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 для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-8

Писаренко Є.М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Онищенко К. Г.
(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 Писаренко Єлизаветі Михайлівні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024
3. Вихідні дані до роботи Розробити програмну систему для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників, що включає в себе мобільний застосунок, веб-застосунок і IoT
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	18.04.2024	*виконано*
2	Створення специфікації ПЗ	22.04.2024	*виконано*
3	Проектування ПЗ	24.04.2024	*виконано*
4	Розробка ПЗ	01.06.2024	*виконано*
5	Тестування ПЗ	03.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	13.06.2024	*виконано*
9	Нормоконтроль, рецензування	14.06.2024	*виконано*
10	Здача роботи у електронний архів	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	17.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Писаренко Є. М.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Онищенко К. Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 63 стор., 22 рис., 34 джерел.

ПРОГРАМНА СИСТЕМА, ЦИРКАДНІ РИТМИ, ПСИХОЕМОЦІЙНИЙ СТАН, PERL, C#, TENSORFLOW.JS, IOT, REACT NATIVE, NODE.JS

Об’єкт розробки – програмна система для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників.

Мета розробки – створення програмної системи для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників, яке забезпечить не лише ефективне розподілення завдань та ресурсів, але й сприятиме підвищенню загального самопочуття та продуктивності працівників

Метод рішення – середовище розробки VS Code та Arduino IDE, мови програмування JS та C#.

У результаті розробки розробленопрограмну систему для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників.

SOFTWARE SYSTEM, CIRCADAN RHYTHMS, PSYCHOEMOTIONAL STAND, PERL, C#, TENSORFLOW.JS, IOT, REACT NATIVE, NODE.JS

The object of development is a software system for organizing work considering circadian rhythms and the psycho-emotional state of employees.

The purpose of the development is to create a software system for organizing work considering circadian rhythms and the psych emotional state of employees, which will ensure not only the effective distribution of tasks and resources, but also contribute to increasing the general well-being and productivity of employees.

The solution method is VS Code development environment and Arduino IDE, JS and C# programming languages.

As a result of the development, a software system was created for organizing work taking into account circadian rhythms and the psycho-emotional state of employees.

Я, Писаренко Єлизавета Михайлівна, студент гр. ПЗПІ-20-8, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників», 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

Темою кваліфікаційної роботи є програмна система для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників. Основне завдання програмної системи полягає у покращенні умов праці і зменшенні ризиків для підприємств за допомогою урахування циркадних ритмів і психоемоційного стану працівників.

Метою роботи є проєктування і розробка програмної системи, що складається з мобільного додатку, веб-застосунку і IoT. Для розробки планується використовувати веб-застосунок на базі Node.js, Express, React, PostgreSQL. Для мобільного застосунку буде використано React Native, а для IoT – мову програмування C++ і додаткові бібліотеки.

В рамках даної роботи також передбачається використання передових технологій машинного навчання, зокрема TensorFlow.js, для аналізу та оптимізації робочого процесу. Це дозволить створити потужні моделі, які сприятимуть покращенню умов праці працівників та підвищенню продуктивності.

Передбачається, що API системи буде забезпечувати інтеграцію з іншими системами та додатками, що сприятиме розширенню та розвитку системи в майбутньому.

Відповідно до вибраних технологій планується використовувати середовища розробки, такі як VS Code для веб-застосунків та Arduino IDE для розробки для IoT.

При розробці системи було проведено аналіз предметної області, виявлено основні проблеми і знайти методи їх вирішення. Також були поставлені задачі, проведено UML-проєктування і розроблено структури зберігання даних. Програмну систему було розроблено і детально протестовано.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Досить часто люди на роботі стикаються з перевтомою, вигорянням і проблемами зі сном. У опитуванні Deloitte Global 2022 Gen Z та Millennial Survey половина респондентів шукали нову роботу, а 25% повідомили, що звільнилися через стрес, який це спричинило. 54% повідомляють, що стрес від роботи негативно вплинув на їх життя. 35% сказали, що вони вийшли з себе на роботі 39% змушені були взяти незаплановану відпустку. Більше половини повідомили, що стрес негативно впливає на їхню здатність спати.

Крім того, 25% респондентів відчули погіршення якості роботи через стрес [1].

У 2021 році 44% людей із приблизно 100 країн зазнали професійного вигорання. Ця цифра зросла з 39% у 2019 році [2].

Синдром вигорання становить 8% усіх випадків професійних захворювань (захворювання, викликане впливом роботи) [3].

33% співробітників кажуть, що вони менш зосереджені на роботі внаслідок вигорання, тоді як 31% повідомляють про втрату інтересу до роботи, а 21% повідомляють про збільшення прокрастинації [4].

Принаймні 79% співробітників Великобританії відчувають виснаження, а близько 35% повідомляють про екстремальний або високий рівень вигорання.

Основновними причинами вигоряння є: відсутність балансу між особистим життям і роботою, неможливість відірватися з роботи під час відпустки чи поза робочим часом, невизначені терміни виконання задач, невиправдані часові обмеження у роботі, страх перед звільненням, частина працівників хвилюється через незавершену роботу, понаднормова праця (особливо це стосується молодих працівників, 27% 16-24-річних і 19% 25-34-річних працюють щонайменше 5 годин на тиждень понад договірні години [5]).

Жінки частіше, ніж чоловіки, страждають від синдрому вигорання. Вони також частіше, ніж чоловіки (54% проти 35%), зверталися за лікарняними відпустками. Це може бути через існуючу стигму щодо психічного здоров’я

чоловіків на робочому місці, тобто чоловіки рідше говорять про своє психічне здоров’я та шукають підтримки [6]. Це можна вважати більш культурно-соціальним явищем, ніж дискримінацією жінок на робочому місці. Чоловіки значно менше звертаються за психологічною допомогою, але значно частіше завершують життя самогубством. Хоча іншого боку жінки стикаються з так званими «другою зміною» і «третьою зміною», де на жінок покладається обов'язок виконувати низку неоплачуваної роботи (забезпечувати побут) – друга зміна, та нести когнітивне навантаження за здоров’я всієї сім'ї і виховання дітей. Звичайно це впливає на стомлюваність і ефективність на роботі в негативному ключі [7].

Незважаючи на те, що професійне вигорання вважається небезпекою, є багато неробочих факторів, які також можуть сприяти вигорянню. 79% працівників вважають, що погане фізичне здоров’я сприяє вигоранню, тоді як 83% людей вважають причиною погані звички до сну [8].

Існує широко поширене занепокоєння тим, що середня тривалість сну зменшилася за останні 50 років, і що недостатній сон став основною проблемою охорони здоров’я. Несприятливі наслідки депривації сну мають потенційно важливі наслідки для економічної діяльності. Недостатній сон може погіршити когнітивні здібності і пластичність мозку. Це може призвести до помилок у судженні, впливаючи на організаційні можливості, а також на ризик. Позбавлення сну також може передбачити вищий рівень нещасних випадків на виробництві і більшу поширеність серцевих нападів і хронічних захворювань. Проте, незважаючи на такі згубні наслідки, мало уваги приділено економічним наслідкам депривації сну, особливо її впливу на показники ринку праці.

Щоб оцінити причинно-наслідковий вплив сну на ефективність роботи, важливо контролювати індивідуальну неоднорідність режиму сну, генетичну схильність до часу сну або здатність для боротьби з депривацією сну, яка, ймовірно, корелює як з тривалістю сну, так і з результатами на ринку праці. Хоча деякі з цих основоположних факторів можуть змінюватися з часом, вони, ймовірно, будуть незмінними для окремих людей. Тому для того, щоб мати справу з такими пропущеними змінними, важливо покладатися на лонгітюдні дані

та включати окремі фіксовані ефекти для оцінки причинного впливу сну на продуктивність роботи. Щоб уникнути зворотного причинно-наслідкового зв’язку, поєднуємо поздовжні дані з інструментальною стратегією, яка спирається як на часові, так і на місцеві варіації часу заходу сонця, щоб вимірювати тривалість сну. Припущення, що лежить в основі цього зв’язку на першому етапі, просте: ранній захід сонця спонукає працівників лягати спати раніше, і оскільки графіки роботи не так сильно реагують на коливання часу заходу сонця, ранній час сну означає більше сну[9].

