та тест-планів. У третьому спринті було розпочато створення Software Requirements Specification (SRS) документу — це документ, який описує вимоги до програмного продукту. Також був початий процес створення тест-планів, які допоможуть у виконанні тестування продукту та перевірці відповідності вимогам.

Третій спринт був важливим етапом у розробці продукту, який включав в себе ряд ключових завдань, таких як обробка OCPP, робота з повідомленнями, додавання підтримки Azure, розробка клієнтської частини, створення SRS документу та тест-планів. Ці кроки покладають основу для подальшої розробки, тестування та вдосконалення продукту.

Четвертий спринт був спрямований на доробку проекту, проведення його тестування та виправлення виявлених помилок.

Доробка проекту. У четвертому спринті команда зосередилася на доробці функціональності продукту, враховуючи вимоги та відгуки від користувачів та тестерів. Це включало в себе додавання нових можливостей, виправлення помилок та оптимізацію роботи системи.

Тестування. Під час четвертого спринту було проведено інтенсивне тестування продукту для виявлення будь-яких дефектів, помилок або недоліків у функціональності. Це включало в себе ручне та автоматизоване тестування різних частин системи та взаємодій між ними.

Виправлення помилок. Після проведення тестування команда приступила до виправлення виявлених помилок. Це включало в себе аналіз причин помилок, їх усунення та перевірку коректності роботи після виправлень.

Оптимізація та підготовка до релізу. Після доробки проекту та виправлення помилок, команда провела оптимізацію системи для покращення її продуктивності та ефективності роботи. Також була проведена підготовка до релізу продукту,

включаючи документацію, збірку та тестування фінальної версії перед випуском на ринок.

Четвертий спринт відіграв важливу роль у завершенні проекту та підготовці його до випуску. Команда успішно доробила функціональність, виправила виявлені помилки, провела тестування та оптимізацію системи. Ці кроки дозволили підготувати продукт до релізу з вищим рівнем якості та надійності, що є ключовими факторами успіху на ринку.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мануальне тестування та тестування навантаженням є важливою частиною процесу тестування програмного забезпечення в рамках розробки програмної системи моніторингу зарядних станцій електромобілів.

Мануальне тестування полягає у виконанні тестів розроблених функціональних вимог програми вручну, без використання автоматизованих скриптів чи інструментів. Це важливо для виявлення проблем, які можуть залишитися непоміченими автоматизованим тестуванням.

- Створення тест-кейсів. Розробка детальних інструкцій для виконання кожного тесту, включаючи вхідні дані, кроки виконання тесту, очікуваний результат та підтвердження успішного/неуспішного виконання.
- Виконання тестів. Виконання тестів згідно зі створеними тест-кейсами та реєстрація результатів тестування, включаючи виявлені проблеми.
- Документування дефектів. Детальне описання виявлених проблем у вигляді дефектів з вказанням кроків для відтворення, серйозності проблеми та інших характеристик.
- Ретестування. Повторне виконання тестів для перевірки виправлення дефектів та підтвердження прийняття змін.

Тестування навантаженням (або стрес-тестування) спрямоване на визначення стійкості системи під навантаженням, перевірку швидкості реакції та визначення максимальної робочої межі системи.

- Планування тестування навантаженням. Визначення цілей тестування, вибір навантаження (кількість користувачів, обсяг даних тощо), планування сценаріїв тестування.
- Виконання тестування. Запуск тестів з навантаженням згідно зі сценаріями, моніторинг роботи системи під час тестування.
- Аналіз результатів. Оцінка відповіді системи на навантаження, виявлення слабких місць, оцінка продуктивності системи.
- Оптимізація та усунення проблем. Внесення змін у систему для покращення її продуктивності та виправлення виявлених проблем.

– Документування результатів. Опис отриманих результатів тестування навантаженням, включаючи знайдені проблеми та вжиті заходи їх виправлення.

Ці аспекти допоможуть ретельно вивчити процес мануального тестування та тестування навантаженням у програмі моніторингу зарядних станцій електромобілів.

Мануальне тестування проводилось як і напряму по посиланню ресурсу, так і через веб-сервіс BrowerStack [19]. Останній використовуємо переважно для тестування мобільного додатку.

BrowserStack - це провідна платформа для тестування програмного забезпечення, яка надає розробникам можливість тестувати веб- та мобільні додатки на реальних пристроях і в реальних умовах. Завдяки широкому набору інструментів та функцій, BrowserStack допомагає забезпечити високу якість та продуктивність додатків на різних платформах і пристроях.

Основні функції BrowserStack:

– тестування на реальних пристроях. BrowserStack надає доступ до тисяч реальних мобільних пристроїв та браузерів, що дозволяє розробникам тестувати свої додатки в умовах, максимально наближених до реальних;
– автоматизація тестування;
– тестування в хмарі. Платформа забезпечує хмарне середовище для тестування, що дозволяє зменшити витрати на підтримку фізичних лабораторій пристроїв та забезпечує швидкий доступ до необхідних ресурсів;
– крос-браузерне тестування (рис. 5.1). BrowserStack дозволяє тестувати додатки на різних браузерах і їх версіях, що допомагає забезпечити сумісність додатків з усіма популярними браузерами;
– тестування мобільних додатків (рис. 5.2 та рис. 5.3). Платформа підтримує тестування як на Android, так і на iOS пристроях, включаючи тестування на різних версіях операційних систем;

– перевірка доступності. BrowserStack пропонує інструменти для перевірки доступності веб-додатків, що дозволяє виявити та виправити проблеми з доступністю відповідно до стандартів WCAG (Web Content Accessibility Guidelines).

BrowserStack є потужним інструментом для забезпечення якості програмного забезпечення, що надає широкий спектр можливостей для тестування веб- та мобільних додатків. Завдяки своїй функціональності, сумісності з різними інструментами та можливості тестування на реальних пристроях, платформа допомагає розробникам створювати високоякісні та надійні додатки, що відповідають потребам користувачів.

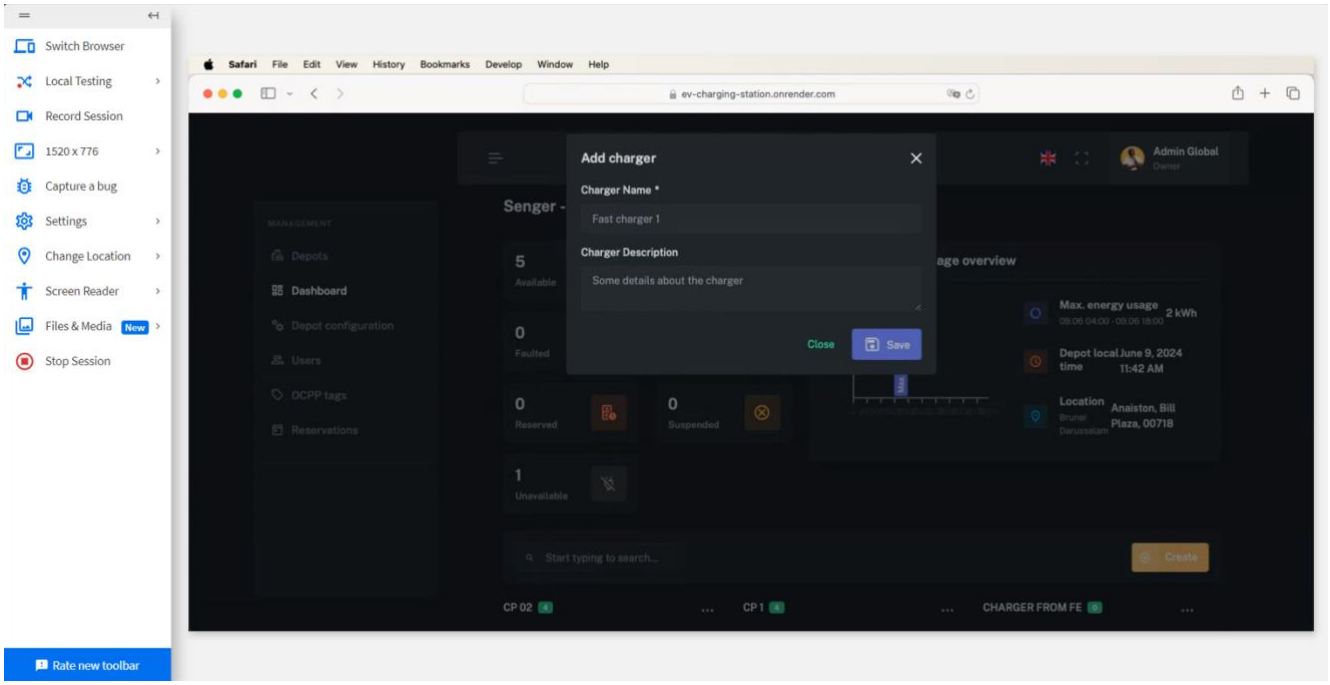


Рисунок 5.1 - Форма додавання розетки (рисунок виконано самостійно)

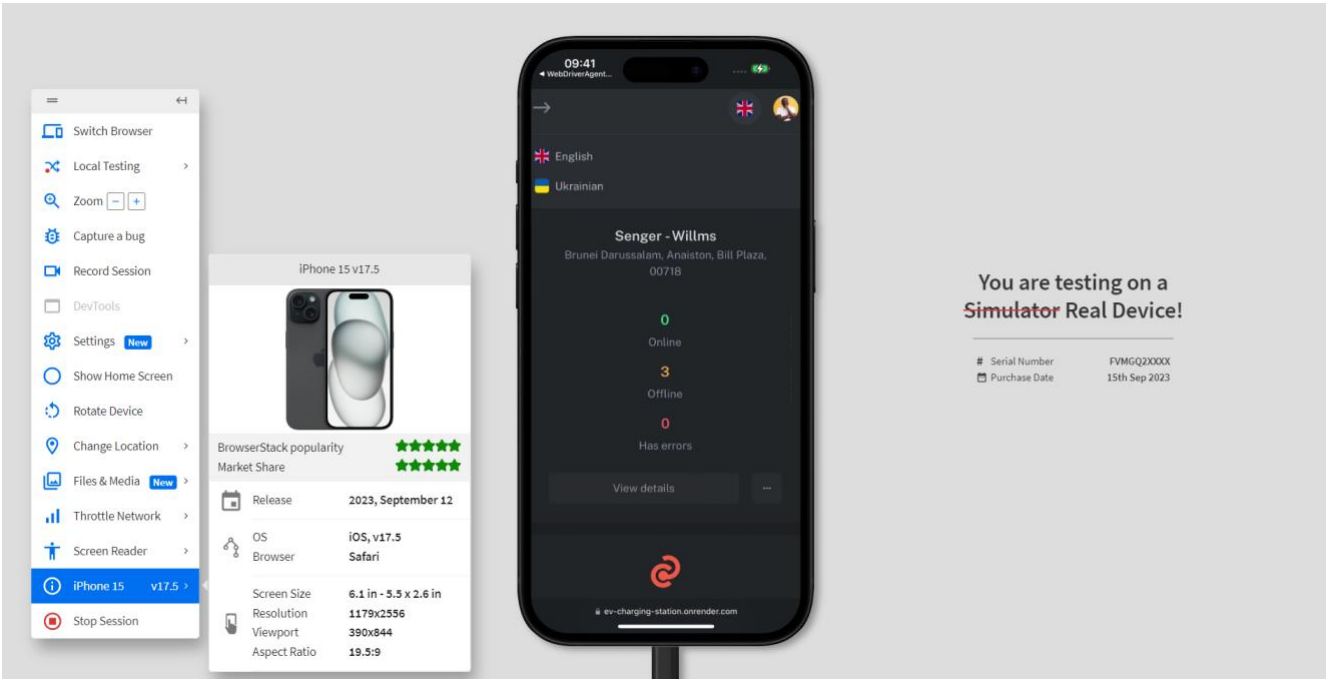


Рисунок 5.2 - Сторінка депо (рисунок виконано самостійно)

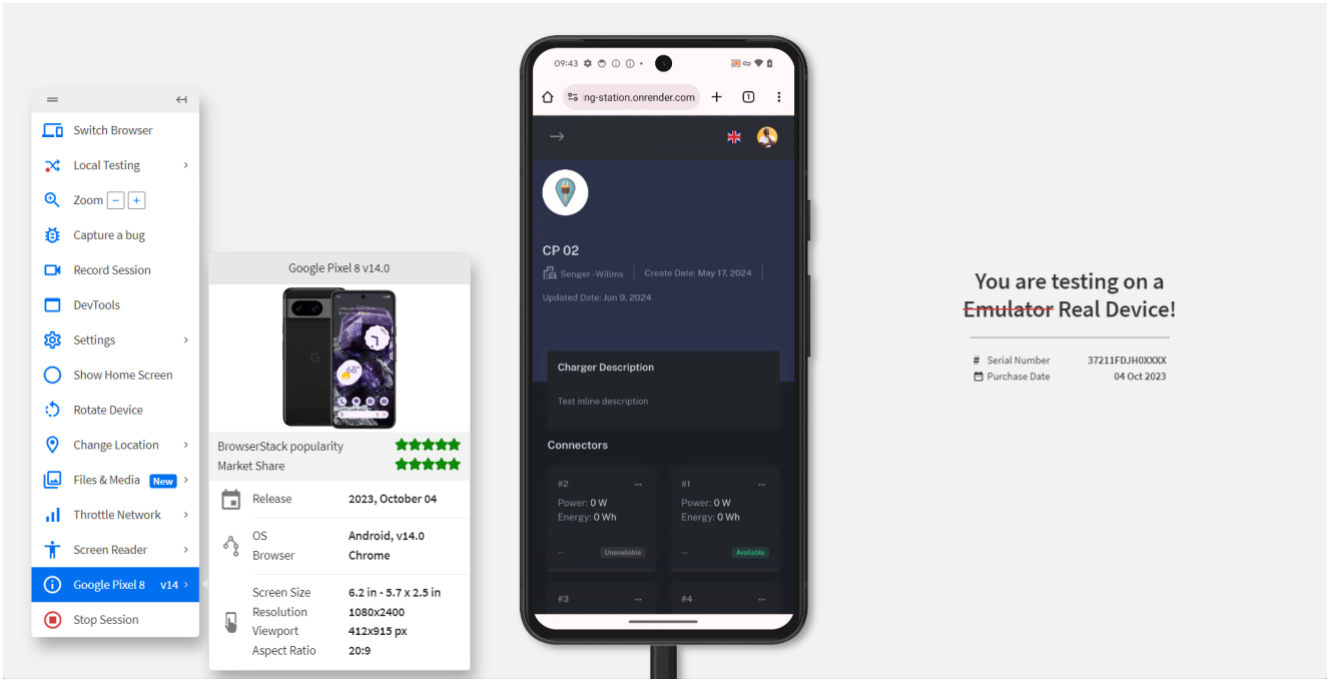


Рисунок 5.3 - Сторінка розетки (рисунок виконано самостійно)

Також ми отримали результат тестування доступності за допомогою сервісу BrowserStack (див. рис. 5.4).

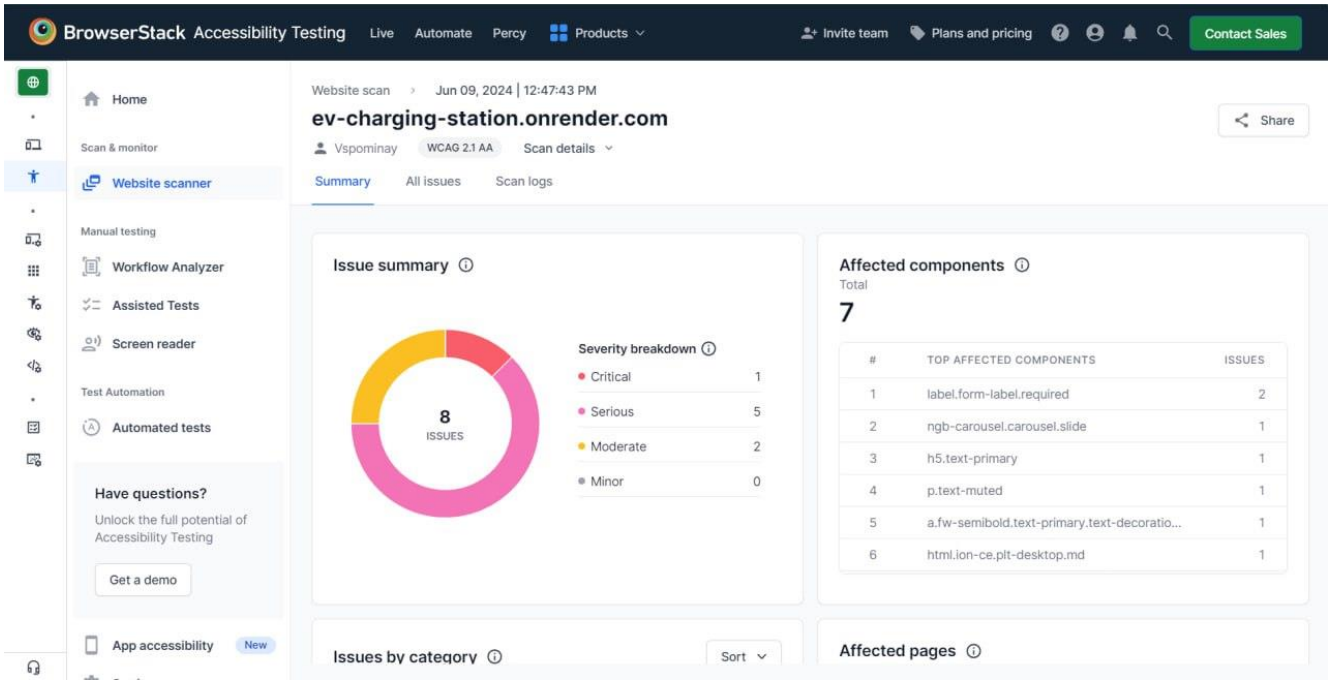


Рисунок 5.4 - Результат доступності (рисунок виконано самостійно)

Було виявлено 8 помилок, серед яких 5 серйозних, 1 критична, 2 помірні та 0 незначних.

Кроки для покращення доступності:

а) аналіз кожної проблеми:

1) критичні проблеми повинні бути виправлені першочергово, оскільки вони можуть значно впливати на доступність сайту;
2) серйозні проблеми також потребують негайного виправлення, оскільки вони можуть створювати значні труднощі для користувачів з обмеженими можливостями.

б) виправлення компонентів:

1) зосередимось на компонентах, які мають найбільше проблем (label.form-label.required) та вирішімо їх першочергово;
2) перегля коженого компоненту для переконання, що він відповідає стандартам WCAG 2.1 AA.

в) мануальне тестування:

1) використаємо мануальне тестування для підтвердження виправлень. Перевіримо, чи вирішено всі виявлені проблеми.

Першим кроком мануального тестування після виправлення помилок, які були виявлені BrowserStack протестуємо форму авторизації та реєстрації (див. рис. 5.5 та 5.6).

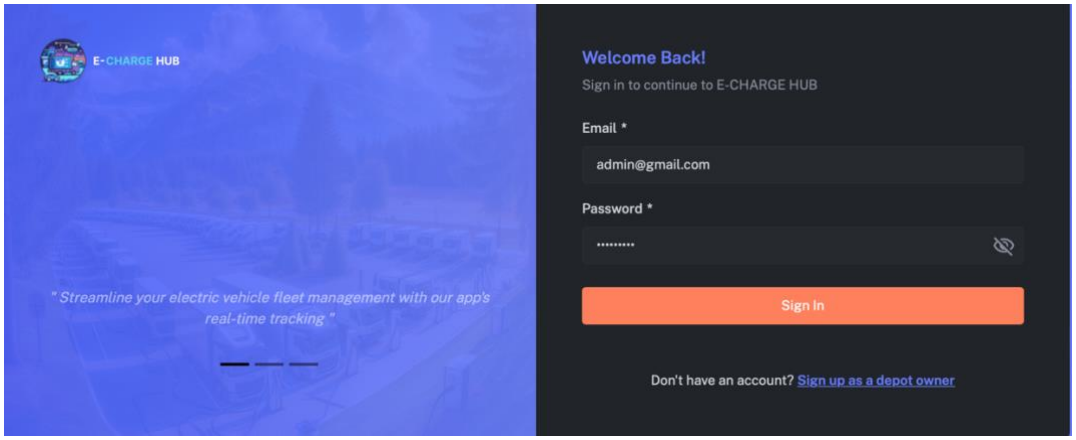


Рисунок 5.5 - Форма авторизації (рисунок виконано самостійно)

Під час тестування не було виявлено помилок при виконанні дії авторизації. Після натискання кнопки «Sign In» переходимо до основного функціоналу веб-застосунку, форма працює справно. Посилання на реєстрацію користувача активне.

Під час вводу даних у форму реєстрації, всі поля валідовані та мають обмеження. Є підказки щодо вводу формату даних.

Після натискання кнопки «Sign Up» переходимо до основного функціоналу веб-застосунку.

Із помилок було виділено - відсутність української локалізації на цьому етапі користуванням продукту.

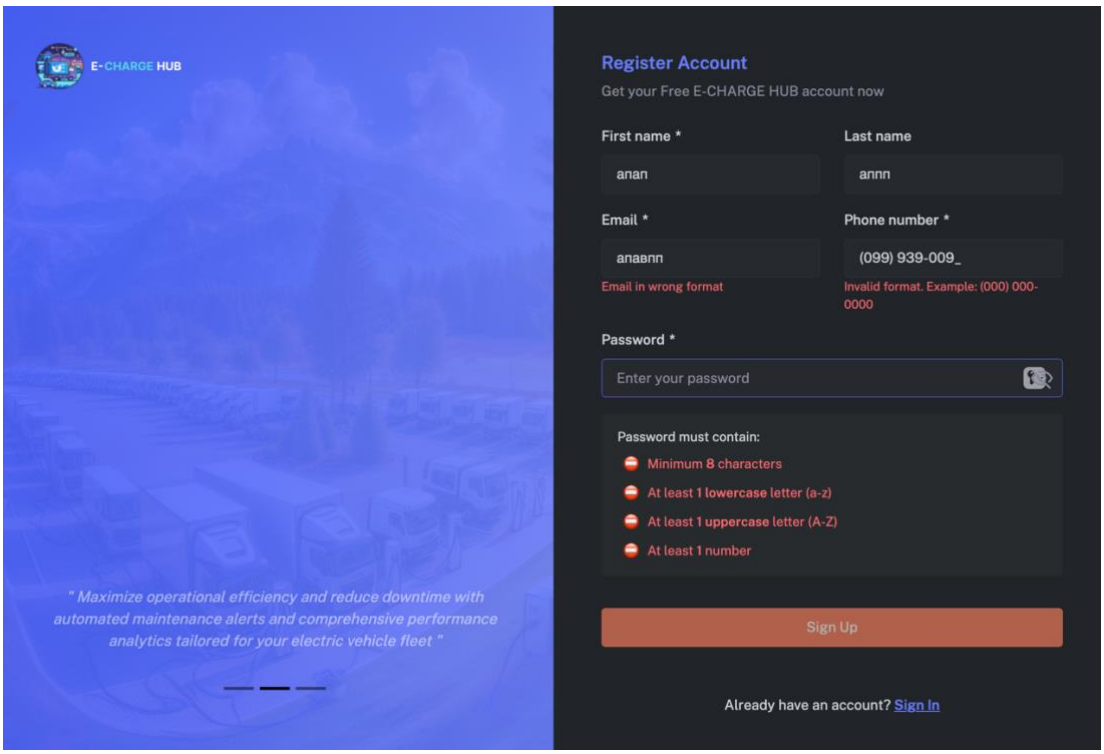


Рисунок 5.6 - Сторінка реєстрації (рисунок виконано самостійно)

Рисунки 5.7 та 5.8 містить форму створення нового депо.

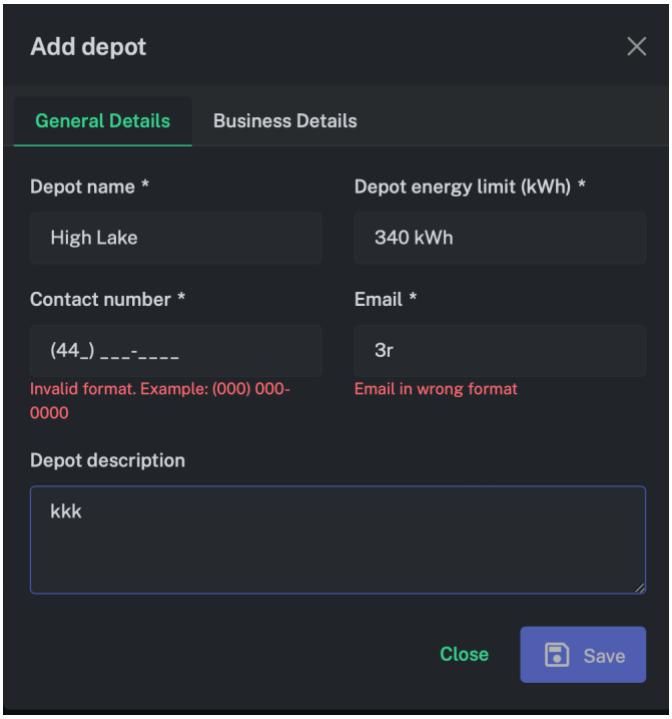


Рисунок 5.7 - Створення нового депо (рисунок виконано самостійно)

Під час тестування не було виявлено помилок при виконанні дії додавання нового депо. Після натискання кнопки «Save» переходимо до основного переліку наступної форми, форма працює справно.

Під час вводу даних у форму реєстрації, всі поля валідовані та мають обмеження. Є підказки щодо вводу формату даних.

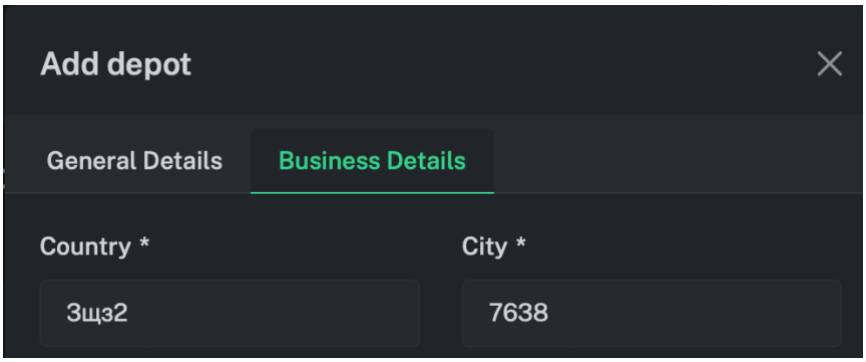


Рисунок 5.8 - Створення депо (рисунок виконано самостійно)

Під час тестування вкладки «Business Details» було виявлено помилку, про яку повідомлено команді.

Статистика, пошук та локалізація працює без помилок.

Панель інструментів для певного депо (див. рис. 5.9) справно відображає усі компоненти: помилки та зайнятість розеток, використання енергії.

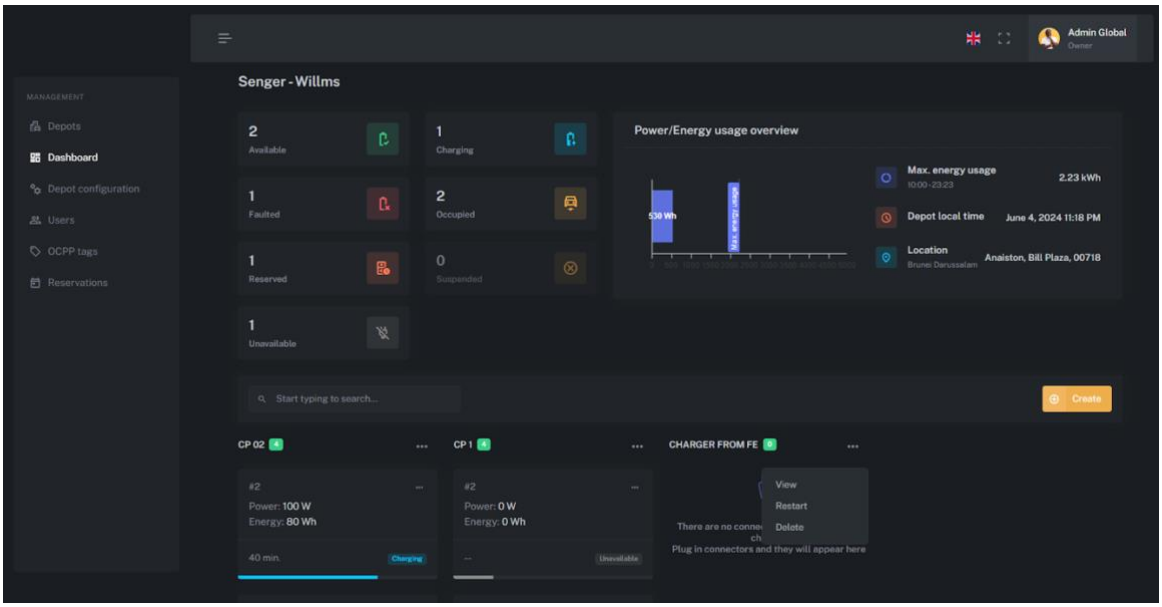


Рисунок 5.9 - Панель інструментів (рисунок виконано самостійно)

Інтерфейс створення "смарт-профілю" (рис. 5.10) у системі керування енергоспоживанням чи зарядкою пристроїв. Вкладка "Limits & Periods" в ній встановлюються обмеження та періоди для смарт-профілю. Для налаштування потрібно обрати одиницю вимірювання, мінімальну ставку. Відображаються часовий діапазон та значення для кожного часу. Кнопки створення та закриття (сказування) працюють справно.

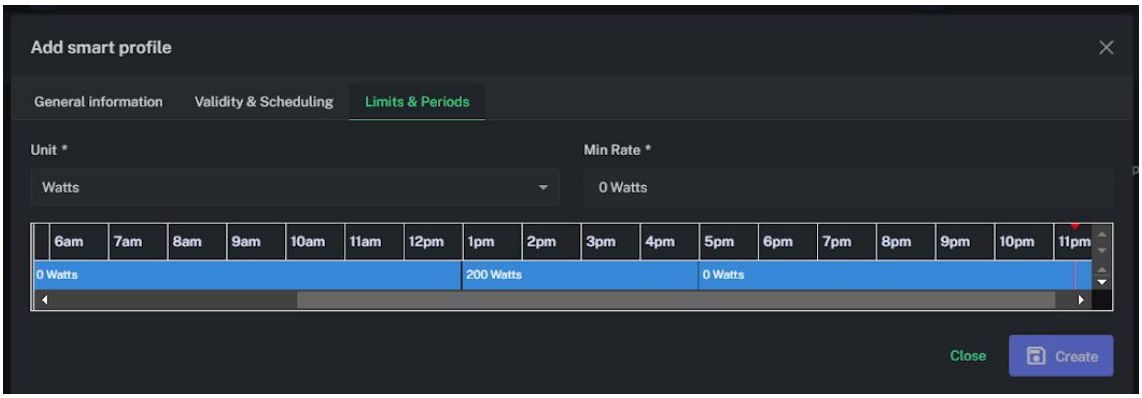


Рисунок 5.10 - Налаштування профіля для обмеження потужностей (рисунок виконано самостійно)

На сторінці резервації (рис. 5.11) відображається календар, який можна перегортати та показує актуальний місяць, тиждень, день. Працює функція перетворення календаря з табличного вигляду до списку. Можна додати (створити) резервацію, яка відображається в календарі та побачити вже створенні.

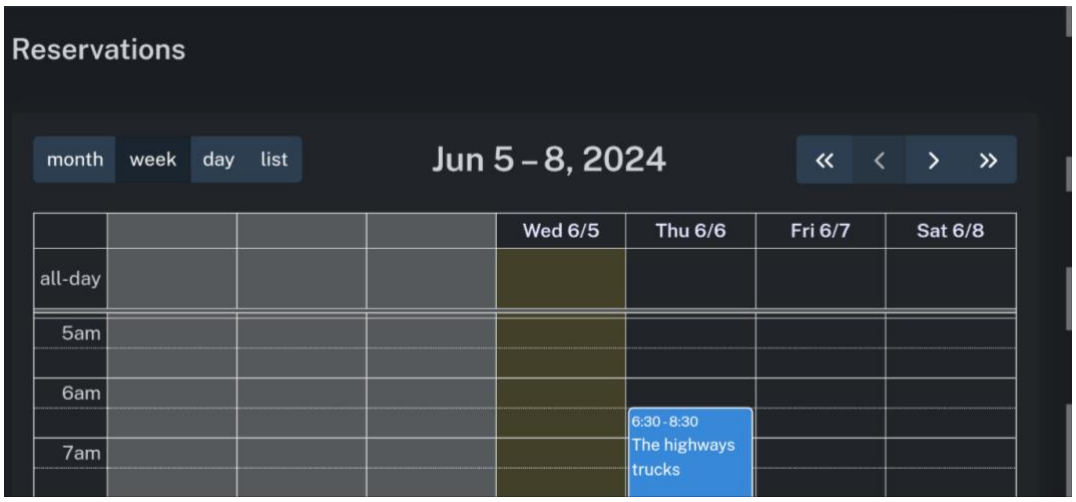


Рисунок 5.11 - Резервація у календарі (рисунок виконано самостійно)

Висновок з мануального тестування програмної системи моніторингу зарядних станцій електромобілів демонструє, що процес тестування був ефективним у виявленні різних помилок та недоліків у програмному забезпеченні. Після виявлення цих помилок вони були відправлені на допрацювання для подальшої виправлення та покращення системи.

Детальні тест-кейси містяться в додатку В.

За умови успішного виправлення знайдених помилок та підтвердження їхнього виправлення через ретестування, можна зазначити, що система стане готовою до користування. Однак, важливо продовжувати моніторити роботу системи після виправлення помилок та забезпечувати постійну підтримку та оновлення для забезпечення найвищої якості та ефективності для кінцевих користувачів.

Apache JMeter [20] - це потужний інструмент для тестування навантаження та продуктивності програмного забезпечення. Він дозволяє створювати та виконувати різноманітні тести для вимірювання продуктивності веб-додатків, служб, серверів та інших типів систем.

Основні особливості Apache JMeter:

- можливості тестування навантаженням. Apache JMeter дозволяє моделювати реальне навантаження на систему шляхом відправлення запитів та отримання відповідей. Це дозволяє визначити, як система веде себе під навантаженням та визначити її межі продуктивності.
- підтримка різноманітних протоколів. JMeter підтримує різні протоколи, включаючи HTTP, HTTPS, JDBC, FTP, SOAP, REST та інші, що робить його універсальним для тестування різноманітних систем.
- створення складних тестових сценаріїв. Інструмент дозволяє створювати складні тестові сценарії, включаючи послідовність запитів, обробку даних, виконання умовних дій та інші маніпуляції.
- моніторинг та аналіз результатів. Apache JMeter надає засоби для моніторингу та аналізу результатів тестування, включаючи графіки, звіти та статистику продуктивності системи під навантаженням.

– розширюваність через плагіни. Інструмент підтримує розширення через плагіни, що дозволяє розширювати його функціонал для виконання різних видів тестів та інтеграції з іншими інструментами.

Apache JMeter широко використовується в індустрії програмного забезпечення для проведення тестів навантаження та продуктивності, допомагаючи розробникам та QA-інженерам забезпечувати якість та ефективність різних програмних систем.

Thread Group в Apache JMeter - це елемент, який визначає конфігурацію та параметри потоків (threads) для виконання тестів (див. рис. 5.12). Thread Group дозволяє вам контролювати кількість користувачів, які взаємодіють з вашим додатком під час тестування. Основні параметри, які можна налаштовувати у Thread Group, включають:

– кількість користувачів (Number of Threads). Це параметр, який вказує скільки віртуальних користувачів або потоків буде взаємодіяти з системою під час тестування. Встановили 100 користувачів, то JMeter створить 100 потоків, які будуть виконувати запити до вашого додатку.
– час виконання (Ramp-Up Period). Цей параметр вказує, протягом якого часу всі потоки будуть запускатись. Встановили час розгортання в 5 секунд, а кількість потоків - 100, то новий потік буде запускатись кожні 100 мілісекунд протягом 10 секунд.
– кількість ітерацій (Loop Count). Цей параметр вказує, скільки разів кожен потік повторить свій сценарій дій. Встановили 100 ітерацій, то кожен потік виконає свій сценарій 100 разів перед завершенням тесту.
– розподіл навантаження (Scheduler). Цей параметр дозволяє розподілити навантаження на різні години, дні тижня або інші періоди часу, що дозволяє симулювати реальний трафік на вашому додатку.
– інші параметри. Thread Group також має інші налаштування, такі як затримка перед запуском, введення змінних (variables), налаштування моніторингу та інші.

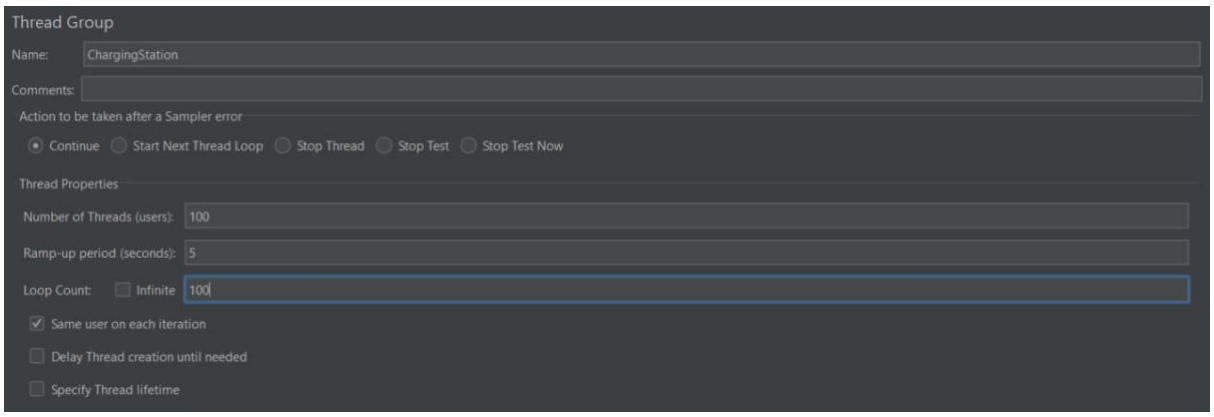


Рисунок 5.12 - Thread Group (рисунок виконано самостійно)

Отже, Thread Group в Apache JMeter - це важливий елемент для налаштування та управління потоками користувачів під час виконання тестів навантаження та продуктивності додатку.

Test Plan (план тестування) в Apache JMeter (див. рис. 5.13) - це структура, яка містить всі елементи тестування, необхідні для виконання тестів.

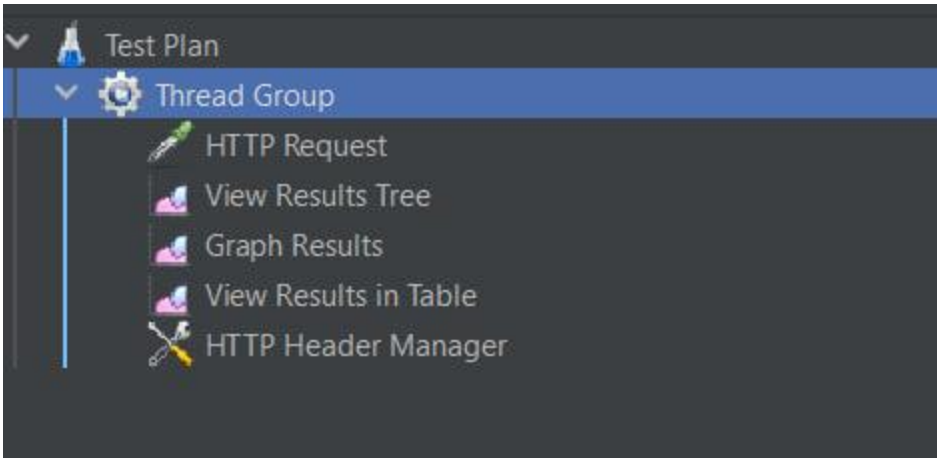


Рисунок 5.13 - Test plan (рисунок виконано самостійно)

HTTP Request (HTTP запит) - це один з основних елементів Apache JMeter, який використовується для надсилання HTTP запитів на сервер під час тестування (див. рис. 5.14). HTTP Request дозволяє виконувати різні типи HTTP запитів, такі як GET, POST, PUT, DELETE та інші, і отримувати відповіді від сервера.

Основні параметри HTTP Request включають:

– Протокол (Protocol). HTTP або HTTPS протокол, в залежності від того, чи потрібна шифрована комунікація.
– Метод (Method). Тип HTTP методу, такий як GET, POST, PUT, DELETE тощо.
– Сервер (Server Name or IP). Адреса сервера, до якого потрібно відправити запит.
– Порт (Port Number). Номер порту сервера, на який відправляється запит. Зазвичай для HTTP це 80, а для HTTPS - 443.
– Шлях (Path). Шлях до ресурсу на сервері, з яким ви хочете взаємодіяти.
– Параметри запиту (Parameters). Додаткові параметри, які включаються у запит, такі як параметри URL для GET запиту або дані форми для POST запиту.
– Тіло запиту (Body Data). Дані, які включаються у тіло POST або PUT запиту, наприклад, JSON або XML дані.
– Параметри авторизації (Authentication). Налаштування для авторизації на сервері, такі як базова авторизація або OAuth.
– Управління з'єднанням (Connection). Налаштування для управління з'єднанням з сервером, такі як Keep-Alive, Close тощо.