Варто зазначити, що проблеми зі сном і емоційним станом працівників впливає не тільки на їх життя, а і на збитки компаній і держав, значно підвищує ризик травмувань на робочих місцях і нещасних випадків. Від цих факторів страждають пацієнти лікарень, які потрапили до стомленого після нічної зміни лікаря, водії, які втрапили в ДТП через сонного та запрацьованого далекобійника, жертви стихійних явищ, пожеж, яким не пощастило викликати дуже втомленого або знервованого рятівника.

1.2 Виявлення та вирішення проблем

Співробітники, які борються з виснаженням, на 63% частіше візьмуть лікарняний [10].

Понад 10 мільйонів співробітників беруть відпустку через виснаження, при цьому компанії втрачають понад 80 мільйонів годин на рік через лікарняні [11].

Вигорання коштує компаніям Великобританії понад 700 мільйонів фунтів стерлінгів щороку через те, що працівники телефонують хворими з ознаками стресу та виснаження [11].

Щорічні витрати на охорону здоров’я на вигорання на робочому місці становлять від 125 до 190 мільярдів доларів США [12].

Співробітники, які вигоріли, обходяться в 3400 доларів з кожних 10 000 доларів зарплати через високу плинність кадрів і нижчу продуктивність [13].

Більше половини співробітників Великобританії змінили б місце роботи в організаціях на ті, які пропонують кращу допомогу при вигорянні [14].

Малі та середні підприємства (МСП) мають величезний ризик вигорання [15].

47% працівників малого та середнього бізнесу кажуть, що працюють 4 або більше годин понаднормово щотижня, і для більш ніж половини цих працівників це неоплачувана робота.

22% співробітників мають менше 30 хвилин на обід, 19% скасували час зі своїми близькими, а 19% пропустили важливі сімейні події.

Люди, які належать до сфери охорони здоров'я (82%), сфери послуг, туризму, ресторанів (82%) та сектору будівництва та нерухомості (77%), швидше за все, повідомлятимуть про вигорання [16].

73% юристів відчувають себе вигорілими, а 27% кажуть, що відчувають себе вигорілими щодня [8, 17].

Дослідження показують, що більше половини лікарів у Великій Британії страждають від стресу, пов'язаного з роботою [18].

Ключові працівники повідомили про більше ознак вигорання, більше презентеїзму та частіше повідомляли, що вони або залишили, або планують залишити свою роботу у 2022 році [19].

Більше половини британських працівників (52%) кажуть, що не відчувають, що роботодавці підтримують їхні потреби в психічному здоров'ї [19].

36% співробітників кажуть, що в їхніх компаніях немає нічого, щоб запобігти виснаженню співробітників [8].

Лише 3 з 10 керівників докладають зусиль для боротьби з виснаженням співробітників [13].

На думку 75% менеджерів з персоналу, дозволити гібридну роботу або гнучкий графік роботи є одним із найефективніших способів уникнути вигорання на робочому місці [8].

72% штатних працівників сказали, що щорічна відпустка є ефективним способом боротьби з виснаженням. Лише третина працівників заявили, що роботодавець заохочував їх взяти всю відпустку.

Співробітники на 40% менше схильні до вигорання, якщо у них є сильний союзник на роботі [8]. Співробітники на 32% менше відчувають себе вигорілими, коли керівництво допомагає їм справлятися з робочим навантаженням [8].

Вигорання – це кумулятивний процес, який розвивається з часом. Чим довше його не лікувати, тим важчим він стає [20].

Вигорання існує набагато довше, ніж думає більшість людей. Дослідники почали зосереджуватися на цьому в середині 1970-х років. Його вперше дослідили психіатр Герберт Фройденбергер і соціальний психолог Крістіна Маслах [21].

Дослідження пов’язують вигорання з надмірним використанням технологій. Постійне перебування на зв’язку змушує відчувати, що робітники завжди повинні бути «доступними», що призводить до розмитості меж між роботою та особистим життям і підвищеного ризику вигорання [22].

Вигорання на робочому місці впливає не лише на час неспання. Одним із серйозних і виснажливих побічних ефектів може бути безсоння [23].

Загалом, враховуючи ці факти, можна зробити висновок про потребу необхідного вивчення та впровадження стратегій, спрямованих на підтримку психічного та фізичного здоров’я працівників на робочому місці. Ефективне управління робочим часом, підтримка балансу між роботою та особистим життям, а також сприяння здоровому сну можуть значно підвищити якість життя працівників і продуктивність праці, що, у своєму разі, позитивно вплине на діяльність компаній та загального економічного стану.

У сучасному світі, де темп життя постійно зростає, важливість балансу між професійною діяльністю та особистим життям стає все більш актуальною. Організація робочого процесу, яка враховує не лише професійні завдання, але й фізіологічні та емоційні потреби співробітників, може стати вирішальним фактором для досягнення успіху як підприємства, так і його працівників [24].

У цьому контексті виникає потреба у створенні інноваційного програмного забезпечення, яке сприяло б оптимізації робочих процесів з урахуванням циркадних ритмів та психоемоційного стану співробітників. [25].

Організації, які активно застосовують циркадне вирівнювання на робочих місцях, відчувають значні переваги, включаючи підвищення задоволеності

працівників, покращення концентрації та творчих здібностей, а також зменшення кількості прогулів і проблем зі здоров'ям. Використання наукових підходів хронобіології дозволяє організаціям створити культуру добробуту та високої продуктивності, що призводить до загального підвищення ефективності та успіху [26, 27].

Проте реалізація стратегій циркадного вирівнювання супроводжується певними викликами. Ці виклики включають задоволення різноманітних потреб співробітників та управління часовими поясами для глобальних команд. Однак, завдяки ретельному аналізу цих проблем та прийняттю обґрунтованих рішень, організації можуть впроваджувати інновації та адаптуватися, створюючи робоче середовище, де пріоритетом є здоров'я та продуктивність працівників.

Таким чином, результати аналізу медичних досліджень свідчать про потенційну ефективність використання програмного забезпечення для поліпшення організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників.

Такий продукт міг би розглядати різні аспекти робочого процесу, такі як графік роботи, тривалість робочих змін, регулярність перерв, а також враховувати індивідуальні особливості сну та робочої продуктивності кожного працівника. Він міг би аналізувати дані про сон і робочу активність, щоб рекомендувати оптимальний графік сну та роботи для кожного користувача.

Крім того, програмний продукт може включати інструменти для моніторингу рівня стресу та втоми, а також надавати корисні поради та вправи для розрядки та підвищення енергії. Такий підхід дозволяє підприємству створити більш сприятливе робоче середовище, що сприятиме здоров’ю та ефективності працівників.

Остаточно, програмне забезпечення, спрямоване на підтримку ритмів та психоемоційного стану працівників, може стати інструментом для підвищення якості життя та продуктивності працівників, а також для підвищення конкурентоспроможності підприємств на ринку.

1.3 Постановка задачі

Дослідження підтверджують, що впровадження циркадного вирівнювання на робочих місцях приносить значні переваги для організацій. Враховуючи наукові підходи хронобіології, можна створити сприятливу культуру для працівників, що сприяє підвищенню їхнього задоволення, концентрації та креативності, а також зменшенню прогулів та проблем зі здоров'ям.

Вирішення викликів, пов'язаних з впровадженням циркадного вирівнювання, вимагає уваги до різноманітних потреб співробітників та керування часовими поясами для глобальних команд. Проте, при належному аналізі цих проблем і в житті обґрунтованих заходів, організації можуть досягти інновацій та адаптуватися, створюючи сприятливе робоче середовище, де пріоритетом є здоров'я та продуктивність працівників.