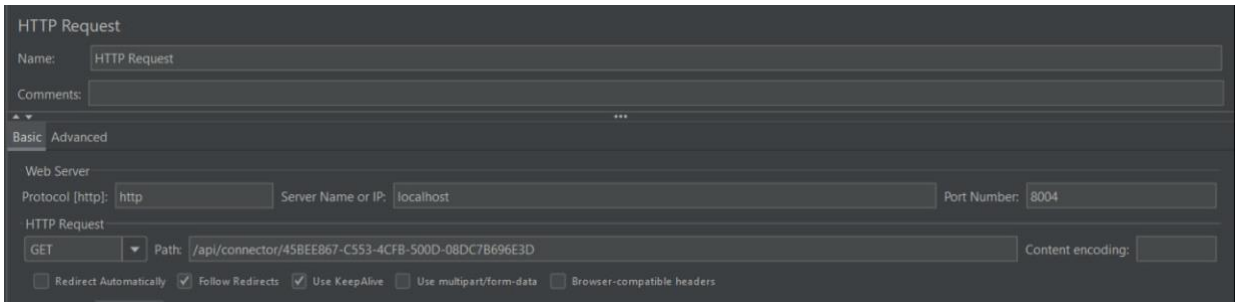


Рисунок 5.14 - HTTP Request (рисунок виконано самостійно)

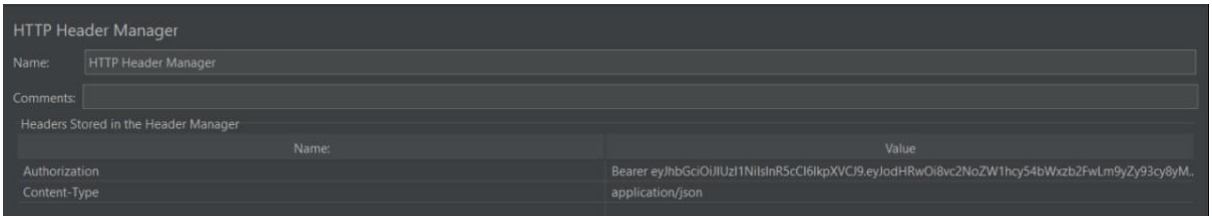
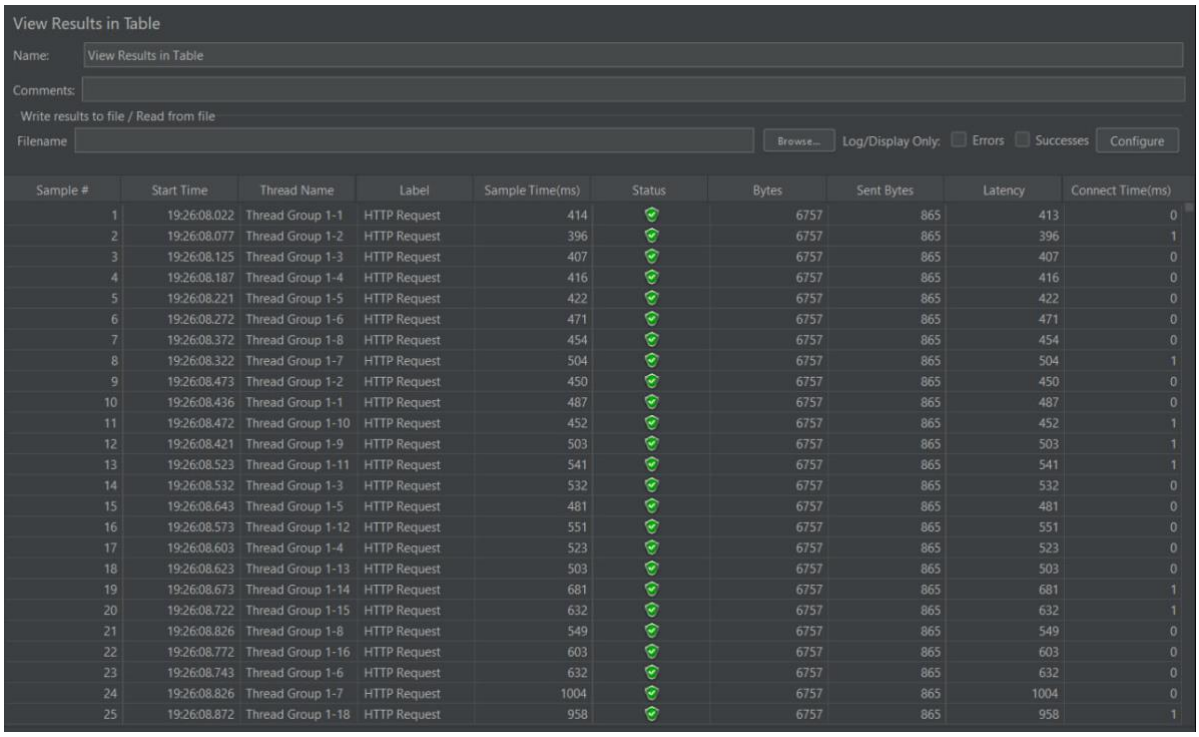


Рисунок 5.15 - Приклад заголовків запиту (рисунок виконано самостійно)

HTTP Request дозволяє вам моделювати різні типи запитів до вашого веб-сервера під час тестування навантаження та продуктивності.

"View Results in Table" (Перегляд результатів у вигляді таблиці)- це один з елементів "Listeners" (Слухачів) в Apache JMeter, який використовується для відображення результатів тестування у зручному табличному вигляді (див. рис. 5.16) . Цей елемент дозволяє аналізувати інформацію про кожний запит, який був відправлений під час тестування, включаючи час виконання, коди відповіді, розмір відповіді та інші параметри.



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	19:26:08.022	Thread Group 1-1	HTTP Request	414		6757	865	413	0
2	19:26:08.077	Thread Group 1-2	HTTP Request	396		6757	865	396	1
3	19:26:08.125	Thread Group 1-3	HTTP Request	407		6757	865	407	0
4	19:26:08.187	Thread Group 1-4	HTTP Request	416		6757	865	416	0
5	19:26:08.221	Thread Group 1-5	HTTP Request	422		6757	865	422	0
6	19:26:08.272	Thread Group 1-6	HTTP Request	471		6757	865	471	0
7	19:26:08.372	Thread Group 1-8	HTTP Request	454		6757	865	454	0
8	19:26:08.322	Thread Group 1-7	HTTP Request	504		6757	865	504	1
9	19:26:08.473	Thread Group 1-2	HTTP Request	450		6757	865	450	0
10	19:26:08.436	Thread Group 1-1	HTTP Request	487		6757	865	487	0
11	19:26:08.472	Thread Group 1-10	HTTP Request	452		6757	865	452	1
12	19:26:08.421	Thread Group 1-9	HTTP Request	503		6757	865	503	1
13	19:26:08.523	Thread Group 1-11	HTTP Request	541		6757	865	541	1
14	19:26:08.532	Thread Group 1-3	HTTP Request	532		6757	865	532	0
15	19:26:08.643	Thread Group 1-5	HTTP Request	481		6757	865	481	0
16	19:26:08.573	Thread Group 1-12	HTTP Request	551		6757	865	551	0
17	19:26:08.603	Thread Group 1-4	HTTP Request	523		6757	865	523	0
18	19:26:08.623	Thread Group 1-13	HTTP Request	503		6757	865	503	0
19	19:26:08.673	Thread Group 1-14	HTTP Request	681		6757	865	681	1
20	19:26:08.722	Thread Group 1-15	HTTP Request	632		6757	865	632	1
21	19:26:08.826	Thread Group 1-8	HTTP Request	549		6757	865	549	0
22	19:26:08.772	Thread Group 1-16	HTTP Request	603		6757	865	603	0
23	19:26:08.743	Thread Group 1-6	HTTP Request	632		6757	865	632	0
24	19:26:08.826	Thread Group 1-7	HTTP Request	1004		6757	865	1004	0
25	19:26:08.872	Thread Group 1-18	HTTP Request	958		6757	865	958	1

Рисунок 5.16 - View Results in Table (рисунок виконано самостійно)

Таблиця "View Results in Table" є корисним інструментом для аналізу результатів тестування, оцінки продуктивності сервера та виявлення будь-яких проблем чи недоліків у веб-додатку під час виконання тестів навантаження або функціонального тестування у Apache JMeter.

"Graph Results" (Графічні результати) - це один з елементів "Listeners" (слухачів) в Apache JMeter, який дозволяє візуалізувати результати тестування у

вигляді графіків (див. рис. 5.17). Цей елемент використовується для аналізу продуктивності та навантаження вашого додатку під час тестування навантаження.

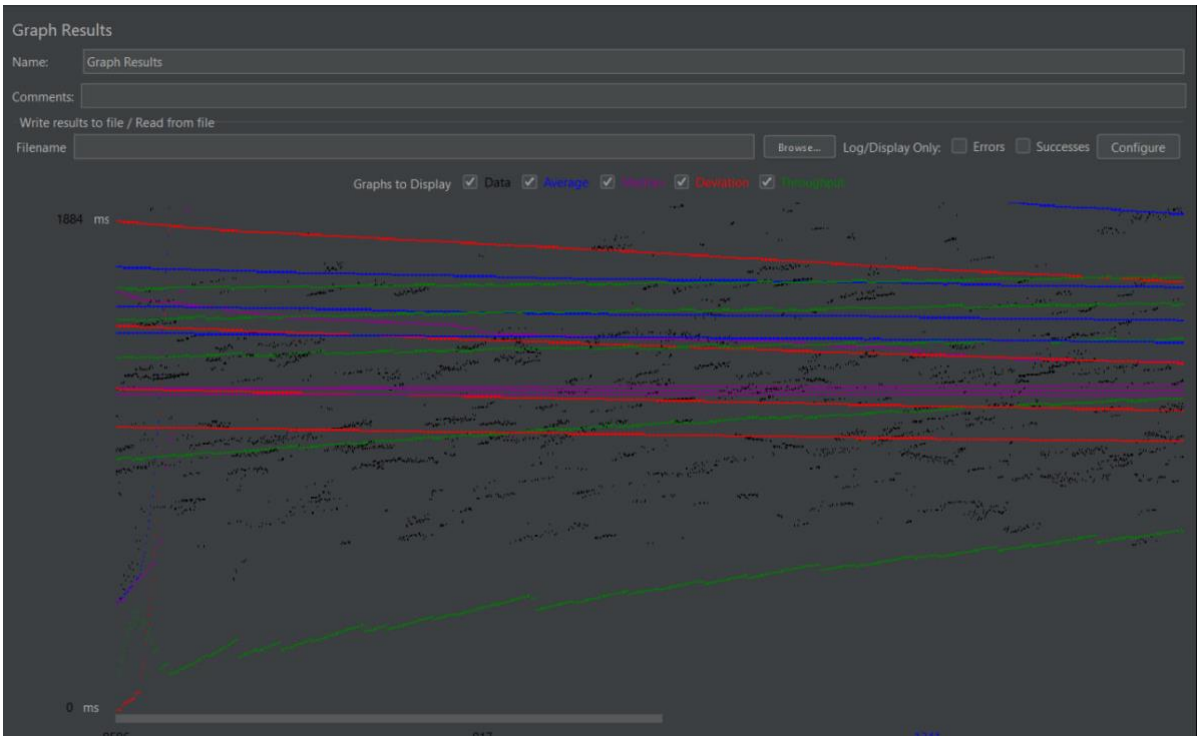


Рисунок 5.17 - Graph Results (рисунок виконано самостійно)

Графіки "Graph Results" дозволяють вам легко візуалізувати та аналізувати результати тестування навантаження в JMeter. Вони допомагають ідентифікувати піки навантаження, визначати час відгуку сервера та виявляти будь-які проблеми з продуктивністю вашого додатку під час великого навантаження.

Візуалізація даних. Графік має багато ліній різних кольорів, кожна з яких представляє певний тип даних: середнє значення (синій), медіану (фіолетовий), відхилення (червоний) та пропускну здатність (зелений).

Середнє значення (Average, синя лінія):

– лінія середнього значення показує час відповіді для різних запитів під навантаженням. Якщо лінія середнього значення залишається стабільною, це вказує на те, що система витримує навантаження стабільно. Якщо вона має тенденцію до зростання, це може вказувати на зниження продуктивності під навантаженням.

Медіана (Median, фіолетова лінія):

– лінія медіани вказує на час відповіді, який є середнім для всіх запитів. Це дає уявлення про те, як більшість запитів обробляється системою. Зростання медіани також може свідчити про погіршення продуктивності.

Відхилення (Deviation, червона лінія):

– лінія відхилення показує змінність часу відповіді. Велике відхилення означає, що деякі запити можуть займати значно більше часу, що може бути ознакою нестабільності системи під навантаженням.

Пропускна здатність (Throughput, зелена лінія):

– пропускна здатність показує кількість оброблених запитів за одиницю часу. Якщо пропускна здатність зростає стабільно, це свідчить про добру масштабованість системи.

Аналіз розташування ліній:

– лінії на графіку розташовані паралельно одна до одної з деякими перехресними точками. Це може вказувати на різні рівні навантаження в різні моменти часу.

Загальні висновки:

– якщо середні значення і медіани залишаються стабільними та низькими, це свідчить про добру продуктивність;
– зростання відхилень може сигналізувати про необхідність оптимізації системи для уникнення великих затримок у відповіді;
– стабільна або зростаюча пропускна здатність є позитивним індикатором.

"View Results Tree" (дерево результатів) - це один з елементів "Listeners" (слухачів) в Apache JMeter, який використовується для детального аналізу результатів тестування у вигляді деревовидної структури (див. рис. 5.18). Цей елемент дозволяє переглядати всі деталі щодо кожного запиту, включаючи параметри запитів, заголовки, тіло відповіді та інші дані.

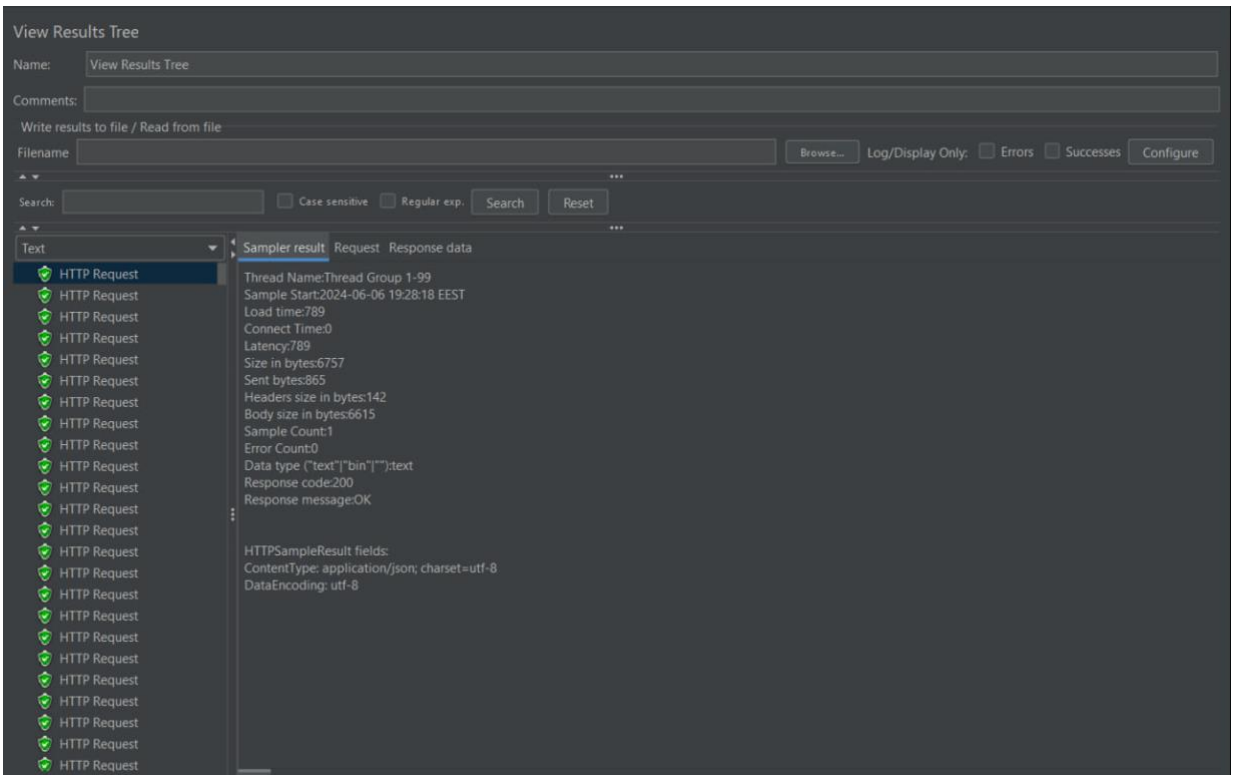


Рисунок 5.18 - View Results Tree (рисунок виконано самостійно)

"View Results Tree" дозволяє вам докладно аналізувати кожен запит, що був виконаний під час тестування, та перевіряти, чи були вони успішними, а також детально досліджувати дані відповідей сервера. Цей елемент є корисним інструментом для виявлення проблем та відладки під час тестування навантаження та функціонального тестування у Apache JMeter.

На основі наданих скріншотів JMeter, аналіз та висновки щодо результатів тестування продуктивності.

Дерево результатів (View Results Tree):

а)HTTP-запити:

1) усі показані запити успішні, що відзначено зеленими галочками.
2) кожен запит має `Код відповіді: 200` та `Повідомлення відповіді: OK`, що свідчить про успішні HTTP-відповіді.

б) Деталі для одного зразка запиту:

1) назва потоку: Thread Group 1-99
2) час початку зразка: 2024-06-06 19:28:18 EEST
3) час завантаження: 789 мс

4) час з'єднання: 0 мс

5) затримка: 789 мс

6) розмір у байтах: 6757 байт (всього), 865 байт (заголовки), 6615 байт (тіло)

7) кількість зразків: 1

8) кількість помилок: 0

9) дані відповіді: у форматі JSON з `ContentType: application/json; charset=utf-8` та `DataEncoding: utf-8`

Затримка і час завантаження для цього конкретного запиту становлять 789 мс, що вказує на відсутність додаткової затримки, окрім самої затримки.

Графічні результати (Graph Results):

а) Огляд графіка:

1) графік показує кілька ліній, що представляють різні метрики (Дані, Середнє, Медіана, Відхилення, Пропускна здатність).

2) вертикальна вісь представляє час у мілісекундах (мс), який коливається від 0 до 1884 мс.

3) горизонтальна вісь представляє кількість зразків або прогресування часу.

б) Тенденції даних:

1) середній час відповіді (синя лінія) та медіанний час відповіді (фіолетова лінія) показують стабільну тенденцію без значних піків або спадів.

2) відхилення (червона лінія) показує деяку мінливість, що свідчить про коливання часу відповіді на різні запити.

3) пропускна здатність (зелена лінія), схоже, збільшується з часом, що свідчить про зростання кількості оброблюваних запитів.

Висновки.

а) Продуктивність:

1) запити постійно успішні з `Кодом відповіді: 200` для всіх зразків.

2) час відповіді знаходиться в прийнятних межах для проаналізованих запитів (наприклад, 789 мс для одного зразка).

б) Стабільність:

1) середній та медіанний час відповіді є відносно стабільними, що свідчить про добру стабільність продуктивності.

2) деяка мінливість у відхиленні свідчить про періодичні коливання у часі відповіді, що може вимагати подальшого розслідування для забезпечення прийнятних меж.

в) Пропускна здатність:

1) збільшення пропускної здатності свідчить про те, що система здатна справлятися з зростаючим навантаженням, що є позитивним знаком для масштабованості.

Рекомендації

а) Моніторинг:

1) продовжуйте моніторинг часу відповіді та відхилень, щоб забезпечити їх дотримання прийнятних меж.

2) досліджуйте причини коливань часу відповіді для виявлення та усунення можливих проблем.

б) Навантажувальне тестування:

1) виконуйте додаткові навантажувальні тести з різним навантаженням, щоб зрозуміти, як система поводиться за різних умов стресу.

2) забезпечте, щоб система зберігала стабільність продуктивності та пропускну здатність при більш високих навантаженнях.

в) Оптимізація:

1) оптимізуйте виявлені вузькі місця, які можуть викликати коливання часу відповіді.

2) перегляньте конфігурації сервера та продуктивність додатку для подальшого покращення часу відповіді.

Ці висновки та рекомендації повинні допомогти забезпечити продуктивність та масштабованість системи.

Усі детальні результати тестування навантаженням для HTTP запитів містяться в додатку Г.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Одним із ключових аспектів розробки програмного забезпечення є публікація результатів у наукових журналах та на конференціях. Під час створення системи управління зарядними станціями для електромобілів було опубліковано кілька статей, що висвітлюють різні етапи її проектування та впровадження.

Я та Горкун Дмитро брали участь у V Міжнародній студентській науковій конференції «Розвиток суспільства та науки в умовах цифрової трансформації». У роботі було висвітлено тему проектування Angular-додатку для програмної системи моніторингу зарядних станцій електромобілів з використанням архітектури Onion. У цих тезах розглядаються ключові аспекти проектування клієнтської частини системи, зокрема використання Onion архітектури для забезпечення модульності та зручності у підтримці коду. Було детально описано структуру додатку, підходи до реалізації основних функцій та забезпечення продуктивності системи (див. додаток Д).

Також було опубліковано тези Чубарова Євгена та Рубля Дениса на конференції «Інформаційні інтелектуальні системи» XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». У цих тезах розглядаються аспекти проектування внутрішньої частини системи, зокрема новітні технології та підходи, що забезпечують надійність, масштабованість та безпеку системи.

ВИСНОВКИ

Розроблена програмна система для моніторингу зарядних станцій електромобілів є ефективним інструментом для контролю, аналізу та оптимізації їх роботи.

Попередній аналіз вимог та постановка завдань дозволяють зрозуміти, що важливими аспектами для успішної реалізації проекту є високий рівень надійності та продуктивності програмної системи, здатність взаємодіяти з різними типами користувачів та іншими системами, а також гнучкість для майбутнього розширення та оновлення.

Був створений план тестування, який включає в себе ретельний аналіз функціональності, продуктивності та навантаження системи, а також перевірку взаємодії з іншими компонентами та сторонніми сервісами.

Процес керування проектом виявився добре організованим завдяки використанню методологій та інструментів управління проектами. Формування вимог до системи було проведено з урахуванням потреб користувачів та вимог щодо надійності та ефективності функціоналу.

Мануальне тестування дозволило виявити і усунути багато дрібниць та проблем, що можуть виникнути в процесі роботи системи. Тестування навантаженням підтвердило високу стабільність та швидкодію системи при великому обсязі даних та навантаженні.

Отже, в результаті проведеного дослідження можна зробити висновок, що розроблена програмна система є ефективним та надійним інструментом для моніторингу та управління зарядними станціями електромобілів, що дозволяє підтримувати високу якість обслуговування та забезпечує зручний та безперебійний процес зарядки електротранспорту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Зарядка для електромобілів. Дія.Бізнес - Головна сторінка. URL: https://business.diia.gov.ua/idea/energetika/zaradka-dla-elektromobilej

2. Розвиток інфраструктури для електромобілів. EU4Business в Україні. URL: https://eu4business.org.ua/success-stories/developing-the-infrastructure-for-electric-cars/

3. Проблеми та рішення в розвитку інфраструктури зарядки електромобілів | BENY Нова енергія. Beny New Energy | BENY Electric. URL: https://www.beny.com/uk/challenges-and-solutions-in-electric-vehicle-charging-infrastructure-development/.

4. Ринок зарядних станцій для електромобілів: тенденції, прогнози та факти. Розумне рішення для зарядних станцій | GO TO-U. URL: https://go-tou.com/ua/news/electric-vehicle-charging-stations-market-trends-forecast-and-facts.

5. Charge Management Software - bp pulse. *bp pulse fleet*. URL: https://bppulsefleet.com/fleet/products/charge-management-software/

6. JET Charge - Australia's Leading Experts in EV Charging Stations & Chargers. JET Charge. URL: https://jetcharge.com.au

7. ev.energy - Smart EV Charging App. ev.energy - Smart EV Charging App. URL: https://www.ev.energy

8. Скрам. Що це таке та як цим користуватися. Створення і розробка інтернет-магазинів від Brander. URL: https://brander.ua/blog/skram-shcho-tse-take-ta-yak-tsym-korystuvatysya.

9. Синкевич М. Е., Лєсна Н. С. ДОСЛІДЖЕННЯ ЗАСОБІВ І ТЕХНОЛОГІЙ МІЖСЕРВІСНОГО ЗВ'ЯЗКУ В МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ. International Electronic Scientific Journal "Science Online" http://nauka-online.com/. 2019. С. 1–2. URL:https://nauka-online.com/publications/informatsionnye-tehnologii/2019/5/doslidzhennya-zasobiv-i-tehnologij-mizhservisnogo-zv-yazku-v-mikroservisnij-arhitekturi/.

10. Скрам. Що це таке та як цим користуватися. Створення і розробка інтернет-магазинів від Brander. URL: https://brander.ua/blog/skram-shcho-tse-take-ta-yak-tsym-korystuvatysya .

11. Методологія Scrum – основні принципи. Apix-Drive. URL: https://apix-drive.com/ua/blog/useful/metod-scrum

12. Singureanu C. Ручне тестування - типи, процес, інструменти та інше!. ZAPTEST. URL: https://www.zaptest.com/uk/ручне-тестування-що-це-таке-типи-проц

13. Що таке тест-кейс? Приклади створення тест-кейсів. FoxmindEd. URL: https://foxminded.ua/test-keis/

14. Що таке тест-план, для чого він потрібен і з чого складається | Онлайн-курси від компанії QATestLab. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/test-plan/

15. White/black/grey box-тестування - QALight. QALight. URL: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/

16. Тестування продуктивності - QALight. QALight. URL: https://qalight.ua/baza-znaniy/testuvannya-produktivnosti/

17. Що таке PEST-аналіз. *Дія.Бізнес - Головна сторінка*. URL: https://business.diia.gov.ua/handbook/marketing/so-take-pest-analiz

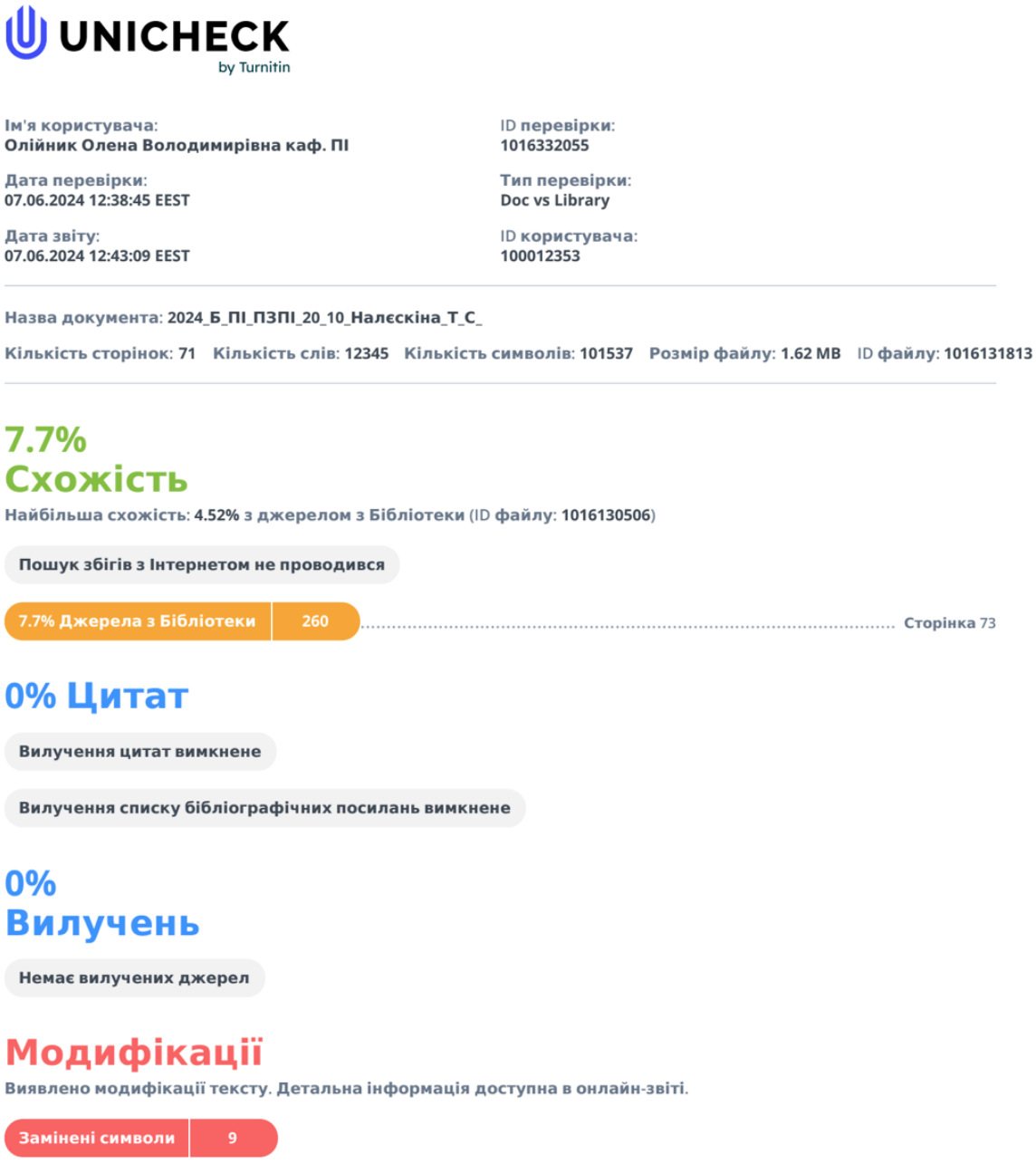
18. Що таке SWOT аналіз?. *Дія.Бізнес - Головна сторінка*. URL: https://business.diia.gov.ua/handbook/marketing/so-take-swot-analiz

19. Підручник з Browserstack: Платформа для тестування додатків та браузерів (ПОСІБНИК) - Інший. *Огляди, Ігри, Розваги, Червень 2024*. URL: https://uk.myservername.com/browserstack-tutorial.

20. JMeter в тестуванні. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/using-jmeter-in-testing/

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016332055

Дата перевірки:
07.06.2024 12:38:45 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
07.06.2024 12:43:09 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_10_Налєскіна_Т_С_

Кількість сторінок: 71 Кількість слів: 12345 Кількість символів: 101537 Розмір файлу: 1.62 MB ID файлу: 1016131813

7.7%
Схожість

Найбільша схожість: 4.52% з джерелом з Бібліотеки (ID файлу: 1016130506)

Пошук збігів з Інтернетом не проводився

7.7% Джерела з Бібліотеки 260 ..... Сторінка 73

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.

Замінені символи 9

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроніки
Кафедра ПІ
Кваліфікаційна робота бакалавра

ПРОГРАМНА СИСТЕМА МОНІТОРИНГУ ЗАРЯДНИХ СТАНЦІЙ ЕЛЕКТРОМОБІЛІВ. КЕРУВАННЯ ПРОЄКТОМ, ФОРМУВАННЯ ВИМОГ, МАНУАЛЬНИЙ ТЕСТУВАЛЬНИК ТА ТЕСТУВАННЯ НАВАНТАЖЕННЯМ.

Виконала ст.гр. ПЗПІ-20-10 Налєскіна Т.С.
Керівник: доц. каф. ПІ Русакова Н.Є.

Мета роботи

АНАЛІЗ предметної галузі та ринку програмних продуктів

ФОРМУВАННЯ ВИМОГ до розроблювального програмного забезпечення

МЕНЕДЖМЕНТ проєкту під час всього часу його розробки

Тестування готового проєкту для виявлення помилок. Мануальне та навантаженням

2

Існуючі аналоги

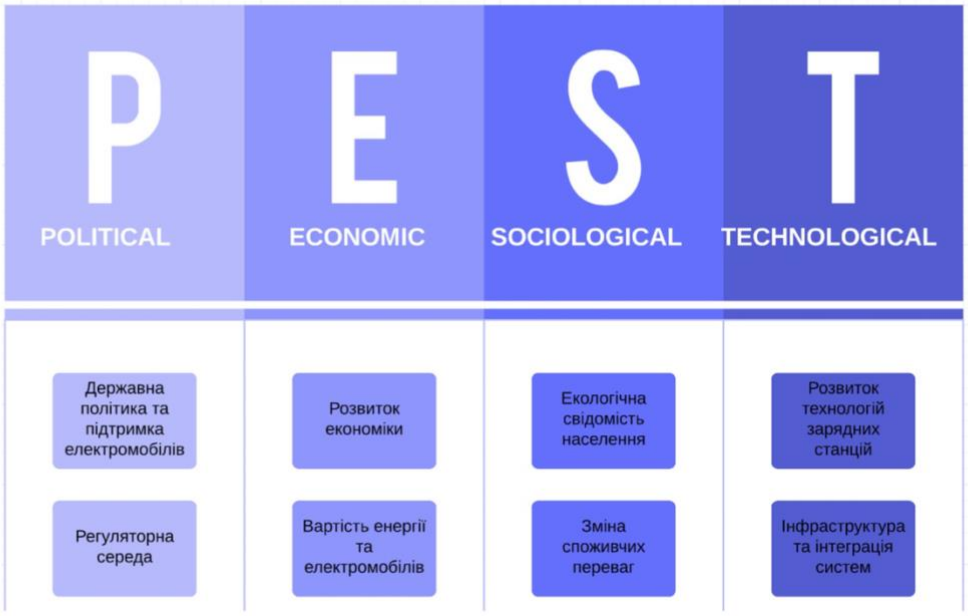
bp pulse
Сервіс для зарядки електричних транспортних засобів від компанії BP

Ev.energy
Платформа для керування зарядкою електричних транспортних засобів

ChargeLab
ПЗ для управління зарядними станціями електричних транспортних засобів

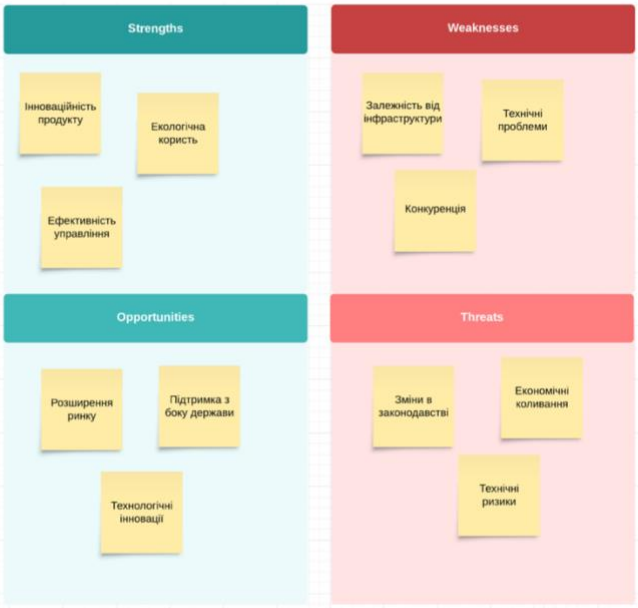
3

PEST аналіз



4

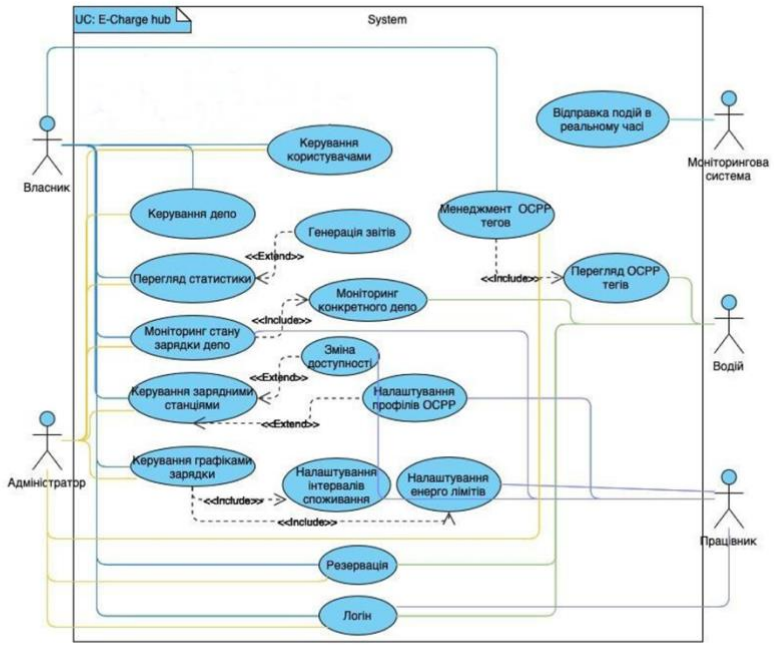
SWOT аналіз



5

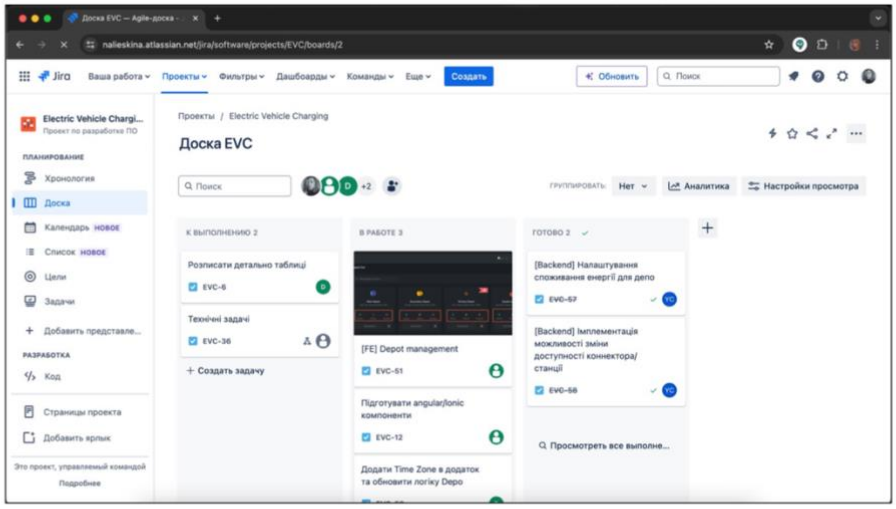
Розробка MVP

Діаграма прецедентів



6

Управління проєктом в Jira



7

Використанні техніки SCRUM

Sprint Planning
Планування спринту в Google Meet

Sprint Review
Обговорення результатів спринту в Google Meet

SCRUM Board
Використання дошки у Jira

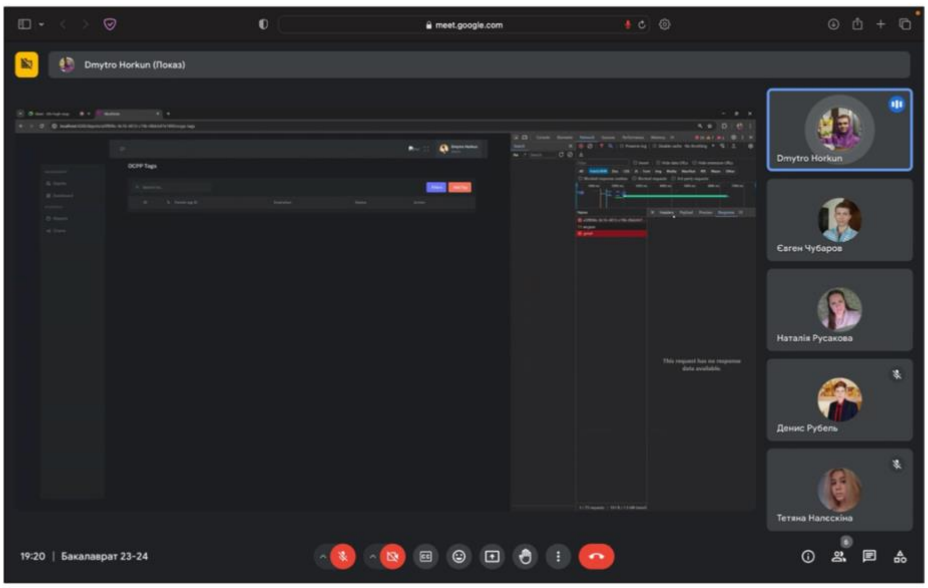
Daily Scrum
Щоденний звіт у Telegram / Discord

Sprint Retrospective
Оцінка підсумків спринту та робочого процесу, висновки у Google Meet

User Stories
Документ з короткими описами функціональності, написаними з перспективи користувача.

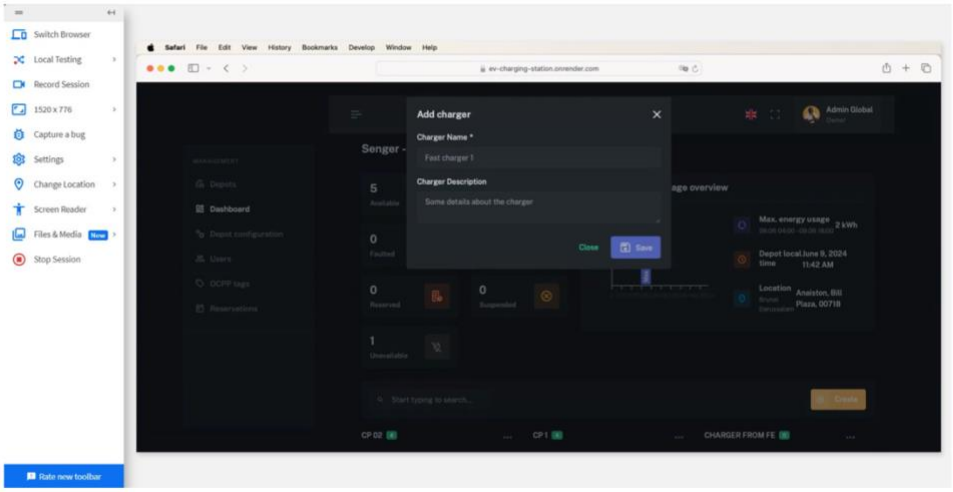
8

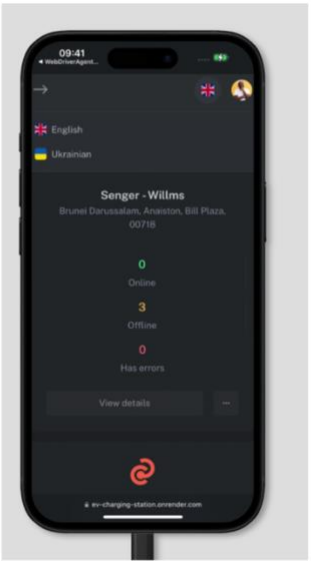
Проведення зустрічей у Google Meet



9

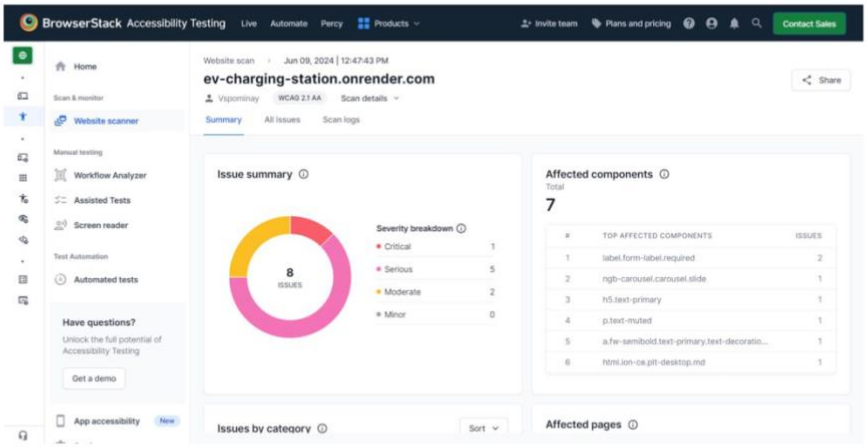
Мануальне тестування





10

Мануальне тестування



11

Тестування навантаженням

Test and Report information

Source file	"AggregatorDepotById.jtl"
Start Time	"6/6/24, 1:41 AM"
End Time	"6/6/24, 1:44 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.295	500 ms	1 sec 500 ms	Total
0.295	500 ms	1 sec 500 ms	HTTP Request

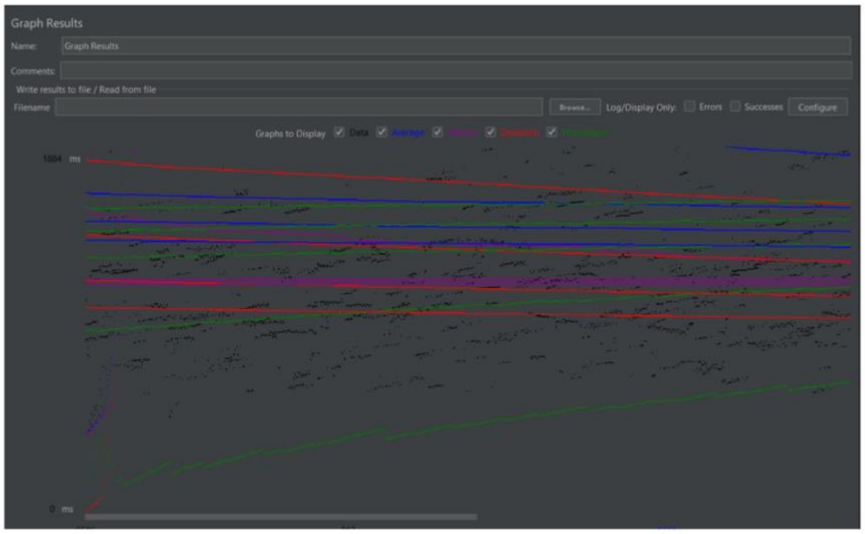
Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	1490.33	659	5858	1378.00	1945.00	2368.00	3287.94	64.89	37.51	56.97
HTTP Request	10000	0	0.00%	1490.33	659	5858	1378.00	1945.00	2368.00	3287.94	64.89	37.51	56.97

12

Тестування навантаженням



13

Апробація

Проектування Angular додатку для програмної системи моніторингу зарядних станцій електромобілей використовуючи Onion архітектуру.