Успішне впровадження програмного забезпечення залежить від активності користувачів, правильного підключення та функціонування IoT-пристроїв, а також від коректної обробки та аналізу зібраних даних. Такий підхід може значно підвищити ефективність роботи організацій та сприяти їхньому успіху в довгостроковій перспективі.

Для збору даних про циркадні ритми та психоемоційний стан працівників будуть використовуватися різноманітні IoT пристрої, такі як датчики освітлення BH1750, датчики температури та вологості DHT22, а також модулі вимірювання рівня шуму KY-037. Інтеграція цих пристроїв буде здійснюватися з використанням технології ESP8266 для бездротового зв'язку та передачі даних до серверної частини програмного забезпечення [25].

Машинне навчання для аналізу даних: для аналізу даних про циркадні ритми та психоемоційний стан працівників будуть використані алгоритми машинного навчання, такі як класифікація, кластеризація та прогнозування. Моделі машинного навчання будуть навчені на великій кількості даних, зібраних від IoT пристроїв і суб’єктивних оцінок користувачів, та використовуватимуться для ідентифікації паттернів у цих даних, що вказують на стан працівників [27].

Мобільний додаток для співробітників дозволить їм відзначати свій настрій, рівень стресу та якість сну. Ці дані будуть використовуватися для уточнення та

підтримки аналітичних результатів, отриманих від IoT пристроїв та алгоритмів машинного навчання.

Конфігурація та моніторинг: система надасть можливість адміністраторам налаштовувати оптимальні варіанти робочого часу та умови праці для співробітників на основі отриманих аналітичних даних. Крім того, ПЗ буде включати засоби моніторингу та звітності для оцінки продуктивності та внесення необхідних змін.

Після розробки системи буде проведено тестування для перевірки її функціональності та надійності. Оцінка результатів включатиме аналіз якості аналітичних даних, точності рекомендацій та загального впливу системи на ефективність та благополуччя працівників.

Важливо підкреслити, що успішне впровадження програмного забезпечення залежить від декількох ключових факторів. Перш за все, необхідно, щоб користувачі активно використовували систему, передавали дані про свою діяльність, сон та психоемоційний стан. Крім того, належне підключення та функціонування пристроїв IoT для збору даних є критично важливим. Встановлення та використання мобільного додатка є ще однією важливою залежністю, оскільки саме через нього здійснюється збір даних про щоденне самопочуття працівників.

Коректна обробка та аналіз зібраних даних є передумовою для надання оптимальних рекомендацій власнику компанії або адміністратору з метою підвищення ефективності роботи. Доступ до інформації про циркадні ритми, активність, психічний стан та інші параметри працівників є критично важливим для ефективної роботи програмного забезпечення, оскільки аналіз та оцінка стану працівників буде здійснюватися за допомогою машинного навчання.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Розроблений продукт може бути частиною іншої системи, оскільки він має відкриті API. Продукт складається з мобільного додатку, сторони клієнта, сторони сервера та пристрою IoT. Спеціальний веб-додаток як інтерфейсний клієнтський додаток, серверний веб-додаток як серверна частина, яка реалізує API для зв’язку з клієнтом, база даних для зберігання даних. Продукт не має аналогів, оскільки орієнтований не на покращення якості життя конкретного користувача, а на покращення умов праці кожного, хто користується продуктом.

Основний функціонал включає збір даних з підключених пристроїв IoT, аналіз добових ритмів і психоемоційного стану співробітників за допомогою алгоритмів машинного навчання, мобільний додаток для відмітки настрою співробітників, надання рекомендацій адміністратору з організації роботи, можливість налаштувати оптимальні параметри роботи, а також моніторинг і звітність для оцінки продуктивності та внесення змін. Аналіз та оцінка стану співробітників буде здійснюватися за допомогою машинного навчання.

У системі передбачено три типи користувачів - працівник, власник і адміністратор. Адміністратор має найбільше можливостей для управління системою і є внутрішнім співробітником, який забезпечує коректну роботу програмної системи, власник – це особа, яка користується послугами для покращення умов праці своїх підлеглих та підвищення їх ефективності. І працівники, які можуть вводити та редагувати інформацію про стан, що покращить умови їх праці.

Розробка та розробка системи обмежені декількома ключовими принципами, які впливають на функціональність та використання системи. Ці обмеження є вирішальними у формуванні меж системи та керуванні процесом розвитку. Вони включають:

Користувачі зобов’язані здійснювати платежі перед використанням системи, встановлюючи фінансові обмеження на доступ користувачів.

Система покладається на стабільне та надійне підключення до Інтернету для безперебійного обміну даними між мобільними додатками, веб-додатками та пристроями IoT.

Для використання системи користувачі повинні мати доступ до мобільних пристроїв із встановленим мобільним додатком або комп’ютера з підтримкою веб-додатків.

Веб-додаток служить шлюзом для керування та адміністрування даних у системі, встановлюючи функціональні обмеження.

Дизайн системи має відповідати географічним і юрисдикційним нормам, забезпечуючи дотримання правових вимог у різних регіонах.

Дотримання відповідних положень щодо захисту даних і конфіденційності є обов’язковим для забезпечення безпеки та конфіденційності даних користувача. Це обмеження керує методами обробки та зберігання даних.

Компонент мобільного додатку має бути сумісний з основними платформами (iOS, Android). Під час проектування та розробки розробники повинні дотримуватися вказівок і обмежень, що стосуються платформи.

Користувачі будуть активно використовувати програмну систему.

Пристрої IoT будуть належним чином підключені та передаватимуть дані про інтенсивність світла, температуру, вологість, шум.

Мобільний додаток буде встановлюватися на пристрої користувачів, активно використовуватися для передачі даних про їх повсякденну діяльність.

Власник компанії отримає оптимальні рекомендації на основі зібраних даних і використає їх для підвищення ефективності роботи. Успішне впровадження програмної системи залежить від функціонування та належного підключення пристроїв IoT для збору даних.

Встановлення та використання мобільного додатку мають вирішальне значення для збору даних про щоденну діяльність співробітників.

Правильна обробка та аналіз зібраних даних є передумовами для надання оптимальних рекомендацій власнику компанії.

Доступ до інформації про циркадні ритми, активність, психічний стан та інші параметри співробітників необхідний для ефективного функціонування програмної системи, враховуючи, що аналіз і оцінка стану співробітників буде проводитися за допомогою машинного навчання.

Інтерфейс користувача має бути інтуїтивно зрозумілим, зручним і доступним як на мобільних пристроях, так і на комп'ютерах. Він повинен сприяти легкому введенню даних, візуалізації та взаємодії з функціями системи. Дизайн має відповідати галузевим стандартам для бездоганної взаємодії з користувачем.

Система містить пристрої IoT, оснащені датчиками для збору даних, пов'язаних із циркадними ритмами, активністю, інтенсивністю світла, температурою, вологістю та шумом. Ці пристрої мають бути сумісні зі специфікаціями системи, щоб забезпечити точну та надійну передачу даних.

Система розроблена для роботи на поширених мобільних платформах, включаючи iOS та Android. Користувачі повинні мати можливість встановити необхідний мобільний додаток на свої пристрої для активної участі в передачі даних щодо себе.

Програмне забезпечення має бездоганно інтегруватися з указаним стеком технологій (PERN: PostgreSQL, Express.js, React, Node.js). Сумісність із відповідними базами даних, веб-серверами та іншими компонентами програмного забезпечення є важливою для ефективної роботи. Система має відповідати стандартам API для плавної інтеграції з іншими системами.

Комунікаційні інтерфейси системи мають забезпечувати безпечний та ефективний обмін даними між мобільними додатками, веб-додатками та пристроями IoT. Протоколи для передачі даних мають відповідати галузевим стандартам, а для захисту конфіденційної інформації під час зв'язку слід застосовувати засоби шифрування. Система повинна підтримувати надійне підключення до Інтернету для безперервного обміну даними.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Наступним кроком проектування системи було вирішено провести розробку діаграми варіантів використання системи. Найкращим варіантом візуалізації такої частини моделювання системи виявилась діаграма Use Case стандарту проектування UML. Було розроблено дві діаграми Use Case, а саме для мобільного і web-застосунків(див. рис. 3.1 – 3.2).

Все UML проєктування було виконано у draw.io – це онлайн-додаток для створення діаграм, який дозволяє створювати блок-схеми, UML–діаграми, діаграми сутність-зв'язок, мережеві діаграми, макети та інші типи діаграм.

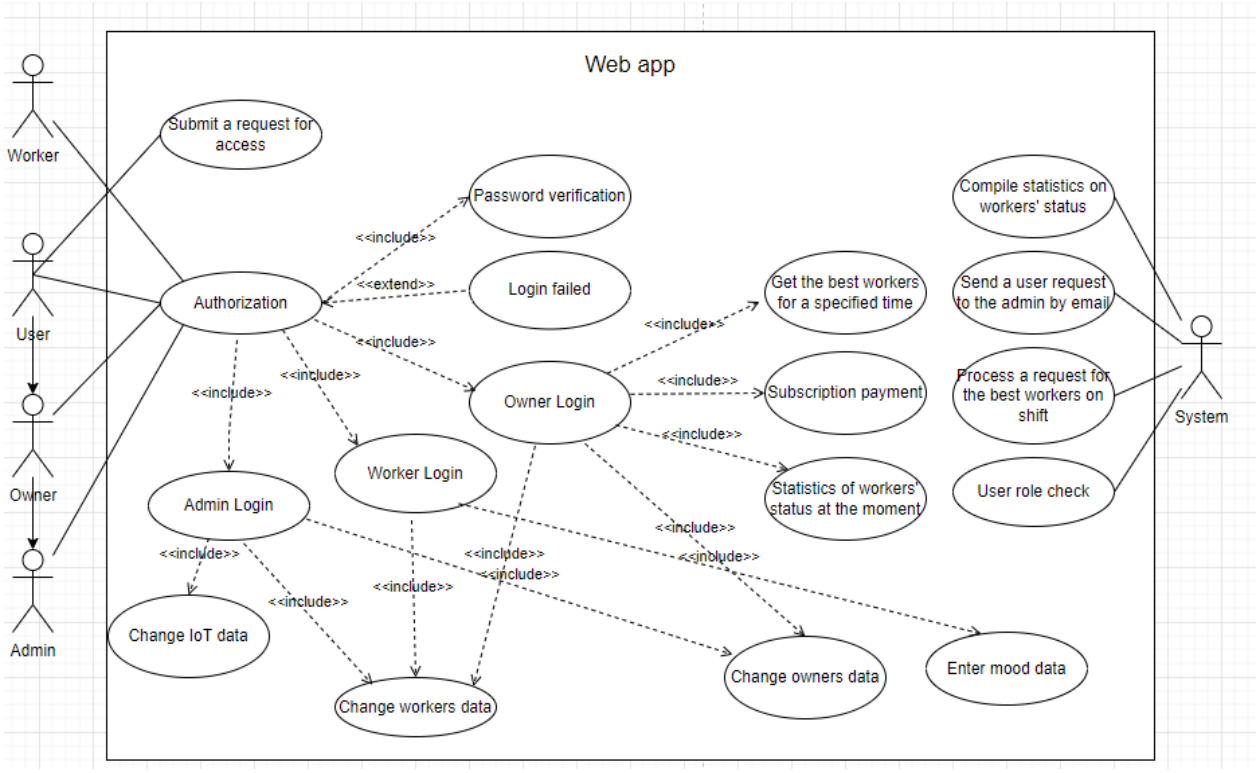


Рисунок 3.1 – Use Case діаграма для web-застосунку (виконано самостійно)

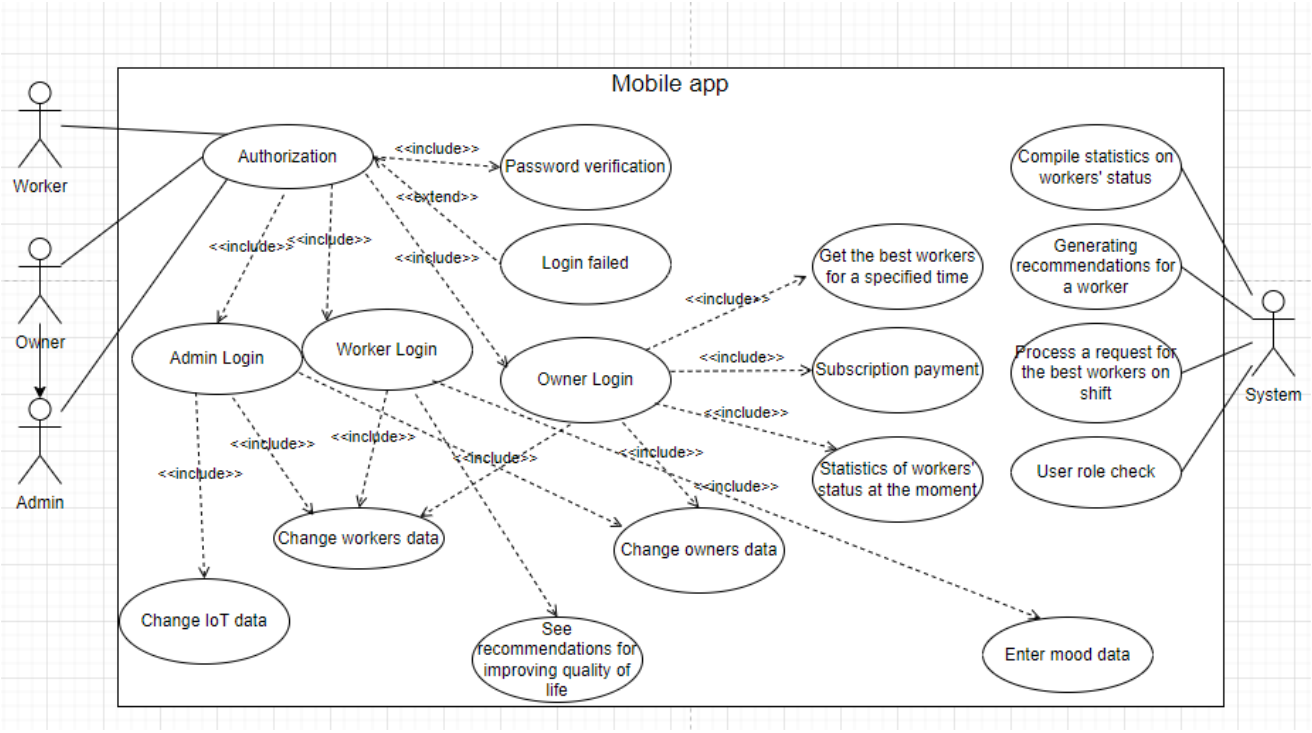


Рисунок 3.2 – Use Case діаграма для мобільного застосунку (виконано самостійно)

На рисунках детально зображена взаємодія акторів (адміністратор, власник і робітник) з системою. Діаграми досить чітко передають важливу відмінність роботи мобільного застосунку і сайту, а саме в мобільному застосунку є можливість отримувати рекомендації від системи з приводу покращення умов сну і відпочинку для робітників. Це було розроблено задля зручності використання ПЗ поза робочим місцем і враховуючи специфіку професій, для яких цей застосунок є найбільш актуальним.

Також було вирішено розробити діаграми послідовностей, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень. Для наочності було обрано 2 сценарії використання системи, web-застосунок, яким користується власник і мобільний застосунок, яким користується робітник (див. рис. 3.3–3.4).