V Міжнародна студентська наукова конференція «Розвиток суспільства та науки в умовах цифрової трансформації».

14

Висновки

- Проведено аналіз предметної галузі та конкурентів на ринку.
- Сформовано вимоги до програмної системи.
- Проведено керування проєктом за допомогою методики SCRUM та середовищем управління проєктами – Jira.
- Проведено мануальне тестування та тестування навантаженням.
- Розроблено діаграми та документи, що передбачаються вимогам програмної системи.

15

Дякую за увагу!

16

ДОДАТОК В

Тест-кейси для мануального тестування

Таблиця В.1 - Тест-кейс №1 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Тестування авторизації користувача на сайті за допомогою введення email та паролю.		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності процесу авторизації та доступу до облікового запису користувача.		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	При успішному виконанні тесту, система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
Авторизація користувача на сайті			
№	Опис випадку	Очікуваний результат	Висновок
1	Введіть дійсний email користувача у поле "Email"	Коректно відображається введений email, обов’язкове поле заповнено	Пройдено
2	Введіть вірний пароль користувача у поле "Пароль"	Пароль відображається скритим, обов’язкове поле заповено	Пройдено
3	Натисніть кнопку "Увійти"	Перенапралено на інші сторінки веб-застосунку, користувача коректно авторизовано	Пройдено
4	Натиснути клавішу «Enter»	Перенапралено на інші сторінки веб-застосунку, користувача коректно авторизовано	Пройдено
5	Вийдіть з облікового запису та перевірте, що користувач виходить зі свого облікового запису без будь-яких проблем.	При виході з облікового запису користувач повертається на відповідну сторінку інтерфейсу сайту без помилок.	Пройдено

Кінець таблиці В.1

6	Заповнення полей некоретною інформацією	Відображається повідомлення про некоректне заповнення полей	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця В.2 - Тест-кейс №2 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №2		
Опис функції:	Пошук користувача за ім'ям чи електронною поштою		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності функціонування функції пошуку користувачів та здатність системи знаходити відповідні результати.		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
Пошук користувача			
№	Опис випадку	Очікуваний результат	Висновок
1	Перейти на сторінку з пошуком користувачів.	Перенаправлено на сторінку	Пройдено
2	Введіть ім'я користувача, якого потрібно знайти, у поле пошуку за ім'ям	Поле коректно заповнено	Пройдено
3	Натисніть кнопку "Пошук"	Система знаходить користувачів, введених за ім'ям чи електронною поштою, якщо вони існують у базі даних.	Пройдено
4	Натиснути клавішу «Enter»	Система знаходить користувачів, введених за ім'ям якщо вони існують у базі даних.	Пройдено

Кінець таблиці В.2

5	Перевірте, чи з'являються в результаті пошуку користувачі з введеним ім'ям	Результати пошуку відображаються коректно і містять відповідну інформацію про користувачів. Якщо користувача з введеним ім'ям чи електронною поштою не знайдено, система показує відповідне повідомлення про відсутність результатів.	Пройдено
6	Повторіть ті ж самі кроки для пошуку за електронною поштою.	Результати пошуку відображаються коректно і містять відповідну інформацію про користувачів. Якщо користувача з введеним ім'ям чи електронною поштою не знайдено, система показує відповідне повідомлення про відсутність результатів.	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця В.3 - Тест-кейс №3 (таблиця зроблена самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №3		
Опис функції:	Додавання нового депо з вказаними параметрами		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності додавання нового депо з усіма обов'язковими параметрами.		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено

Кінець таблиці В.3

3	Користувач повинен мати доступ до функції додавання нового депо.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
Додавання нового депо			
№	Опис випадку	Очікуваний результат	Висновок
1	Перейдіть до розділу додавання нового депо.	Перенаправлено на сторінку	Пройдено
2	Введіть ім'я депо у відповідне поле.	Поле коректно заповнено	Пройдено
3	Введіть енергетичний ліміт у відповідне поле (додайте значення, яке відповідає формату).	Поле коректно заповнено	Пройдено
4	Введіть контактний номер у відповідне поле (додайте вірний номер телефону).	Поле коректно заповнено	Пройдено
5	Введіть email у відповідне поле (додайте вірну email-адресу).	Поле коректно заповнено	Пройдено
6	Додайте опис депо у відповідне поле	Поле коректно заповнено	Пройдено
7	Натисніть кнопку "Додати депо"	Система додає нове депо без помилок, якщо всі обов'язкові поля заповнені вірно.	Пройдено
8	Перевірте, чи нове депо додається без помилок та з введеними даними.	Нове депо з'являється у списку депо з введеними даними (ім'я, енергетичний ліміт, контактний номер, email, опис).	Пройдено
9	Введення у поля некоректні формати даних	Якщо введені дані не відповідають вимогам формату або поля залишені порожніми, система повинна показати відповідне повідомлення про помилку.	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця В.4 - Тест-кейс №4 (таблиця зроблена самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №4		
Опис функції:	Створення резервації зарядної станції		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності додавання резервації на календарь		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
3	У користувача повинно відображатись вкладка «Резервація» в боковому полі інструментів.	Користувач має роль, яка дійсно передбачає змогу до резервації та перегляду календаря	Пройдено
Створення нової резервації			
№	Опис випадку	Очікуваний результат	Висновок
1	Натисніть двічі ПКМ по бажаному вікну для того, щоб створити нову резервацію	Перенаправлено на сторінку	Пройдено
2	Введіть ім'я резервації у відповідне поле.	Поле коректно заповнено	Пройдено
3	Введіть короткий опис у відповідне поле. Це поле є не обов’язковим	Поле коректно заповнено. Або залишено пустим	Пройдено
4	Оберіть з випадаючого списку OCPP тег	Тег обрано	Пройдено
5	Оберіть з випадаючого списку розетку	Розетку обрано	Пройдено
6	Оберіть кількість конекторів або введіть вручну	Кількість вказано	Пройдено
7	Вкажіть час початку резервації в випадаючому полі. Залишити можливість вказувати час самостійно без стрілочок.	Час обрано. Самостійно вказувати не можливо.	Пройдено частково

Кінець таблиці В.4

8	Вкажіть час закінчення резервації в випадаючому полі. Залишити можливість вказувати час самостійно без стрілочок.	Час обрано. Самостійно вказувати не можливо	Пройдено частково
9	Натиснути кнопку «Створити» для того, щоб зберегти створення резервації	Резервація створена та відображається на календарі.	Пройдено
10	Натиснути кнопку «Закрити» для того, щоб не створити створення резервації	Резервація не створена	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця В.5 - Тест-кейс №5 (таблиця зроблена самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №5		
Опис функції:	Перегляд календаря		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності відображення календаря з резервацій		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
3	У користувача повинно відображатись вкладка «Резервація» в боковому полі інструментів.	Користувач має роль, яка дійсно передбачає змогу до резервації та перегляду календаря	Пройдено
Відображення календаря для резервацій			
№	Опис випадку	Очікуваний результат	Висновок
1	За стандартом відображається календарний тиждень поточного дня.	Відображення поточного тижня.	Пройдено

Кінець таблиці В.5

2	За допомогою кнопок навігації (стрілочки) можемо перегортати календар по тижням або місяцям.	Календар перегортається	Пройдено
3	При натисканні кнопки «Місяць» календар відображає поточний місяць з усіма резерваціями	Календар відображає поточний місяць з виділенням сьогоднішньої дати	Пройдено
4	При натисканні кнопки «Тиждень» календар відображає поточний тиждень з усіма резерваціями	Календар відображає поточний тиждень з виділенням сьогоднішньої дати	Пройдено
5	При натисканні кнопки «День» календар відображає поточний день з усіма резерваціями	Календар відображає поточний день з погодинним списком	Пройдено
6	При натисканні кнопки «Список» відображаються резервації поточного місяця.	Календар відображає список з усіма резерваціями поточного місяця.	Пройдено
7	При натисканні на резервацію ми можемо побачити детальну інформацію у відкритому спливаючому вікні.	Інформація відображається	Пройдено
8	У відкритому вікні резервації ми можемо вносити правки (відкриття вікна редагування).	Можемо редактувати. Вікно редагування відкривається.	Пройдено
9	У відкритому вікні резервації ми можемо видалити резервацію. Перед видаленням бачимо спливаюче вікно-попередження	Можемо видалити. Є вікно-попередження	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця В.6 - Тест-кейс №6 (таблиця зроблена самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №6		
Опис функції:	Запрошення користувача		
Власник тесту:	Налєскіна Тетяна Сергіївна		
Дата створення:	10.05.2024		
Мета тесту:	Перевірка правильності запрошення користувача		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Запустити браузер та перейти на сторінку авторизації.	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач повинен бути зареєстрований в системі та мати дійсний обліковий запис.	Система дозволяє йому отримати доступ до внутрішніх функцій або послуг.	Пройдено
3	Користувач має бути рівнем доступу «супер адмін» або «адмін»	Користувач має роль, яка дійсно передбачає змогу до запрошення	Пройдено
Відображення календаря для резервацій			
№	Опис випадку	Очікуваний результат	Висновок
1	Переходимо за допомогою бокової панелі інструментів до вкладки «Користувачі»	Відображення вкладки	Пройдено
2	Бачимо таблицю існуючих користувачів, по якій можемо проходитись пошуком	Відображення таблиці	Пройдено
3	Можемо редагувати користувачів у таблиці, для переходу до редагування натискаємо на іконку «олівець»	Можливість корегування	Пройдено
4	Можемо видалити користувача зі системи, перед цим бачимо вікно-сповіщення	Користувача видалено. Вікно сповіщення є	Пройдено
5	При натисканні кнопки «Запросити» відкривається вікно створення користувача	Вікно відкрилось.	Пройдено
6	Обов’язковий ввод пошти нового користувача. Дані потрібні валідовані.	Дані коректно відображається	Пройдено

Кінець таблиці В.6

7	Обов’язкове поле вибору ролі майбутнього користувача за допомогою випадаючого списка.	Дані відображаються коректно.	Пройдено
8	Поле вводу імені користувача. Дані потрібні валідовані. Не обов’язкове поле.	Дані відображаються коректно. При відсутності даних, поле залишилось порожнім.	Пройдено
9	Поле вводу призвища. Дані потрібні валідовані. Не обов’язкове поле.	Дані відображаються коректно. При відсутності даних, поле залишилось порожнім.	Пройдено
10	При натисканні кнопки «Запросити» спливає вікно-сповіщення, та на пошту майбутнього користувача приходить запрошення з посиланням для завершення реєстрації та зміни пароля.	Запрошення успішно сформоване, лист прийшов на пошту.	Пройдено
11	Після переходу по посиланню відкривається сторінка з вже введеними даними та зміна паролю	Посилання працює, сторінка відкрилась	Пройдено
12	Дані у формах можна змінити	Дані редагуються, якщо інформація введена коректна	Пройдено
13	Користувач відображається в списку користувачів	Користувач відображається	Пройдено
14	На вкладці створення запрошення при натисканні кнопки «Закрити» запрошення не стоврене	Запрошення не створене. Перехід до сторінки з користувачами	Пройдено
15	При умові не коректного заповнення полей (відсутність заповненості обов’язкових / не коректні дані) запрошення не створюється.	Запрошення не створюється. Біля полей текст попередження та рекомендації.	Пройдено
Результати тестування			
Тестувальник: Налєскіна Т.С.	Дата прогону тесту: 10.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

ДОДАТОК Г

Звіти з тестування навантаження HTTP методів

AggregatorDepotById

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.295	**500 ms**	**1 sec 500 ms**	**Total**
0.295	500 ms	1 sec 500 ms	HTTP Request

Рисунок Г.1 - Apdex запиту (рисунок виконано самостійно)

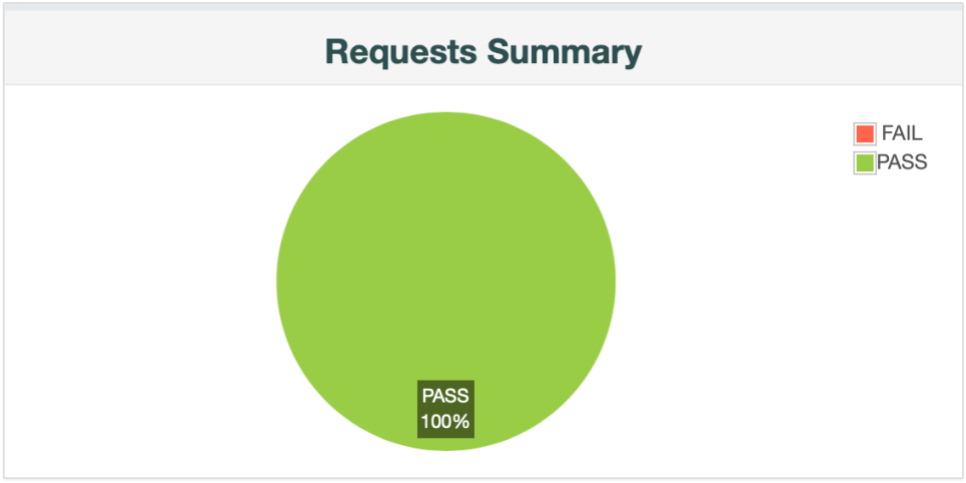
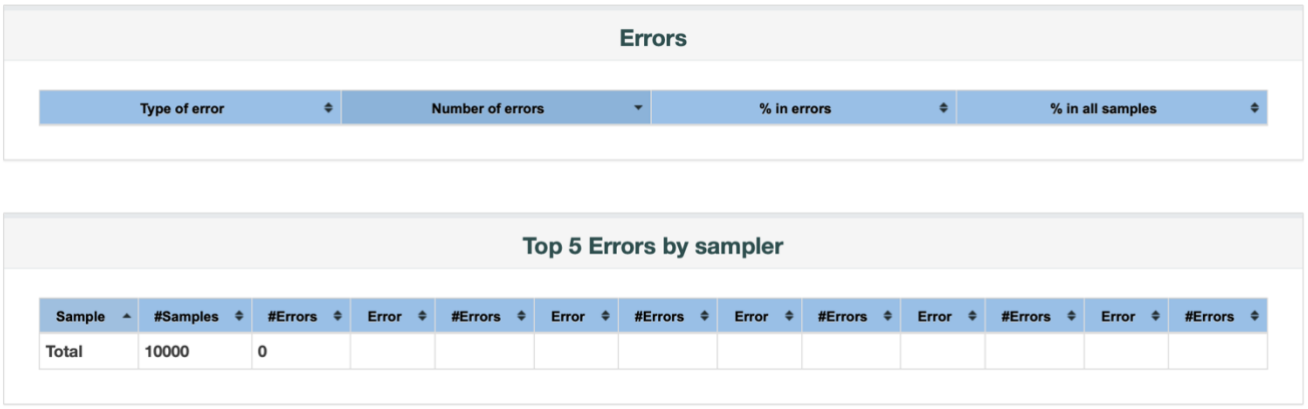


Рисунок Г.2 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	1490.33	659	5858	1378.00	1945.00	2368.00	3287.94	64.89	37.51	56.97
HTTP Request	10000	0	0.00%	1490.33	659	5858	1378.00	1945.00	2368.00	3287.94	64.89	37.51	56.97

Рисунок Г.3 - Статистика запиту (рисунок виконано самостійно)



Errors

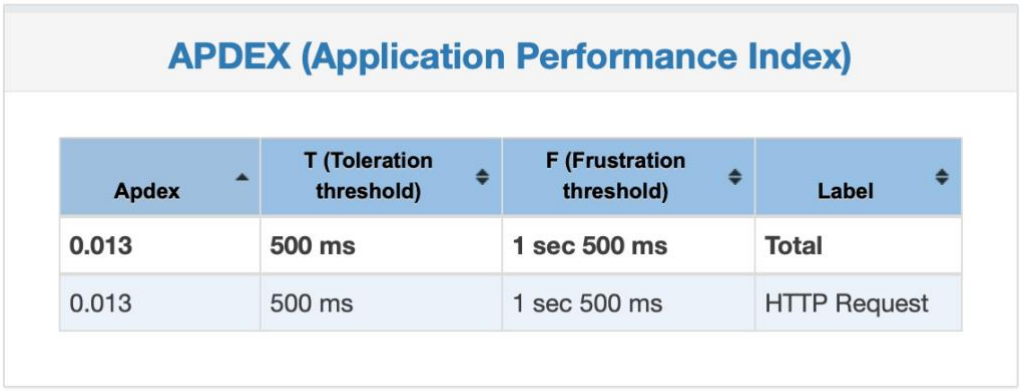
Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	10000	0										

Рисунок Г.4 - Помилки запиту (рисунок виконано самостійно)

AggregatorDepotGetall



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.013	500 ms	1 sec 500 ms	Total
0.013	500 ms	1 sec 500 ms	HTTP Request

Рисунок Г.5 - Apdex запиту (рисунок виконано самостійно)

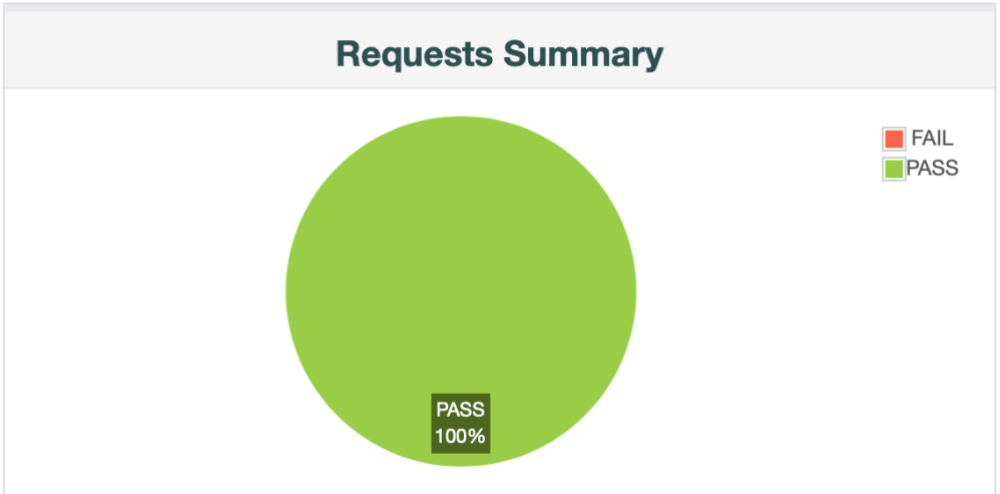


Рисунок Г.6 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	2752.18	689	15485	2220.50	4028.00	5935.00	11465.59	35.71	164.40	30.34
HTTP Request	10000	0	0.00%	2752.18	689	15485	2220.50	4028.00	5935.00	11465.59	35.71	164.40	30.34

Рисунок Г.7 - Статистика запиту (рисунок виконано самостійно)

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	10000	0										

Рисунок Г.8 - Помилки запиту (рисунок виконано самостійно)

ChargepointById

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.160	500 ms	1 sec 500 ms	Total
0.160	500 ms	1 sec 500 ms	HTTP Request

Рисунок Г.9 - Apdex запиту (рисунок виконано самостійно)

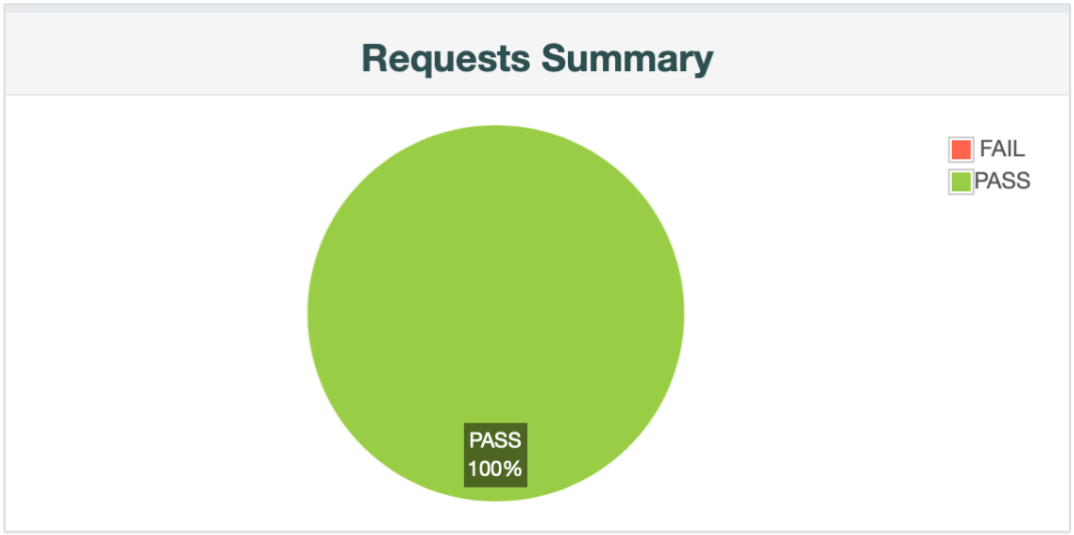


Рисунок Г.10 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	1893.50	322	5512	1797.50	2807.90	3366.00	4482.00	51.27	53.43	44.77
HTTP Request	10000	0	0.00%	1893.50	322	5512	1797.50	2807.90	3366.00	4482.00	51.27	53.43	44.77

Рисунок Г.11 - Статистика запиту (рисунок виконано самостійно)

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	10000	0										

Рисунок Г.12 - Помилки запиту (рисунок виконано самостійно)

ChargingProfileGetall

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.413	**500 ms**	**1 sec 500 ms**	**Total**
0.413	500 ms	1 sec 500 ms	HTTP Request