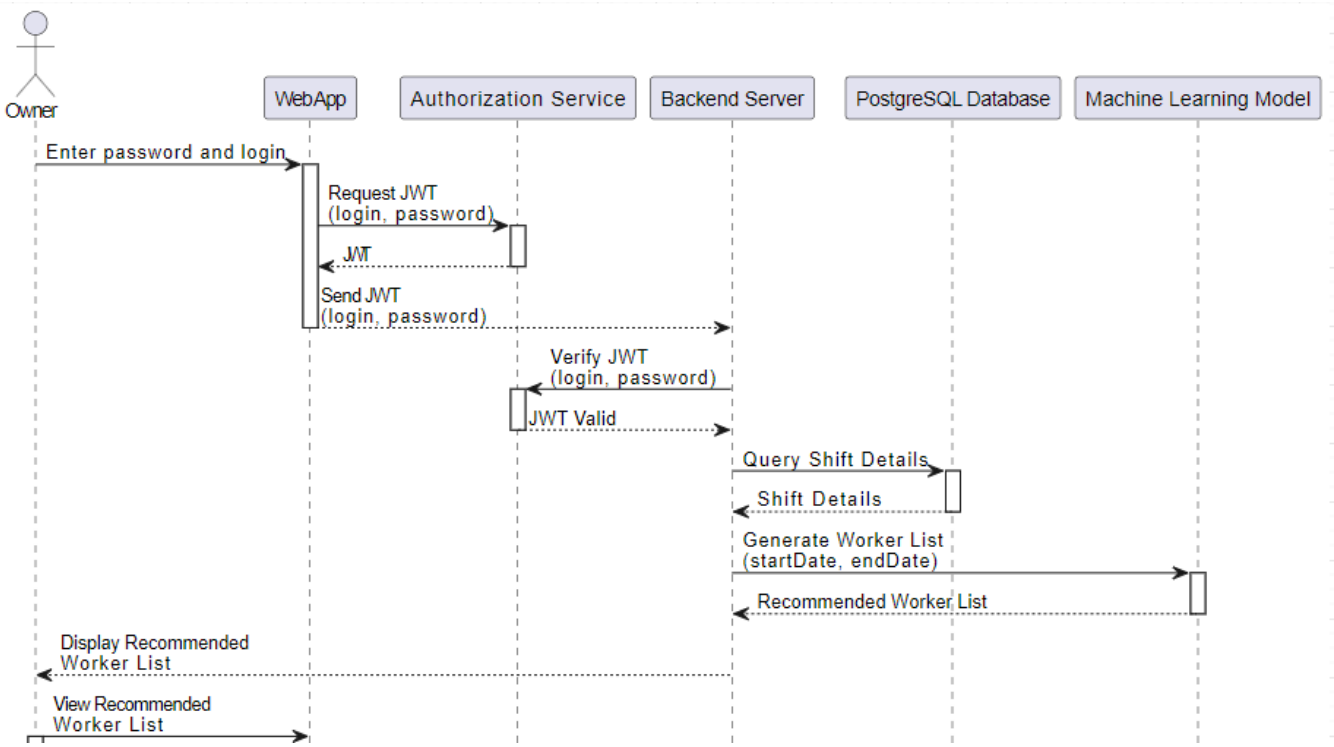


Рисунок 3.3 – Діаграма послідовності для власника (виконано самостійно)

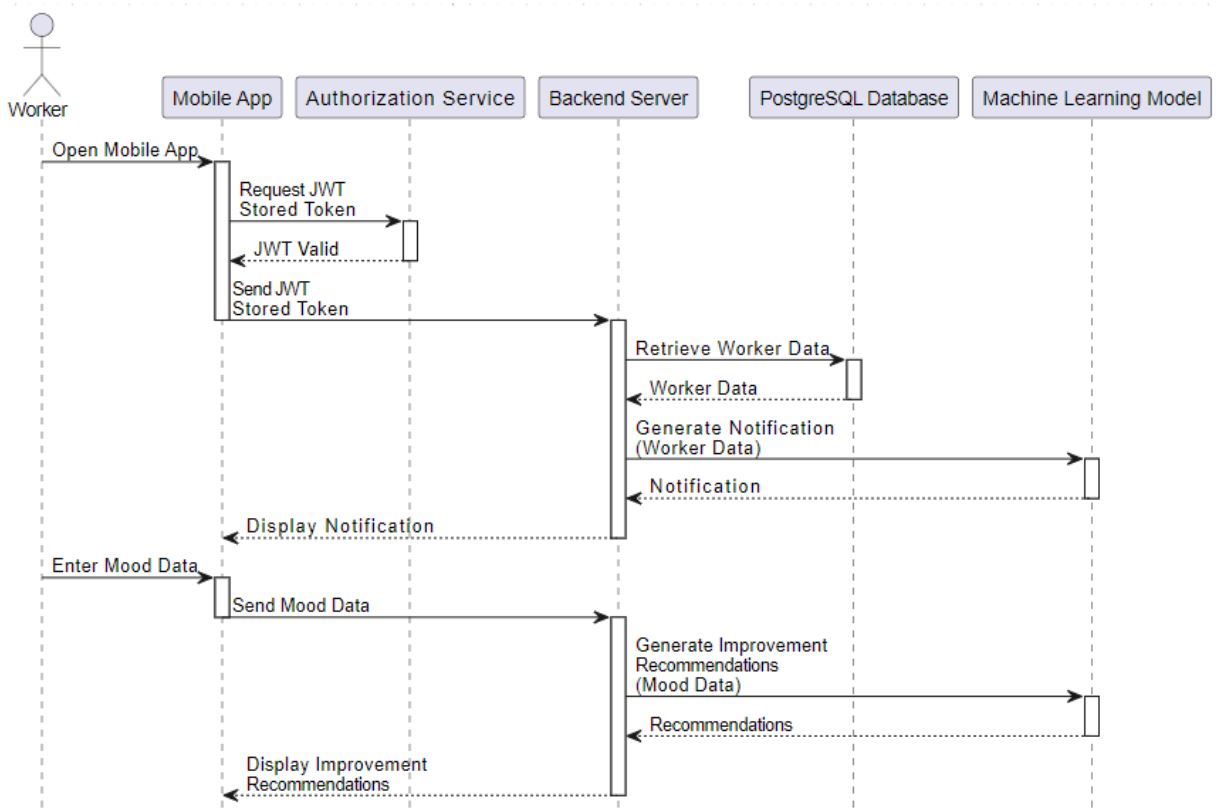


Рисунок 3.4 – Діаграма послідовності для працівника (виконано самостійно)

На діаграмах зображено процес авторизації для власника і працівника за допомогою токена. На рисунку 3.3 показано, як працює функція, що виводить список найбільш працездатних на поточний час робітників (враховуючи дані про

рівень стресу, якість сну і втомленість), також за допомогою машинного навчання система буде аналізувати коментарі, які залишили працівники. Наприклад, якщо працівник написав в цей день, що в нього вихідні, відпустка, або лікарняний, його не буде відображати в цей список.

На діаграмі послідовності для працівника зображено процес авторизації через токен і генерація повідомлень за допомогою машинного навчання, які працюватимуть, як рекомендації з приводу покращення самопочуття, враховуючи дані про його стан за останній час. Далі зображено процес надання працівником даних про його стан і обробка їх на серверній частині.

Було розроблено діаграму переходу станів для візуалізації подій, які відбуваються під час користування системою власником (див. рис. 3.5)

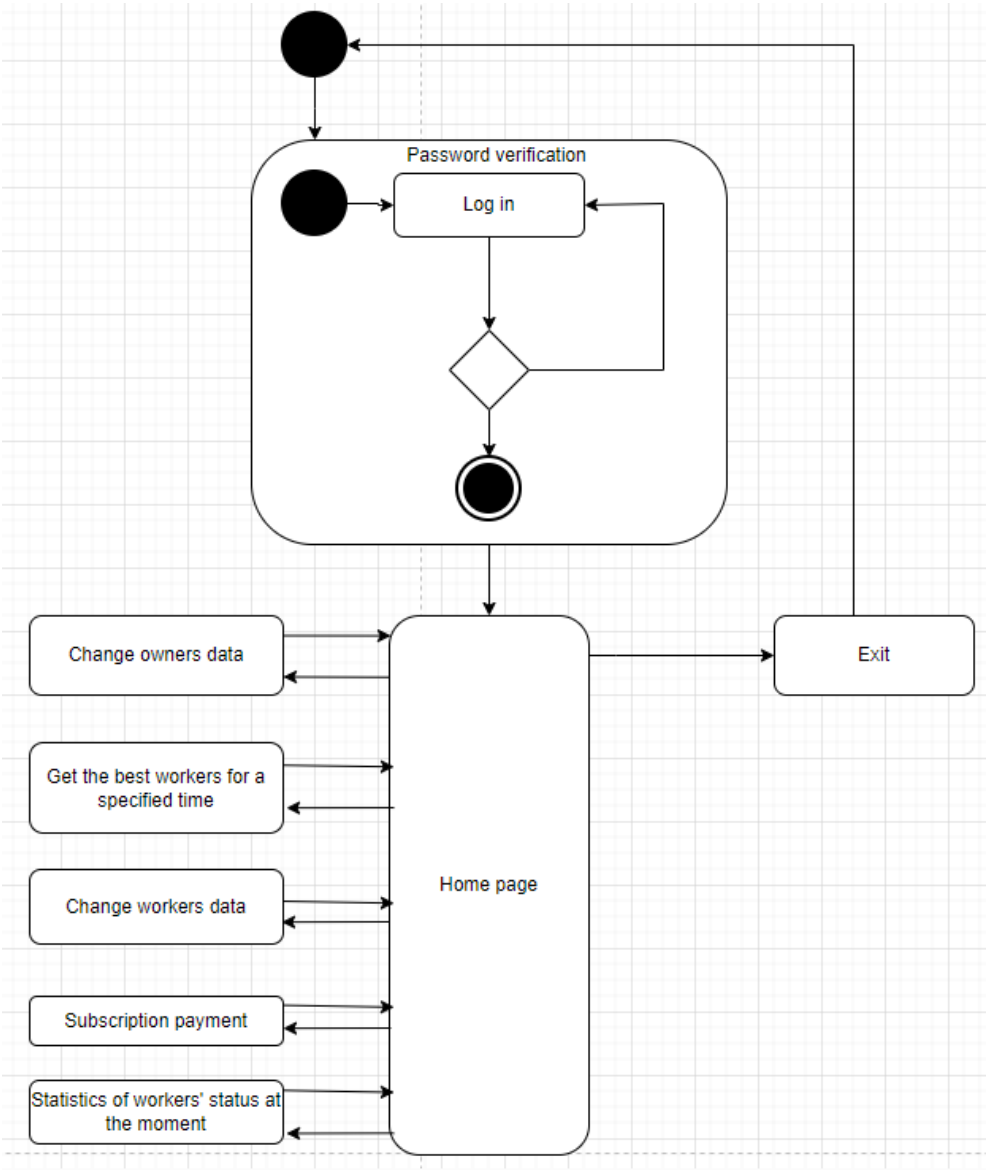


Рисунок 3.5 – Діаграма переходу станів для власника (виконано самостійно)

Діаграма зображує взаємодію клієнтської частини з користувачем, а саме вхід у систему і функції, які розміщені на домашній сторінці власника. Користувач має можливість увійти на свою сторінку змінити дані працівників, отримати дані про працівників і рекомендації на певний час чи на поточний момент, що допоможе оптимізувати робочий графік наперед, чи знайти найкращий варіант у непередбачуваній ситуації. Також користувач з рівнем доступу «власник» має можливість оплатити підписку за користування програмною системою, відредагувати дані про себе і вийти з застосунку.

У ході проєктування ПЗ було розроблено діаграму потоків даних для користувача з рівнем доступу «адміністратор»(див. рис. 3.6)

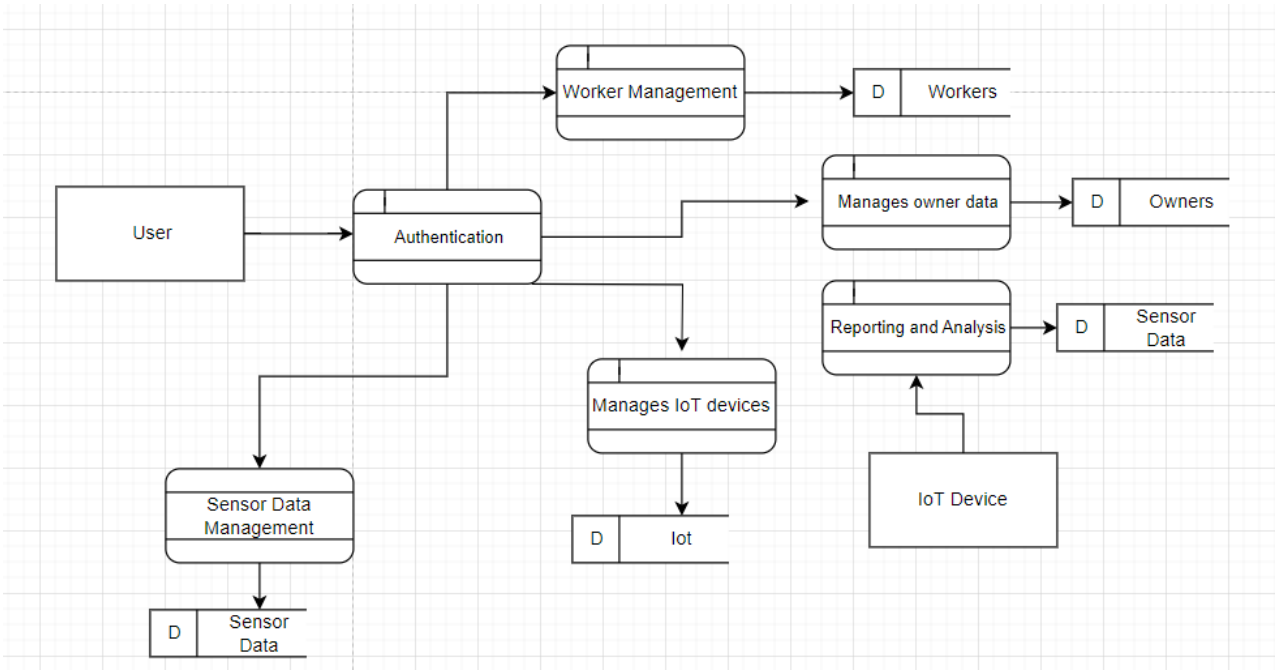


Рисунок 3.6 – Діаграма потоків даних для адміністратора (виконано самостійно)

Для цієї діаграми було обрано роль адміністратора так, як вона надає можливість керувати найбільшим об’ємом інформації у базі даних. Також окремо на діаграмі зображено процес передачі даних з IoT пристрою у БД.

3.2 Проєктування архітектури ПЗ

Для розробки серверної складової програмної системи було обрано платформу Node.js, що є середовищем виконання JavaScript. Node.js базується на рушії JavaScript Chrome V8, яке трансформує виклики на мові JavaScript в машинний код, не вимагаючи їх інтерпретації.

Обираючи Node.js, було враховано ряд факторів. По-перше, його пакетна екосистема npm є найбільшою у світі з бібліотеками відкритого коду. npm надає різноманіття пакетів, які можна легко встановити в проект, сприяючи роботі та допомагаючи швидше та ефективніше вирішувати типові проблеми.

Крім того, для побудови веб-сервера та обробки маршрутів було використано фреймворк Express.js, який добре працює в пари з Node.js. Він дозволяє швидко створити потужні та ефективні веб-застосунки.

Для забезпечення безпеки та автентифікації було використано JWT (JSON Web Tokens). JWT є стандартом для створення токенів доступу, які містять інформацію про користувача та його права. Вони дозволяють використовувати автентифікацію та авторизацію користувачів, а також забезпечити захист даних у системі.

Для вирішення завдань машинного навчання використовується TensorFlow.js. Ця бібліотека дозволяє навчати та виконувати інтерференції моделей машинного навчання прямо в браузері або на сервері Node.js, використовуючи JavaScript. TensorFlow.js надає альтернативний інструментарій для розробки та розгортання різноманітних моделей, що дозволяє ефективно використовувати машинне навчання в цьому проекті.