Рисунок Г.13 - Apdex запиту (рисунок виконано самостійно)

Requests Summary

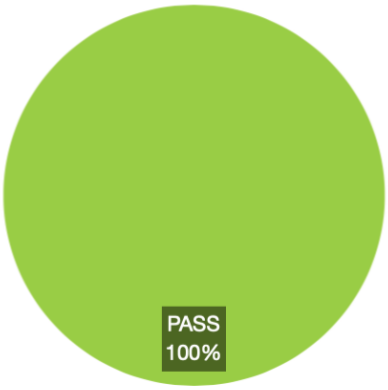


Рисунок Г.14 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**10000**	**0**	**0.00%**	**1390.51**	**414**	**17518**	**1152.00**	**1794.90**	**2250.95**	**12859.41**	**69.85**	**241.26**	**59.27**
HTTP Request	10000	0	0.00%	1390.51	414	17518	1152.00	1794.90	2250.95	12859.41	69.85	241.26	59.27

Рисунок Г.15 - Статистика запиту (рисунок виконано самостійно)

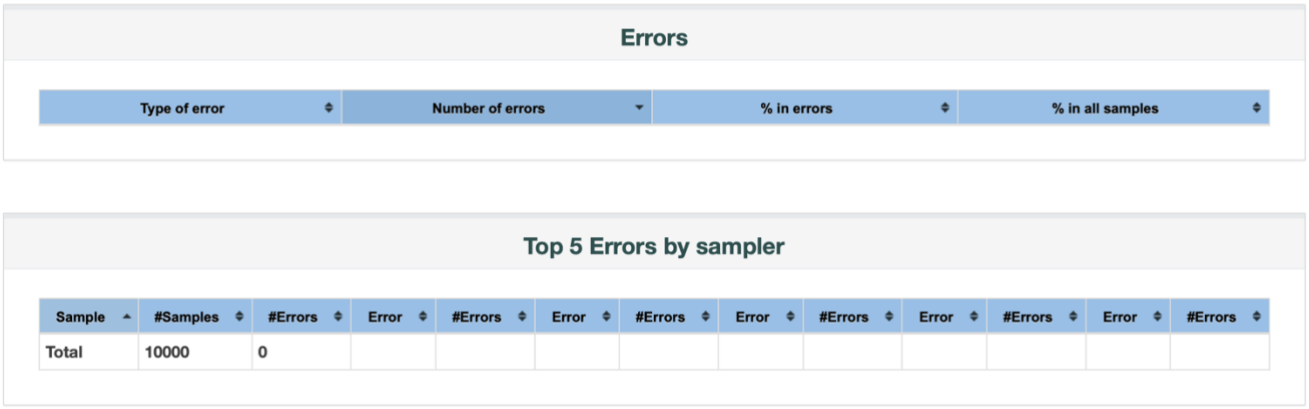


Рисунок Г.16 - Помилки запиту (рисунок виконано самостійно)

UserById

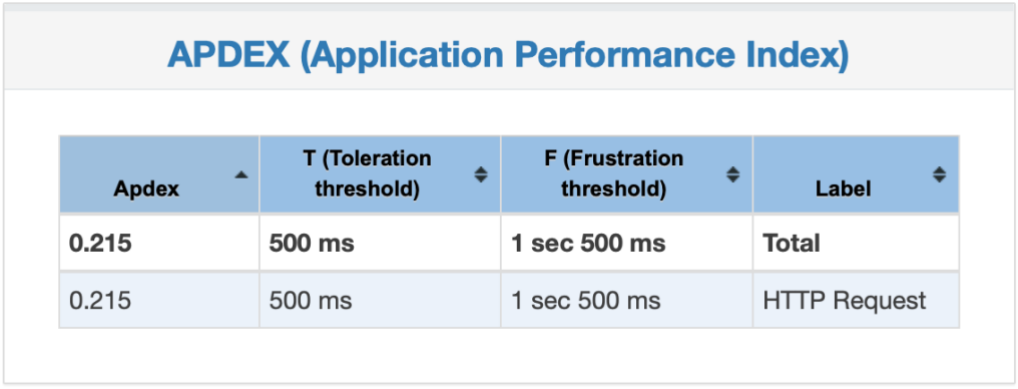


Рисунок Г.17 - Apdex запиту (рисунок виконано самостійно)

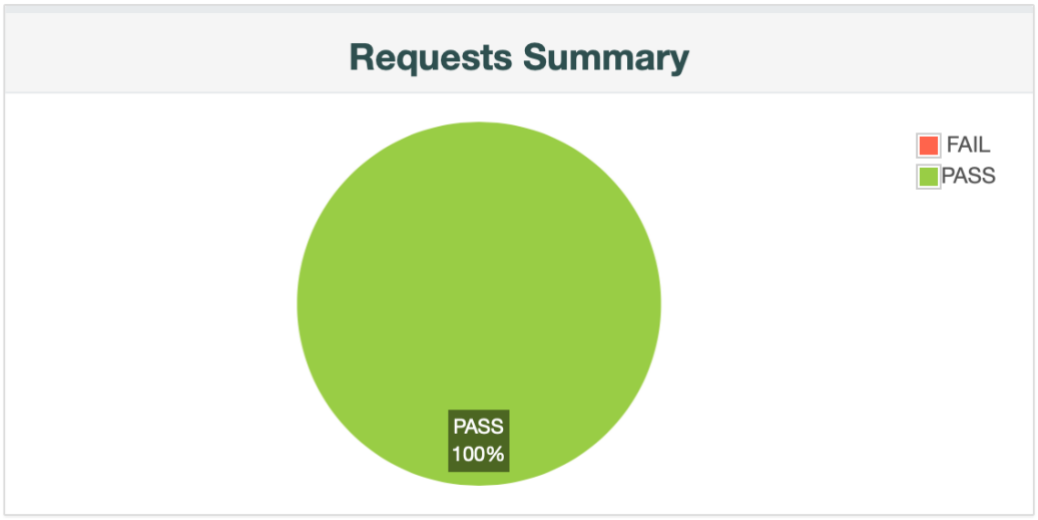


Рисунок Г.18 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	1787.35	336	5417	1642.00	2875.90	3359.00	4203.96	54.47	20.75	47.19
HTTP Request	10000	0	0.00%	1787.35	336	5417	1642.00	2875.90	3359.00	4203.96	54.47	20.75	47.19

Рисунок Г.19 - Статистика запиту (рисунок виконано самостійно)

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	10000	0										

Рисунок Г.20 - Помилки запиту (рисунок виконано самостійно)

TransactionGetall

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.334	500 ms	1 sec 500 ms	Total
0.334	500 ms	1 sec 500 ms	HTTP Request

Рисунок Г.21 - Apdex запиту (рисунок виконано самостійно)

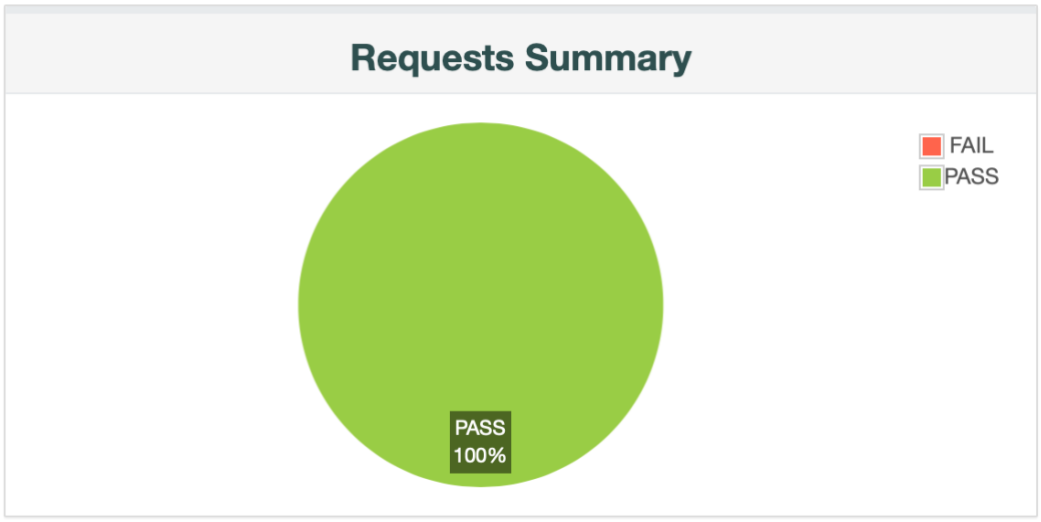


Рисунок Г.22 - Зведення запиту (рисунок виконано самостійно)

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	1414.94	381	3937	1287.00	2084.00	2418.00	3323.99	67.86	346.91	57.32
HTTP Request	10000	0	0.00%	1414.94	381	3937	1287.00	2084.00	2418.00	3323.99	67.86	346.91	57.32

Рисунок Г.23 - Статистика запиту (рисунок виконано самостійно)

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	10000	0										

Рисунок Г.24 - Помилки запиту (рисунок виконано самостійно)

ДОДАТОК Д

Тези V Міжнародна студентська наукова конференція «РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ»

Горкун Дмитро Олександрович, здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Налєскіна Тетяна Сергіївна, здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Русакова Наталія Євгенівна, доцент кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

ПРОЕКТУВАННЯ ANGULAR ДОДАТКУ ДЛЯ ПРОГРАМНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАРЯДНИХ СТАНЦІЙ ЕЛЕКТРОМОБІЛЕЙ ВИКОРИСТОВУЮЧИ ONION АРХІТЕКТУРУ

Розробка сучасних веб-додатків вимагає використання гнучких та масштабованих підходів до архітектури програмного забезпечення. Це особливо важливо для комплексних бізнес-систем, таких як управління автопарками електромобілів. Ефективне споживання електроенергії для українських бізнесів пов'язаних з електротранспортом є важливим, тож зручний додаток, основним завданням якого є автоматизація процесу управління споживання електроенергії, повинен надавати максимально швидку, стійку та інтуїтивно зрозумілу клієнтську частину. Подібні проєкти містять багато бізнес-логіки, інтеграцій з зовнішніми сервісами, користувацьких інтерфейсів та іншої інфраструктури. Щоб ефективно керувати такою складністю, потрібні структуровані архітектурні рішення.

Onion архітектура[1] далеко не інноваційний підхід, котрий використовується на серверній стороні додатків, що не можна сказати про клієнтську частину. Вона поділяє додаток на послідовні рівні відповідальності. Головною перевагою такої організації є слабке зв'язування між компонентами системи та їх висока змінюваність.

Слід відмітити, що Angular та інші сучасні фреймворки підтримують асинхронний код. Що робить цю архітектуру ще цікавішою для реалізації!

Метою даної архітектури є забезпечення розподілу обов'язків і збереження логіки домену ізольованою.

Найбільш критичними частинами Angular-проєкту є:

– наявність легких та чистих компонентів;
– управління станом системи;
– уникнення розповсюдження бізнес-логіки додатку між компонентами та сервісами.

Тепер, визначімо, що є нашим доменом і що потрібно від нього відокремити. У Frontend-проєктах можна виділити чотири великі частини:

– представлення, що відображаються користувачеві, тобто компоненти;
– клієнти, які використовуються для спілкування з API;
– стан, наше єдине джерело істини;
– домен, який містить бізнес-логіку.

Уявімо, що користувачеві доступний список депо. Коли користувач видаляє

Рисунок Д.1 - Скріншот першої сторінки тез зі збірника наукової конференції (рисунок виконано самостійно)

елемент, виконується оркестрація видалення через клієнтський шлюз, а потім видаляє елемент зі стану через шлюз зберігання. Ця оркестрація прихована за методами фасаду [2].

У домені ці шлюзи є інтерфейсами, котрі використовують мезанізм Dependency Injection Angular фреймворку. Використовуючи даний підхід реалізації інтерфейсів можуть бути динамічно замінені в процесі розробки, до прикладу на моковий сервіс в процесі тестування. Бізнес-логіка – це будь-яка логіка, не пов'язана з модулями представлення, клієнта і стану. Якщо додатку потрібно виконувати бізнес-логіку на клієнтській стороні, ця логіка виконується доменом. Відповідно до описаної архітектури була побудована діаграма компонентів (рис. 1).

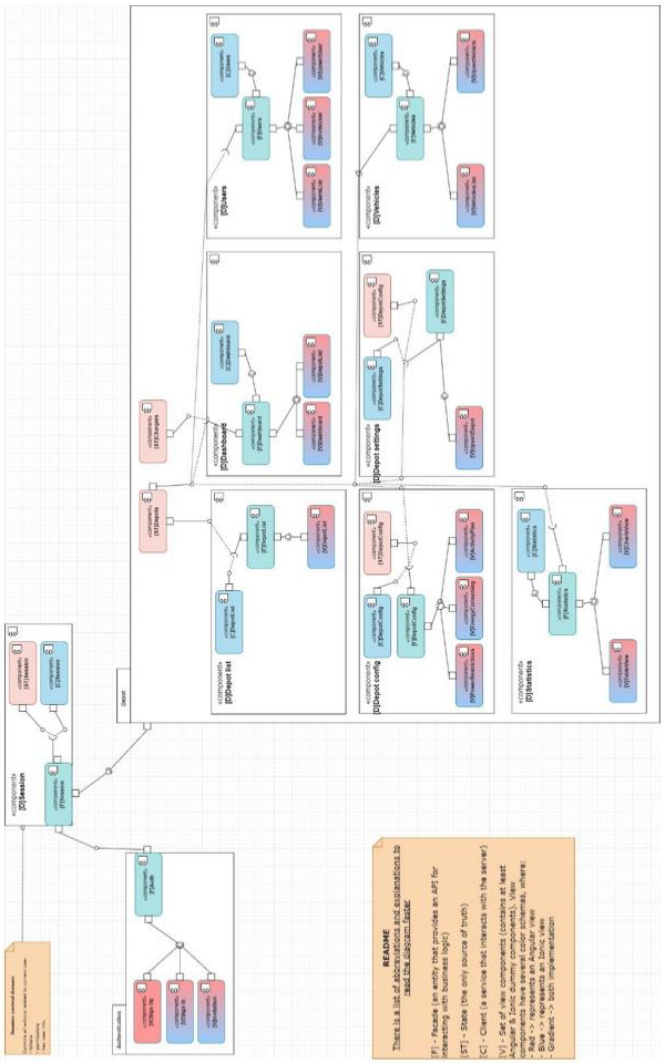


Рис. 1. **Діаграма компонентів системи моніторингу зарядних станцій використовуючи Onion архітектуру**

Рисунок Д.2 - Скріншот другої сторінки тез зі збірника наукової конференції (рисунок виконано самостійно)

Діаграма компонентів дозволить більш наочно ознайомитися з підходом використання Onion архітектуру в контексті розробки веб-додатку моніторингу зарядних станцій електромобілем.

Під час розробки додатку важливо приділити час визначенню чіткої архітектури. Впровадження Onion архітектури в проєкті Angular – це вагомий крок на шляху створення добре структурованого програмного забезпечення.

Інтеграція її архітектурних принципів не лише підвищує якість коду, але й полегшує співпрацю та масштабування.

Отже, розпочинаючи впровадження Onion архітектури у Angular проєкт, треба пам'ятати, що додаткові зусилля, спрямовані в організацію коду та розподіл обов'язків, окупляться в довгостроковій перспективі.

Список використаних джерел:

1. NG Poland Conf // YouTube: [Веб-сайт]. 2023. URL: https://youtu.be/nNIUTBHaiG8?si=-6N7aI5YVEevc2e_ (дата звернення: 27.11.2023).
2. Jérôme Navez // Medium: [Веб-сайт]. 2023. URL: https://medium.com/@navez.jerome/applying-the-onion-architecture-to-angular-projects-b37736d2c996 (дата звернення: 01.12.2023).

Рисунок Д.3 - Скріншот третьої сторінки тез зі збірника наукової конференції (рисунок виконано самостійно)

ДОДАТОК Е

Специфікація проєкту

E-Charge Hub

Специфікація вимог до програмного забезпечення

3.0

01.06.2024

Налєскіна Тетяна

Горкун Дмитро

Чубаров Євген

Рубель Денис

Історія версій

Дата	Опис	Автор	Коментар
01.05.24	Версія 1.0	Налєскіна Тетяна	Створення документу
20.05.24	Версія 2.0	Горкун Дмитро	Оновлення використаних технологій
01.06.24	Версія 3.0	Чубаров Євген	Оновлення використаних технологій з боку бекенду

Зміст

1. Вступ

1.1 Огляд продукту

E-Charge Hub – програмна система моніторингу зарядних станцій електромобілів забезпечує контроль, управління та оптимізацію процесу зарядки електромобілів. Основні функції включають збір даних, аналіз ефективності станцій, управління процесом зарядки, відстеження споживання енергії та забезпечення безпеки даних. Система спрямована на ефективне управління зарядними станціями з урахуванням зростання попиту на електромобілі та необхідності розвитку відповідної інфраструктури. Монетизація проекту досягається шляхом придбання програмного забезпечення нашого продукту.

1.2 Мета

Метою програмної системи моніторингу зарядних станцій електромобілів є забезпечення зручності та ефективності процесу зарядки для користувачів електромобілів. Наша програмна система спрямована на те, щоб забезпечити адміністраторам зарядних станцій інноваційний та інтегрований підхід до управління, моніторингу та оптимізації їх інфраструктури. Ми розглянемо, як система надає централізоване управління, реальний час, аналітику, а також забезпечує високий рівень безпеки та енергоефективності. Розкриючи ці аспекти, ми створимо повністю осяжний образ та зрозуміння проекту, який сприятиме не тільки подальшому розвитку інфраструктури зарядних станцій, але й покращенню користувацького досвіду та сприянню сталому розвитку.

1.3 Межі

Програмна система E-Charge Hub має надавати можливості що дозволить зручно керувати депо та зарядними станціями в них, завчасно бронювати зарядні станції для подальшого використання та моніторингу енергоспоживання. Продукт має складатися з веб-додатку та мобільної версії, яка буде повністю відтворювати функціонал продукту. Система повинна мати можливість реєстрації та аутентифікації користувачів з різними ролями та надавати доступ до функцій згідно з ролями користувачів системи.

Серверна частина має бути реалізована з використанням мікросервісної архітектури, клієнтська з використанням Onion архітектури. Здійснення контролю над процесом розробки має проводитися за допомогою програмного забезпечення Jira.

1.4 Посилання

Текст документу містить наступні посилання:

1. Горкун Д.О., Налєскіна Т.С. Проектування Angular додатку для програмної системи моніторингу зарядних станцій електромобілів з використанням Onion архітектури. Тези V Міжнародна студентська наукова конференція «РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ», м. Умань, 2024. С. 60-62. URL: https://doi.org/10.36074/liga-inter-02.02.2024
2. Чубаров Є.Е., Рубель Д.А. Проєктування серверної частини системи керування та моніторингу зарядними станціями. Тези XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» конференції «Інформаційні інтелектуальні системи», м. Харків, 2024. С. 306-308. URL: https://doi.org/10.30837/IYF.IIS.2024.306
3. Посилання на репозиторій: https://github.com/Nexus20/ChargingStation.Backend , https://github.com/Vspominay/ev-charging-station

1.5 Означення та абревіатури

В документі використовуються наступні означення та абревіатури:

OCPP – Open Charge Point Protocol

HTTP – HyperText Transfer Protocol

gRPC – Google Remote Procedure Calls

REST – REpresentational State Transfer

SQL – Structured Query Language

K8S – Kubernetes

ACR – Azure Container Registry

AKS – Azure Kubernetes Service

.2. Загальний опис

.2.1 Перспективи продукту

У зв'язку зі стрімким розвитком та впровадженням електромобільної технології та зростанням інтересу до сталих енергетичних рішень, створення програмної системи моніторингу зарядних станцій електромобілів стає необхідністю. За даними Міжнародного агентства з енергетики (IEA), кількість електромобілів на світі стрімко зростає, досягнувши позначки у 10 мільйонів одиниць у 2022 році, що свідчить про поступову трансформацію транспортної системи. Цей іноваційний підхід вимагає ретельного врахування численних передумов для успішного розгортання та оптимального функціонування системи.

Програмна система орієнтована на такі сегменти ринку:

1. Ринок послуг з моніторингу та аналізу: розробка системи, яка забезпечує детальний моніторинг ефективності зарядних станцій, створює можливість для підприємців надавати послуги з аналізу та оптимізації зарядних станцій для власників та операторів електротранспорту.

2. Системи управління та сервісні контракти: реалізація систем управління дозволяє підприємцям пропонувати сервісні контракти для підтримки та оптимізації роботи зарядних станцій, що включають в себе віддалене моніторинг, технічну підтримку та регулярне обслуговування.

3. Розвиток інфраструктури зарядних станцій: сприяння розвитку інфраструктури зарядних станцій для електромобілів за допомогою вдосконаленої системи моніторингу може стати ключовим фактором для бізнесу в області будівництва та управління інфраструктурою.

4. Платформи для користувачів електромобілів: розробка мобільних додатків та платформ для користувачів електромобілів, які надають інформацію про доступність та стан зарядних станцій, може створити нові можливості для реклами, партнерських програм та користувачських планів.

5. Аналітика для електромобільних виробників: надання аналітичних звітів та статистики виробникам електромобілів щодо використання їх продукції може стати основою для покращення технічних характеристик, а також розробки нових моделей, що задовольнять потреби ринку.

6. Енергетичні консалтингові послуги: врахування споживання електроенергії та оптимізація часу зарядки може створити можливості для компаній, які спеціалізуються на консалтингових послугах у галузі енергетики.

7. Екологічно орієнтовані ініціативи:розвиток системи моніторингу може бути використаний для підтримки екологічних ініціатив та отримання фінансової підтримки від урядових та неприбуткових організацій.

2.2 Функції продукту

Програмна система для моніторингу зарядних станцій електромобілів є комплексним інформаційним рішенням, яке об'єднує в собі різноманітні компоненти і функції для ефективного керування та контролю зарядними станціями. Основні складові програмної системи включають:

Моніторинг і збір даних. Система здатна отримувати дані з різних зарядних станцій, включаючи інформацію про стан заряду, використання енергії, доступність станцій та інші параметри.

Аналітика та звітність. Можливість аналізувати накопичені дані для отримання важливої інформації щодо ефективності зарядних станцій, споживання енергії, роботи системи тощо. Генерація звітів та статистики для подальшого аналізу та прийняття управлінських рішень.

14 Управління зарядками. Функціонал для керування процесом зарядки,

включаючи планування заряду, встановлення пріоритетів, управління через віддалений доступ тощо.

Система оповіщень та аварійного управління. Можливість автоматичного виявлення проблем та аварій, надсилання сповіщень та управління ситуаціями навіть у відсутності користувача.

Інтерфейс для користувачів. Зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє легко взаємодіяти з програмною системою, переглядати дані, встановлювати параметри та отримувати звіти.

Захист та безпека. Забезпечення захисту даних, використання шифрування, аутентифікації користувачів та інших заходів для забезпечення безпеки і конфіденційності інформації.

2.3 Характеристики користувачів

Функції, які планується реалізувати у першому релізі, було розділено на групи користувачів для полегшення їх представлення. Відомо, що система буде взаємодіяти з п'ятьма типами користувачів: власники, адміністратори від компаній, моніторингова система, водії та працівники.

Власник (Owner) може займатись менеджментом адміністраторів та депо, моніторингом статусу зарядних станцій депо, що включає в собі моніторинг конкретного роз'єму, авторизуватись у системі.

Адміністратор (Administrator) може займатись менеджментом користувачів, дивитись статистику, що отримується зі звітів, зробити резервацію, авторизуватись. Займатись менеджментом графіку зарядки, що включає налаштування інтервалів споживання електроенергії та налаштування обмежень потужності.

Моніторингова система (Monitoring system) відправляє пуш-повідомлення.

Водій (Driver) може моніторити конкретний роз’єм.

Працівник (Employee) може моніторити статус зарядних станцій депо, займатись менеджментом графіку зарядки, що включає налаштування інтервалів споживання електроенергії та налаштування обмежень потужності, робити резервації та авторизуватись у системі.

2.4 Загальні обмеження

Програмна система для моніторингу зарядних станцій електромобілів склається з серверної та клієнтської частини. Серверна частина буде реалізована за

допомогою мікросервісної архітектури. Концепція мікросервісної архітектури полягає у дизайні архітектури програмної системи таким чином, що функціональність розподіляється між сервісами. Розробка мікросервісів для системи буде проводитися на мові програмування C# з використанням платформи .NET 8 та фреймворку ASP.NET Core. Обрання цих конкретних технологій пояснюється їхньою здатністю підтримувати широкий спектр бібліотек, зокрема ті, які використовуються для роботи з хмарними технологіями, gRPC і RabbitMQ. Технологічний стек для розгортання програмної системи моніторингу зарядних станцій електромобілів використовує сучасні інструменти та підходи для ефективного управління та функціонування системи. У цьому контексті, використання Docker є важливим елементом для забезпечення ізольованості сервісів, а K8S є ключовим елементом оркестрації ресурсів. Для якнайшвидшої доставки оновлень клієнту, налаштовано CI/CD процеси за допомогою Azure DevOps Pipelines, що дозволяє автоматизувати розгортання рішення в хмару з використанням ACR та AKS відповідно.

Клієнтська частина розробляється за допомогою декількох технологій: Angular, Ionic, RxJs, Ng-bootstrap, FullCalendar, Ngx-translate, dayjs, Jest. Для управління базами даних у проекті обрано систему Microsoft SQL Server (MS SQL Server).

2.5 Припущення та залежності

Припущення 1. Продукт буде затребуваний на ринку;

Припущення 2. Вихід обладнання з експлуатації може призвести до відмови сервісу;

Припущення 3. Зворотна сумісність;

Залежність 1. Використання протоколу OCPP для стандартизації підходів комунікації між станціями та центральною системою;

Залежність 2. Серверна частина буде реалізована на платформі ASP.NET;

Залежність 3. Використання СУБД Azure SQL Server та Azure Table Storage зберігання даних;

Залежність 4. Використання Docker для контейнеризації та K8S для оркестрації ресурсів;

Залежність 5. Використання брокеру повідомлень RabbitMQ для асинхронної комунікації;

Залежність 6. Використання Redis Cache для кешування;

Залежність 7. Застосування Azure DevOps Pipelines для налаштування процесів CI/CD;

Залежність 8. Використання ACR та AKS для розгортання серверної частини в хмарі Azure;

Залежність 9. Для реалізації клієнтської частини використовується Angular, Ionic, RxJs, Ng-bootstrap, FullCalendar, Ngx-translate, dayjs, Jest.

Залежність 10. Застосування Render для розгортання клієнтської частини.

Залежність 11. Швидкість відгуку та оновлення залежать від потужності та стійкості сервера, який приймає та обробляє дані, а також від швидкості інтернет-з'єднання;

Залежність 12. Залежність часу розробки цього проекту від часу, який розробники витрачають вивчення та розробку паралельних завдань.

3. Конкретні вимоги

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Неавторизований користувач має потрапляти на сторінку входу в систему (рисунок 1).

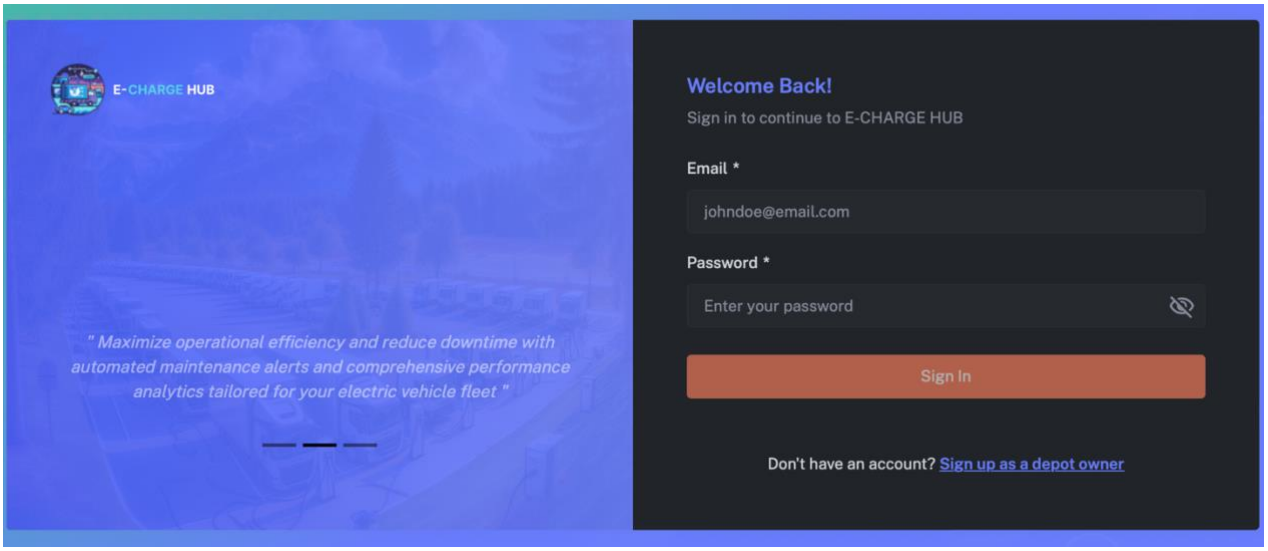


Рисунок 1 – Лендінг сторінка

На сторінці входу мусить бути логотип та слайдер з описом переваг системи. Також має бути форма входу в систему за поштою та паролем. Тут же мусить бути посилання на сторінку реєстрації.

На формі реєстрації (рисунок 2) повинні бути наступні елементи. Поле для введення електронної пошти та номеру телефону, де користувач вводить свою адресу електронної пошти для реєстрації в системі. Поле для введення пароля, яке дозволяє користувачеві створити пароль для захисту свого облікового запису. Також необхідно мати поле для введення імені, де користувач вводить своє ім'я, і поле для введення прізвища, де вводиться прізвище. Після заповнення всіх полів користувач натискає кнопку "Реєстрація", щоб завершити процес реєстрації. Для тих, хто вже зареєстрований, повинна бути кнопка "Увійти" для переходу на сторінку авторизації. Крім того, на формі повинно бути посилання на політику конфіденційності, яку користувач повинен прийняти під час реєстрації.

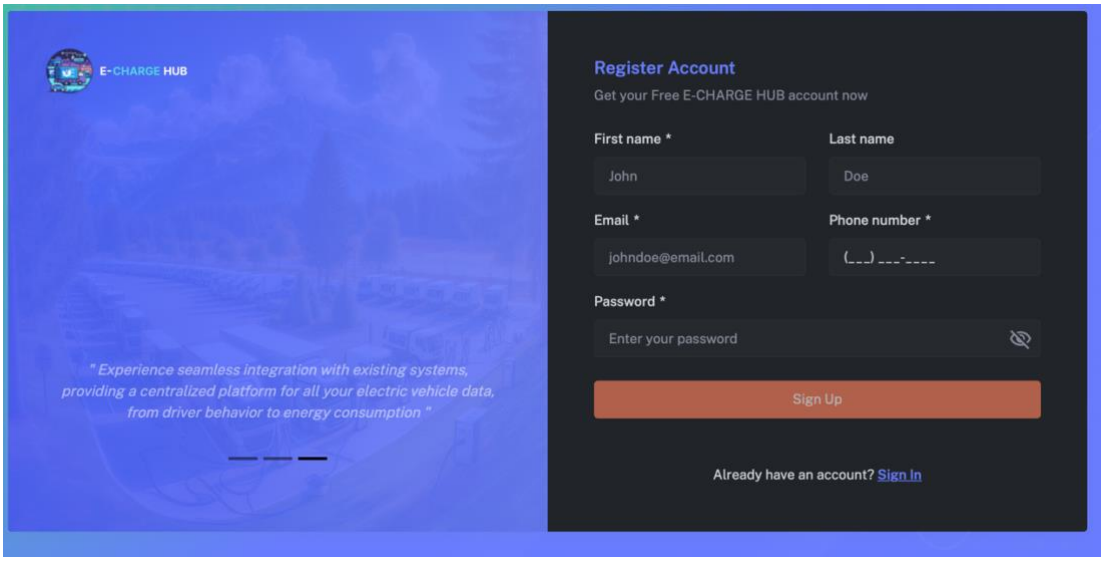


Рисунок 2 – Форма для залишення заявки

На сторінці списку користувачів системи моніторингу зарядних станцій депо (рисунок 3) необхідно відобразити заголовок сторінки, який ясно вказує, що це список користувачів системи. Далі має бути пошуковий рядок, що дозволяє швидко знайти конкретного користувача за ім'ям або іншими критеріями. Сам список користувачів повинен містити таблицю з такими даними: ім'я користувача, прізвище, електронна пошта, роль у системі (наприклад, адміністратор) та дату останнього входу в систему. Кожний рядок таблиці повинен мати кнопки для редагування або видалення користувача, а також можливість перегляду детальної інформації про користувача. Для зручності можна додати можливість сортування даних у таблиці за різними критеріями, такими як ім'я або дата останнього входу. Також бажано мати кнопку для додавання нового користувача, яка перенаправляє на форму реєстрації нового користувача.

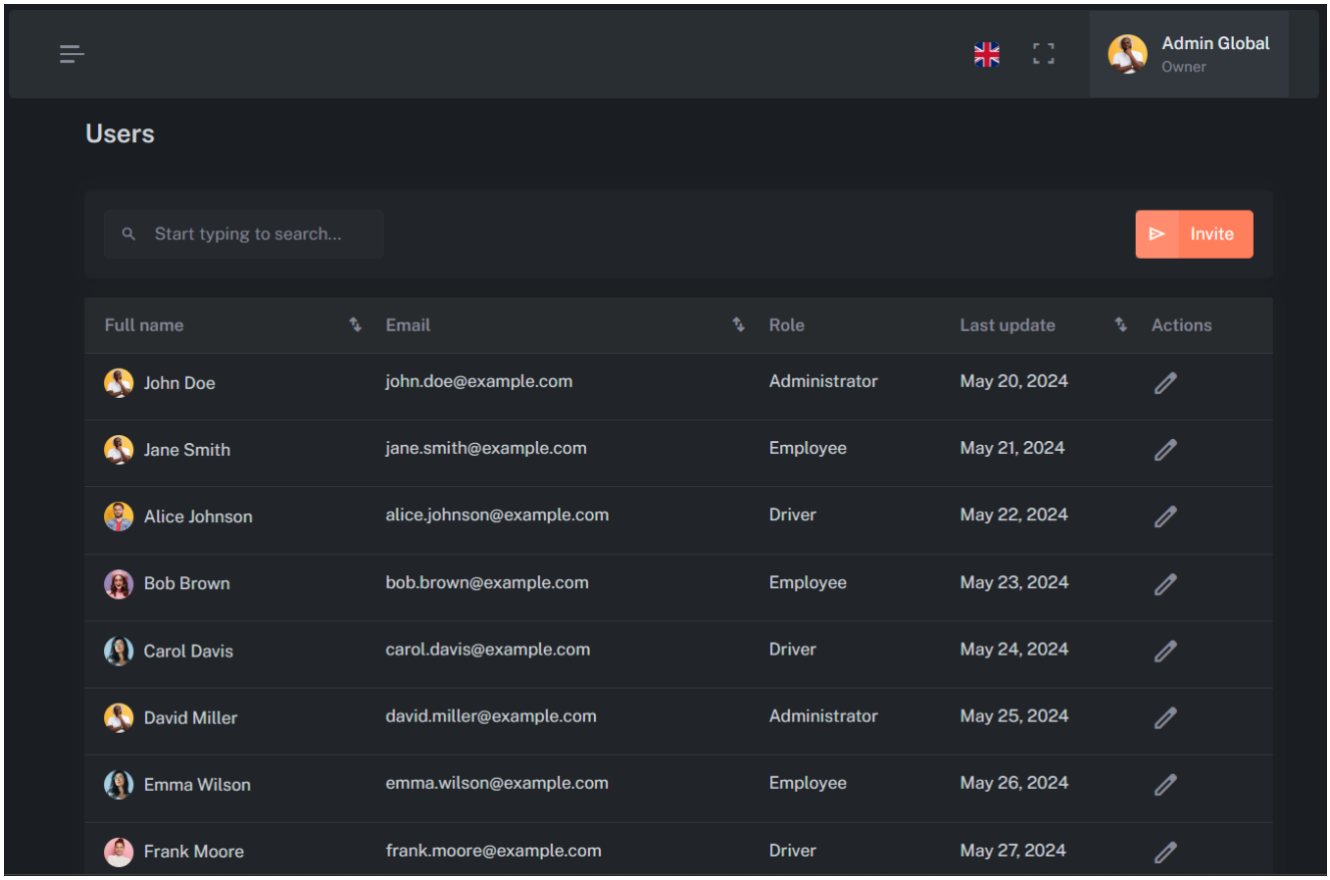


Рисунок 3 - Сторінка списку користувачів системи моніторингу зарядних станцій депо

На сторінці створення резервації зарядної станції (рисунок 4) повинні бути представлені наступні елементи. Спочатку має бути заголовок сторінки, який чітко вказує, що це форма для створення резервації зарядної станції. Повинно бути поле для вибору зарядної станції зі списку доступних станцій, де користувач може вибрати потрібну станцію для резервації та Ocpp тег. Додати поле для коментарів, де користувач може залишити додаткові примітки щодо резервації, початок зарядки та кінець. Кнопка підтвердження резервації повинна бути добре помітною, щоб користувач міг легко завершити процес. Після натискання на кнопку підтвердження повинно з'явитися повідомлення про успішне створення резервації або повідомлення про помилку у разі некоректного введення даних. На сторінці також має бути кнопка для скасування, яка дозволяє користувачеві повернутися до попередньої сторінки без збереження змін.

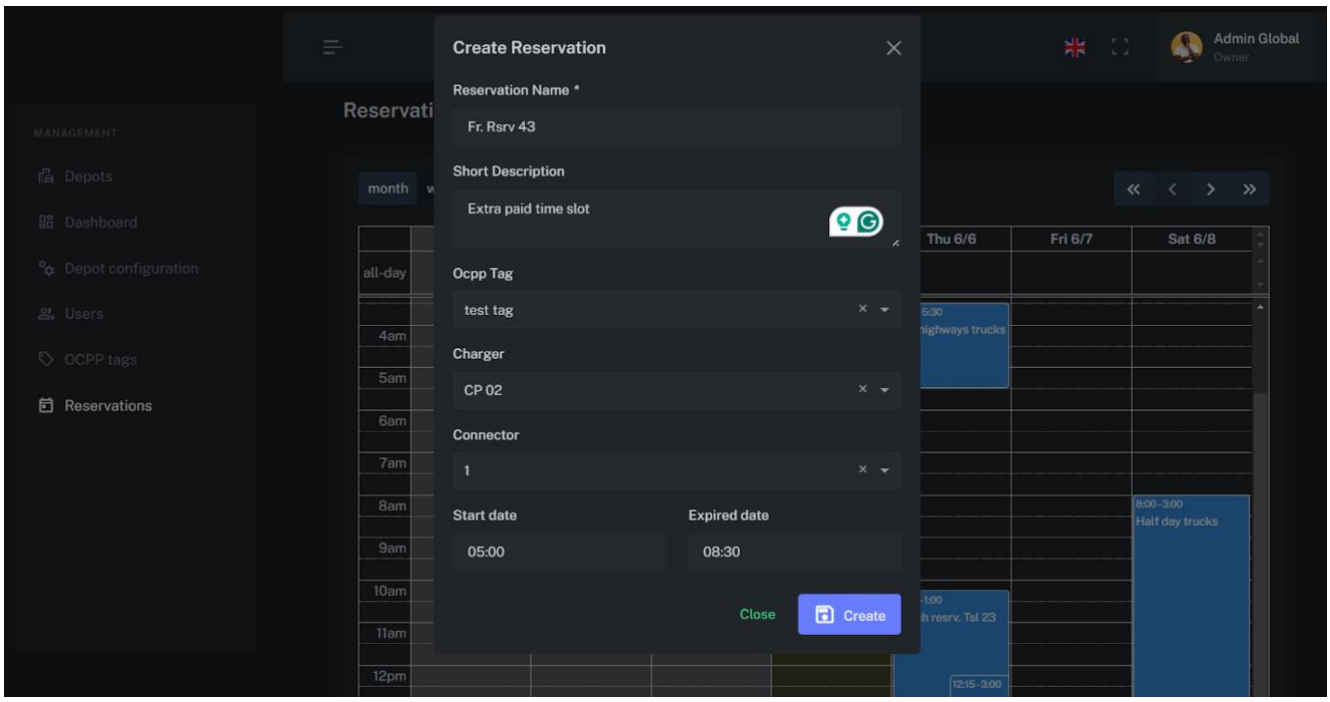


Рисунок 4 - Сторінка створення резервації зарядної станції

На сторінці календаря (рисунок 5) повинні бути представлені наступні елементи. Заголовок сторінки, який чітко вказує, що це календар, де відображаються заплановані резервації зарядних станцій. Сам календар має бути інтерактивним, дозволяючи користувачам переглядати дні, тижні та місяці. Користувачі повинні мати можливість легко переходити між різними періодами, використовуючи кнопки для переміщення вперед і назад. У кожній клітинці календаря має бути відображена інформація про наявні резервації, такі як час і станція, яка зарезервована. Для зручності можна додати можливість перегляду деталей резервації при наведенні курсору або натисканні на конкретну резервацію. Повідомлення про помилки або відсутність резервацій також мають бути чітко відображені.

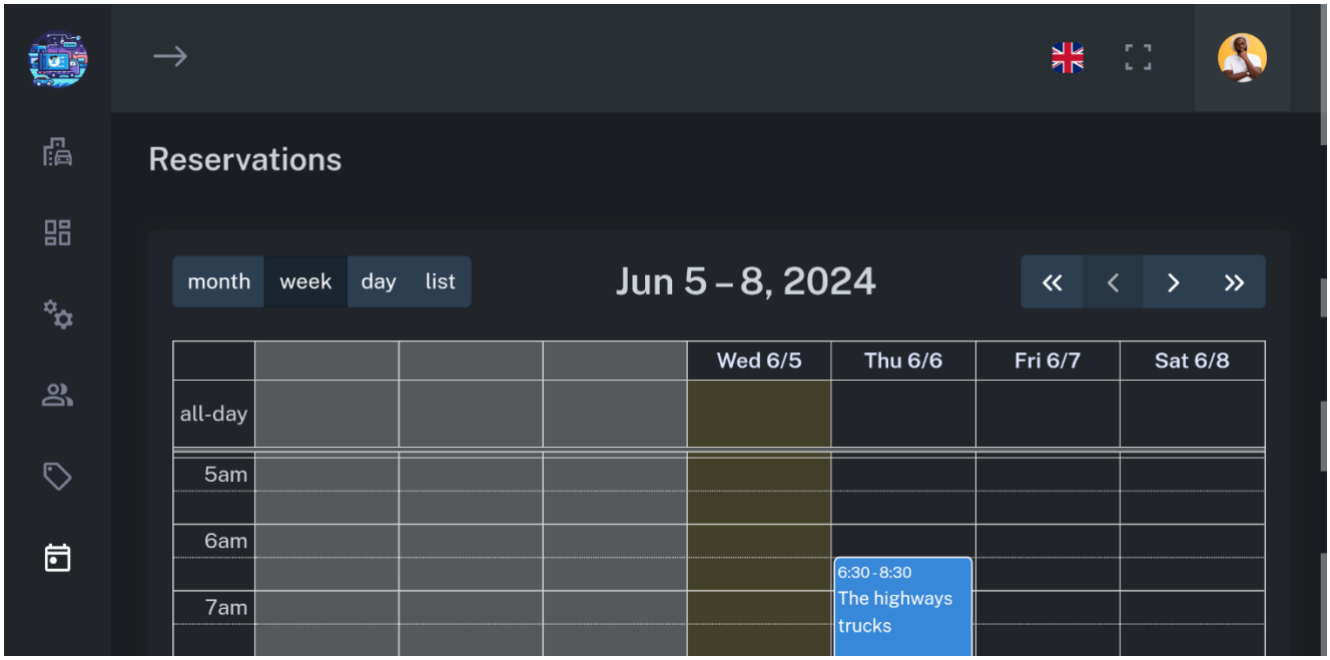


Рисунок 5 - Календар резервацій

.