Для клієнтської частини програмної системи використовується бібліотека React.js, що дозволяє розробникам створювати інтерактивні користувальні інтерфейси.

Мобільний додаток для Android розробляються з використанням React Native.

Для системи управління базами даних (СУБД) використовується PostgreSQL, що забезпечує надійне та ефективне зберігання даних. Для безпечного та ефективного обміну даними між компонентами системи використовують галузеві стандарти протоколів передачі даних.

Пристрої IoT передають дані про циркадні ритми, активність, інтенсивність світла, температуру, вологість та шум на сервері за допомогою API.

Серверна частина отримує дані від пристроїв IoT, зберігає їх у базі даних та надає доступ до цих даних мобільним та веб-додаткам через API. Користувачі

можуть відстежувати дані про своє здоров’я та активність, які підтримуються з пристроїв IoT. Крім того, користувачі можуть встановити мобільний додаток на своїх пристроях для активної участі в передачі даних про себе та взаємодії з системою.

У якості сервера для зберігання та обробки даних тимчасово використовується AWS, що дає безліч переваг, основною з яких є безкоштовний доступ.

Система відповідає стандартам безпеки та ефективності для забезпечення надійного обміну даними та захисту конфіденційної інформації. Інтеграція зі стеком технологій (PERN: PostgreSQL, Express.js, React, Node.js) забезпечує ефективність та гнучкість системи. API забезпечують плавну інтеграцію з іншими системами та додатками, що сприяє розширенню та розвитку системи в майбутньому.

3.3 Проєктування структури зберігання даних

Під час проєктування структури зберігання даних було розроблено ER-діаграму (див. рис. 3.7). ER діаграма допоможе уявити структуру бази даних та взаємозв’язки між сутностями, сприяючи розумінню та розробці бази даних.

Таблиця iot:

- створюється з унікальним ідентифікатором id_iot, який є основним ключем;
- ця таблиця включає поле id_owner, яке є зовнішнім ключем до таблиці owner.

Таблиця owner:

- містить інформацію про власника пристрою IoT;
- має основний ключ id_owner;
- включає поле id_iot, яке є зовнішнім ключем до таблиці iot.

Таблиця sensor_data:

- містить дані, зібрані від датчиків;
- має основний ключ id_sensor;
- включає поле id_iot, яке є зовнішнім ключем до таблиці iot.

Таблиця worker:

– зберігає інформацію про робітника, який використовує систему;

– має основний ключ id_worker;

– включає поле id_owner, яке є зовнішнім ключем до таблиці owner.

Таблиця worker_data:

– містить дані, зібрані від робітників, які використовують систему;

– має автоматично згенерований основний ключ id_worker_data;

– включає зовнішні ключі id_worker та id_sensor до відповідних таблиць.

За допомогою зовнішніх ключів у вище перелічених таблицях встановлюється зв'язок між ними, що допомагає забезпечити цілісність даних в базі даних.

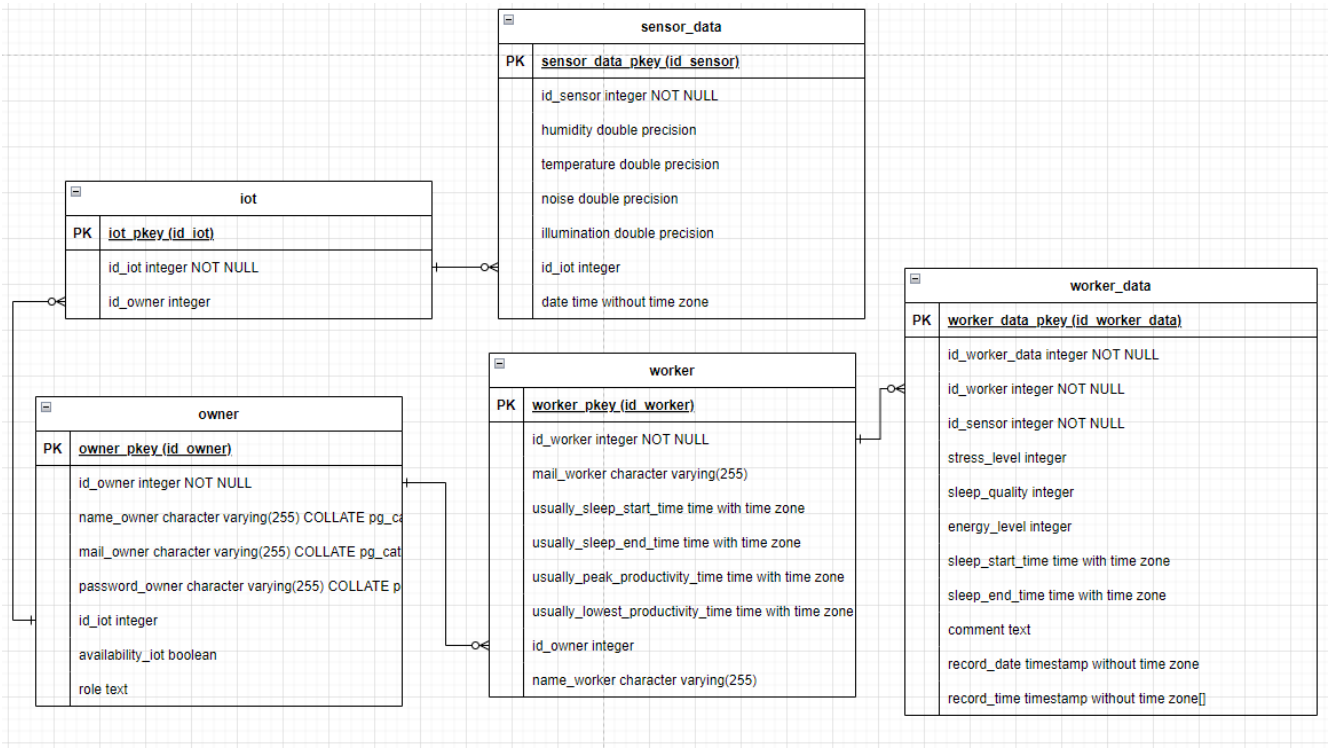


Рисунок 3.7 – ER-діаграма (виконано самостійно)

3.4 Приклади найцікавіших алгоритмів та методів

Функція №1.

Система повинна містити функцію отримання даних про стан усіх співробітників і за допомогою машинного навчання генерувати рейтинг найбільш підходящих співробітників за визначений проміжок часу.

Вхідні дані:

– дані працівника (циркадний ритм, психоемоційний стан);
– діапазон часу (початок і кінець робочої зміни.

Обробка: система буде використовувати алгоритми машинного навчання для аналізу даних співробітників і визначення їх придатності для зазначеного періоду часу з урахуванням циркадних ритмів і психоемоційного стану.

Вихідні дані: рейтинг працездатності співробітників для заданого періоду часу. Список рекомендованих працівників за результатами аналізу.

Обробка помилок: система повинна обробляти помилки, пов’язані з відсутніми або неточними даними про співробітників, забезпечуючи точну обробку та надійні рекомендації.

Функція №2.

Система повинна забезпечувати функцію аналізу даних про навколишнє середовище з пристроїв IoT (температура, інтенсивність світла та шум) і передавати персоналізовані повідомлення співробітникам про те, як покращити якість їхнього життя.

Вхідні дані:

– дані про навколишнє середовище з пристроїв IoT (температура, інтенсивність світла, шум)
– індивідуальні профілі співробітників

Система оброблятиме й аналізуватиме дані про навколишнє середовище, щоб оцінити його вплив на якість сну, надаючи персоналізовані рекомендації на основі профілю кожного працівника.

Вихідні дані: персоналізовані повідомлення з рекомендаціями щодо покращення сну.

Обробка помилок: система повинна обробляти помилки при передачі даних з пристроїв IoT і надавати точні рекомендації навіть за наявності неповних або неточних даних.