3.1.2 Апаратний інтерфейс

Оскільки веб-сайт не має призначеного обладнання, прямих апаратних інтерфейсів немає.

3.1.3 Програмний інтерфейс

Веб-додаток повинен бути розроблений з урахуванням сумісності з усіма сучасними браузерами, такими як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо. Використання технологій HTML5, CSS3 та JavaScript ES6 допоможе забезпечити сумісність та підтримку різних браузерів. Для роботи з мобільною версією потрібно буде її завантажити через Play Market та App Store

3.1.4 Комунікаційний протокол

Зарядна станція взаємодіє через протокол OCPP та веб-сокети з відповідним сервісом обробки підключень. RabbitMQ підтримує протокол AMQP. Протокол OCPP є стандартним протоколом для обміну повідомленнями з зарядними станціями. Взаємодія між сервісами – за допомогою gRPC (окрім випадків, де комунікація відбувається асинхронно за допомогою RabbitMQ). В

3.1.5 Обмеження пам’яті

Веб-сайт не може безпосередньо контролювати кількість доступної пам'яті на комп'ютері користувача. Прямого обмеження пам'яті комп'ютера для користування веб-сайтом немає.

3.1.6 Операції

Для розробки веб-додатку використовується архітектурний стиль REST, що базується на принципах HTTP-протоколу. Основні операції, які можна виконувати з використанням REST, включають наступні:

- GET - для отримання ресурсу або його представлення;
- POST - для створення нового ресурсу;
- PUT - для оновлення існуючого ресурсу або створення нового, якщо він не існує;
- DELETE - для видалення ресурсу.

REST також використовує URL-шаблони для ідентифікації ресурсів та параметризації запитів.

3.1.7 Функції продукту

ГФ-1: Управління інформацією; створення депо. Надання власникам мереж депо можливості керувати інформацією; створювати свої депо.

ГФ-2: Створення співробітника. Надання адміністрації депо можливості створювати користувачів (співробітників/адміністраторів) для управління мережами зарядних станцій.

ГФ-3: Управління стану заряду станцій. Забезпечення інструментів для управління роботи станцій.

ГФ-4: Моніторинг стану депо в реальному часі. Надати можливість слідкувати за станом депо/зарядними станціями в реальному часі.

ГФ-5: Налаштування споживання електроенергії в рамках депо. Можливість глобального налаштування обмежень споживання енергії в заданих часових інтервалах депо.

ГФ-6: Статистика та звітність. Збір та надання статистичних даних для аналізу ефективності роботи зарядних станцій.

ГФ-7: Створення профілів зарядки. Конфігурація індивідуальних та групових профілів для контролю потужності зарядних станцій.

ГФ-8: Управління резервацією зарядних станцій. Організація системи резервації для ефективного використання станцій.

ГФ-9: Зміна мови додатку. Надані інструменту для зміни мови додатку

ГФ-10: Запрошення існуючого співробітника в депо. Реалізувати можливість надсилати запрошення наявним користувачам в системі приєднатися до депо.

3.2 Властивості програмного продукту

Користувачі розроблюваної програмної системи будуть широко розпорошені. Для доступу до системи необхідне стабільне інтернет-з’єднання. Оскільки дані будуть зберігатися віддалено на сервері, максимальний час відгуку встановлено до хвилини. Доступ до різноманітних функцій буде залежати від типу користувача. Цілісність та безпека зберігання даних буде забезпечуватися базою даних.

Програмна система має мобільну версію.

3.3 Атрибути програмного продукту

3.5.1 Надійність

3.5.1.1 Система повинна мати високий рівень надійності, користувач повинен отримувати тільки перевірені дані.

3.5.1.2 В разі виникнення помилки, система не повинна припиняти свою роботу, користувач повинен побачити зрозуміле йому повідомлення з описом

проблеми, що виникла (без коду помилки і інформації, яку зрозуміти зможе тільки розробник).

3.5.2 Доступність

3.5.2.1 Користуватися веб-додатком зможуть всі люди, що володіють українською або англійською мовою.

3.5.2.2 Для отримання доступу до мобільної версії потрібно її завантажити.

3.5.2.3 Тільки авторизовані користувачі можуть отримати доступ до функцій продукту.

3.5.2.4 Тільки адміністратору депо доступні функції видалення / додавання / зміни інформації про працівників.

3.5.3 Безпека

3.5.3.1 Кожен обліковий запис в системі буде захищено паролем, який встановлює користувач. Надійний пароль має складатися з більш ніж з 6 символів і обов'язково містити в собі принаймні одну цифру.

3.5.3.2 Увійти в обліковий запис можна тільки після введення правильних пароля та логіна.

3.5.3.3 Усі дані компанії мають зберігатися в окремій захищеній базі.

3.5.3.4 Вкладки мають бути доступні в залежності від дозволів ролі співробітника.

3.5.4 Супровід

3.5.4.1. Система повинна бути розроблена таким чином, щоб була можливість додавання нових функцій в майбутньому.

3.5.4.2. Система повинна бути розділена на частини (серверна, клієнтська, мобільно версії), щоб виправлення помилок займало якомога менше часу.

3.5.5 Переносимість

3.5.5.1 Веб-додаток повинен запускатися в усіх сучасних браузерах незалежно від пристрою, який використовує користувач.

3.4.5.2 Мобільна версія повинен працювати на Android та iOS пристроях.

3.5.6 Продуктивність

3.5.6.1 Час відгуку на перехід між вкладками веб-додатку повинен бути менше 1 секунди.

3.5.6.2 Після натискання кнопки "Додати співробітника" співробітник має бути одразу доданий у список працівників, йому має прийти лист для встановлення паролю на пошту протягом 5 хвилин.

3.4 Вимоги бази даних

В якості сховища даних було обрано СКБД Microsoft SQL Server, яка має наступні переваги:

масштабованість та продуктивність: MS SQL Server демонструє високу продуктивність та масштабованість. Він відмінно працює на великих корпоративних серверах, так і на десктопах або дрібних серверах, дозволяючи ефективно керувати великими обсягами даних;

інтеграція з іншими продуктами Microsoft: Оскільки MS SQL Server — продукт Microsoft, він бездоганно інтегрується з іншими продуктами Microsoft, такими як Windows Server, .NET, SharePoint, а також з Microsoft Office;

безпека даних: MS SQL Server має потужні механізми безпеки. Він пропонує різноманітні функції для контролю доступу, шифрування даних та аудиту, що дозволяє компаніям відповідати вимогам регуляторів.

3.5 Інші вимоги

Система не повинна дозволяти незареєстрованим користувачам отримувати доступ до функцій сервісу. Лендінг сторінка має бути окремою частиною, яка буде

загальнодоступною, а веб-сайт з функціональністю доступний лише при переході на авторизацію. Використання зображень, фото та інших матеріалів на сайті не повинно порушувати авторських прав. Інтерфейс веб додатку повинен змінюватися (підганяти розміри) під пристрої з різними розмірами екрану. Інтерфейс повинен бути інтуїтивно зрозумілий і приємний у використанні для людей будь-яких вікових категорій.

4. Додаткові матеріали

4.1 Діаграми баз даних

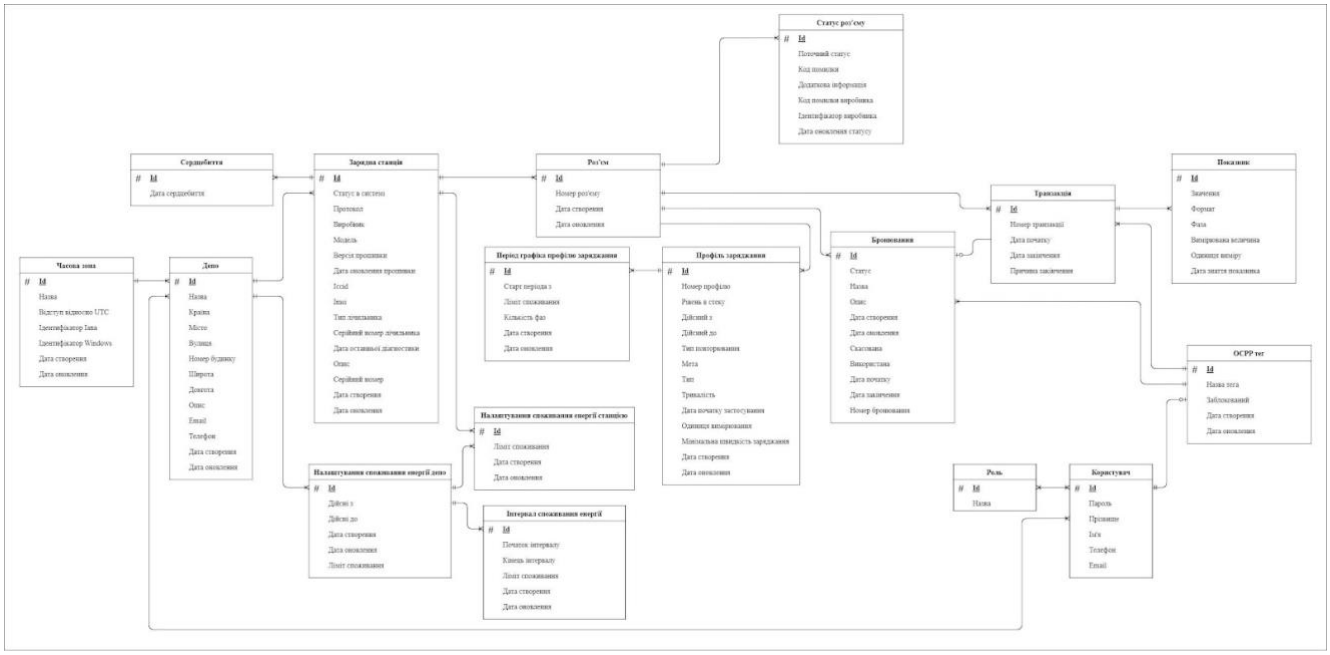


Рисунок 4 – ER-діаграма бази даних

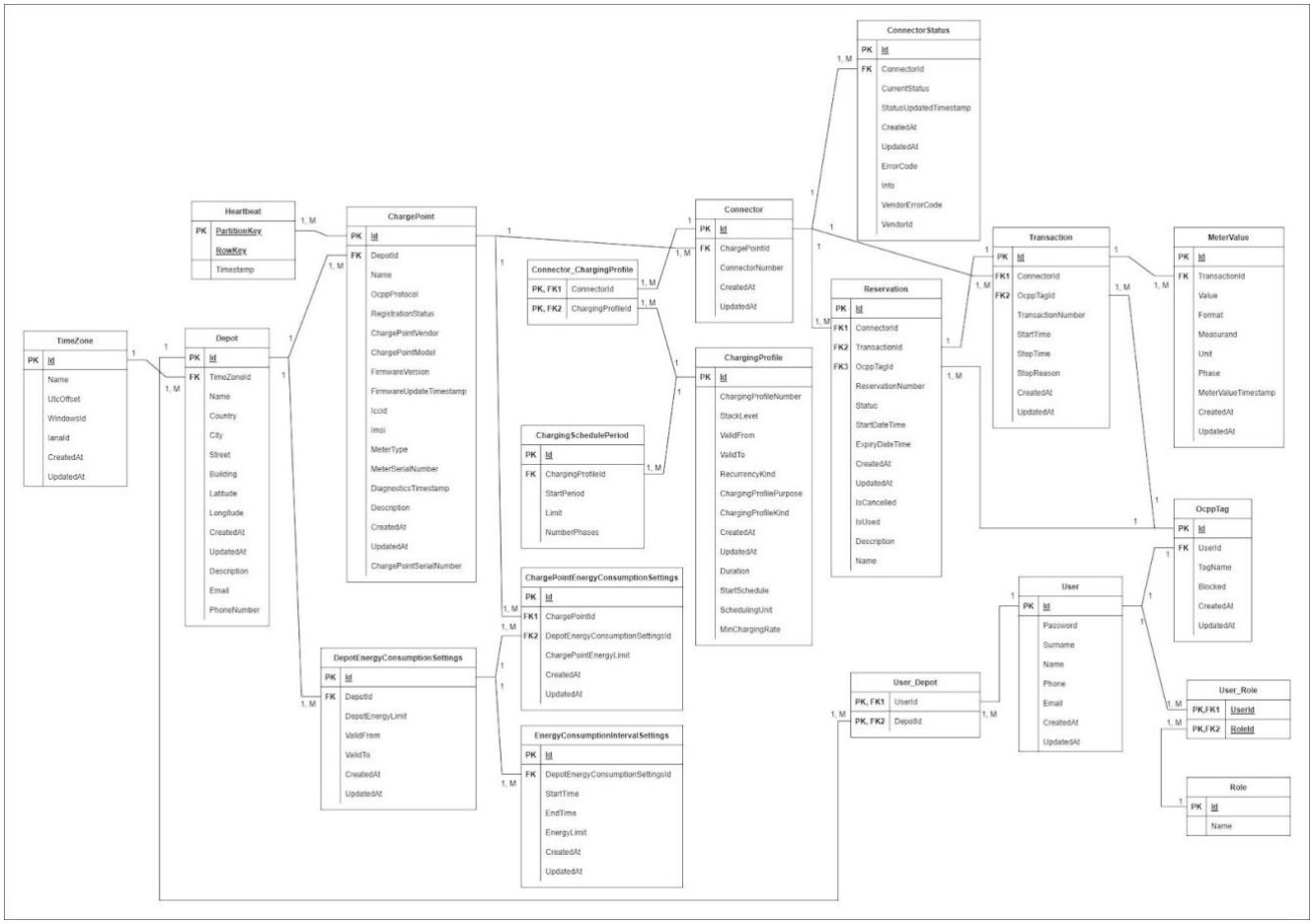


Рисунок 5 – Логічна модель бази даних

4.2 Діаграма прецедентів

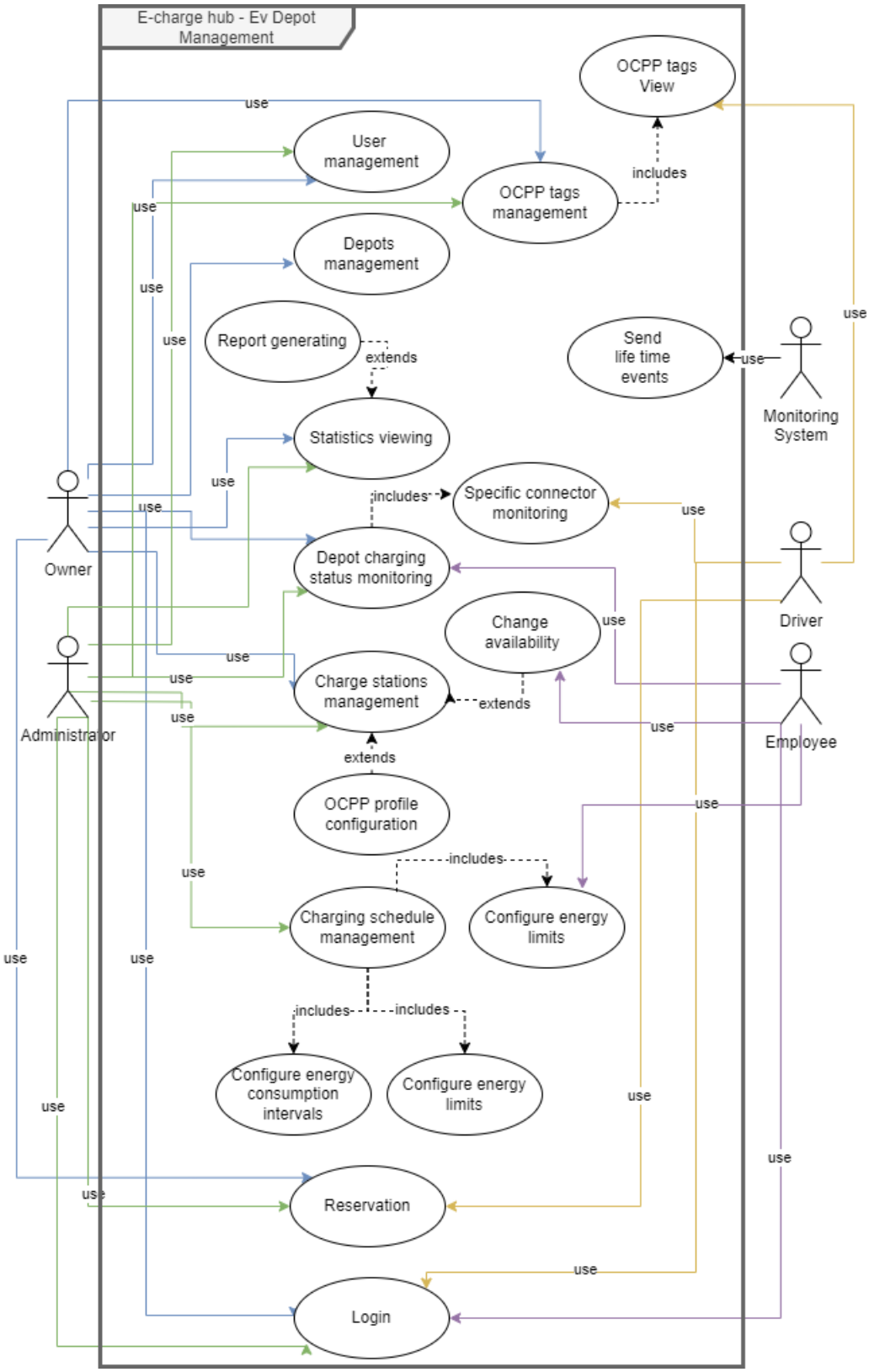


Рисунок 6 – Діаграма прецедентів

4.3 Діаграма діяльності

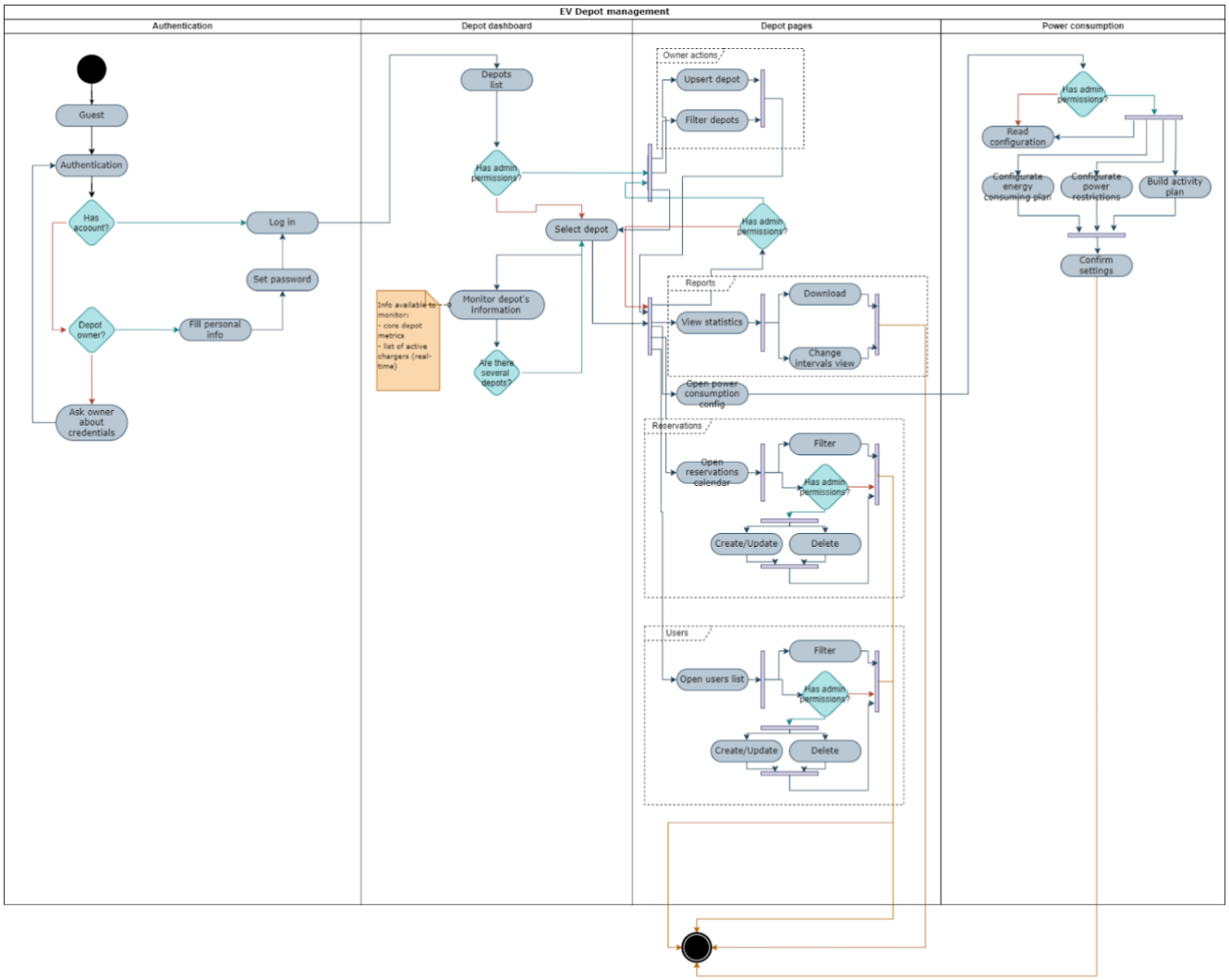


Рисунок 7 – Діаграма діяльності системи для управлінням споживання електроенергії депо електротранспорту