3.5 Створення UI/UX

Для створення UI/UX було використано Canva, платформу для дизайну, яка надає широкий вибір інструментів та шаблонів для швидкого створення привабливого та функціонального інтерфейсу користувача (див. рис. 3.8–3.12).

ChatGPT допоміг згенерувати підходящу і лаконічну назву.

Була розроблена головна сторінка сайту і дизайн логотипу, за допомогою logo.com (див. рис 3.8 – 3.9).

Результатом використання цих інструментів стала створення візуально привабливої та функціональної головної сторінки сайту, а також унікального логотипу, які сприяють позитивному сприйняттю користувачами та підвищують пізнаваність бренду.

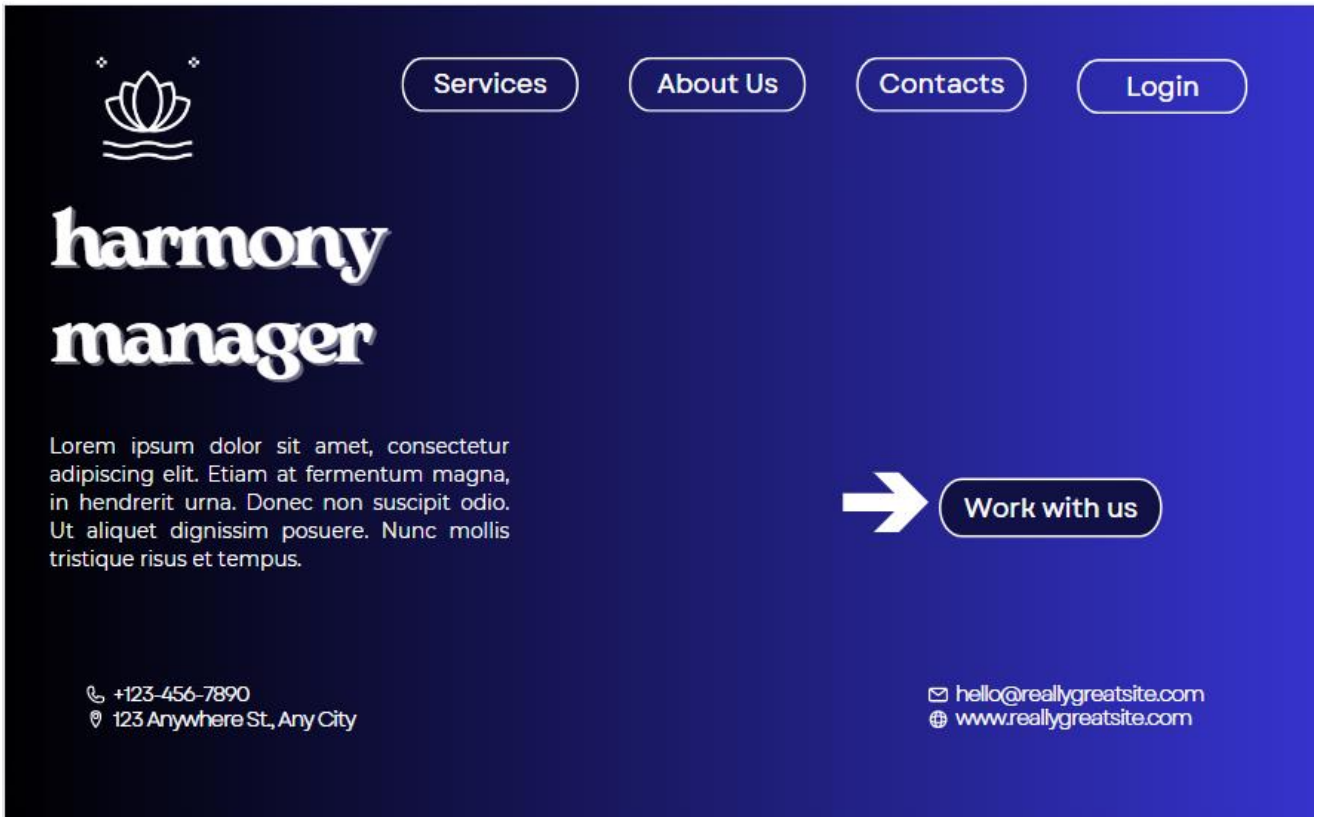


Рисунок 3.8 – Перша сторінка сайту (виконано самостійно)

Рисунок 3.9 – Перші прототипи дизайну логотипу (виконано самостійно)

Далі були розроблені основні функціональні сторінки, а саме авторизація, тарифні плани і головна сторінка користувача з рівнем доступу «власник» (див. рис. 3.10 – 3.12).

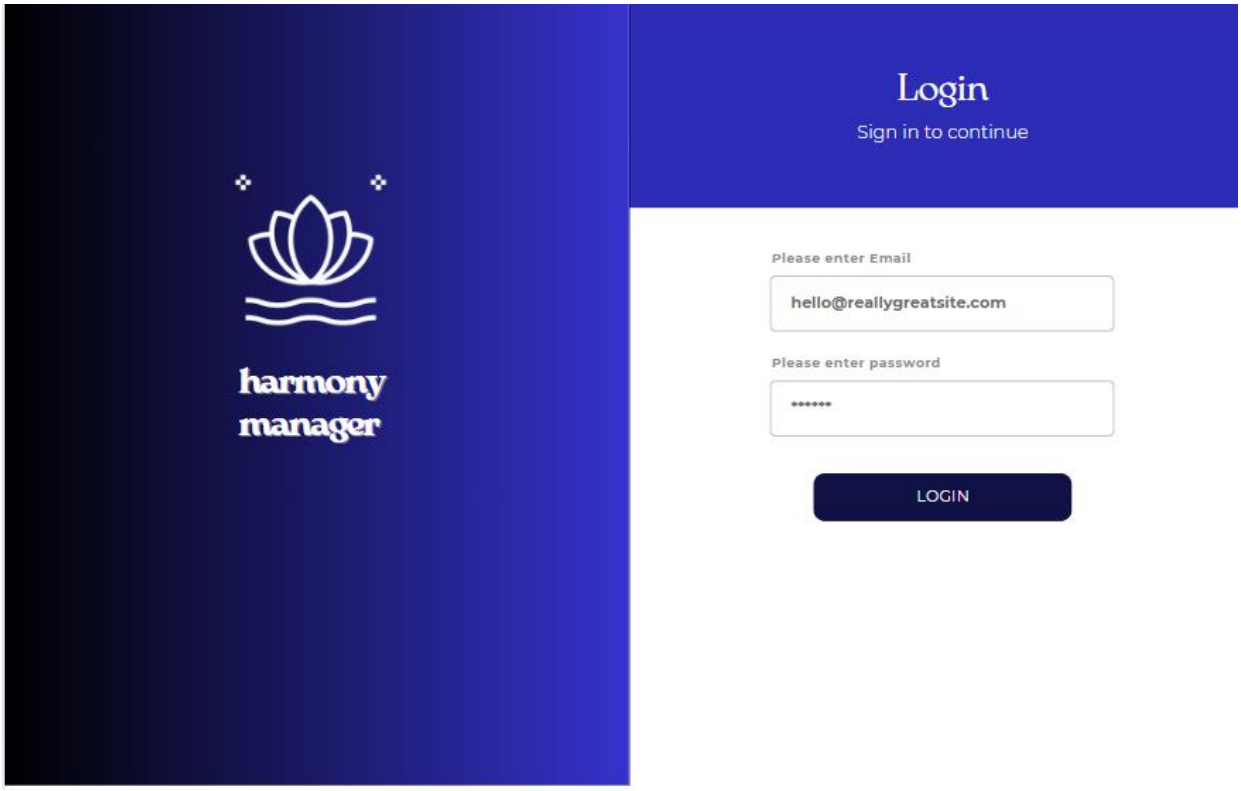


Рисунок 3.10 – Дизайн сторінки авторизації (виконано самостійно)

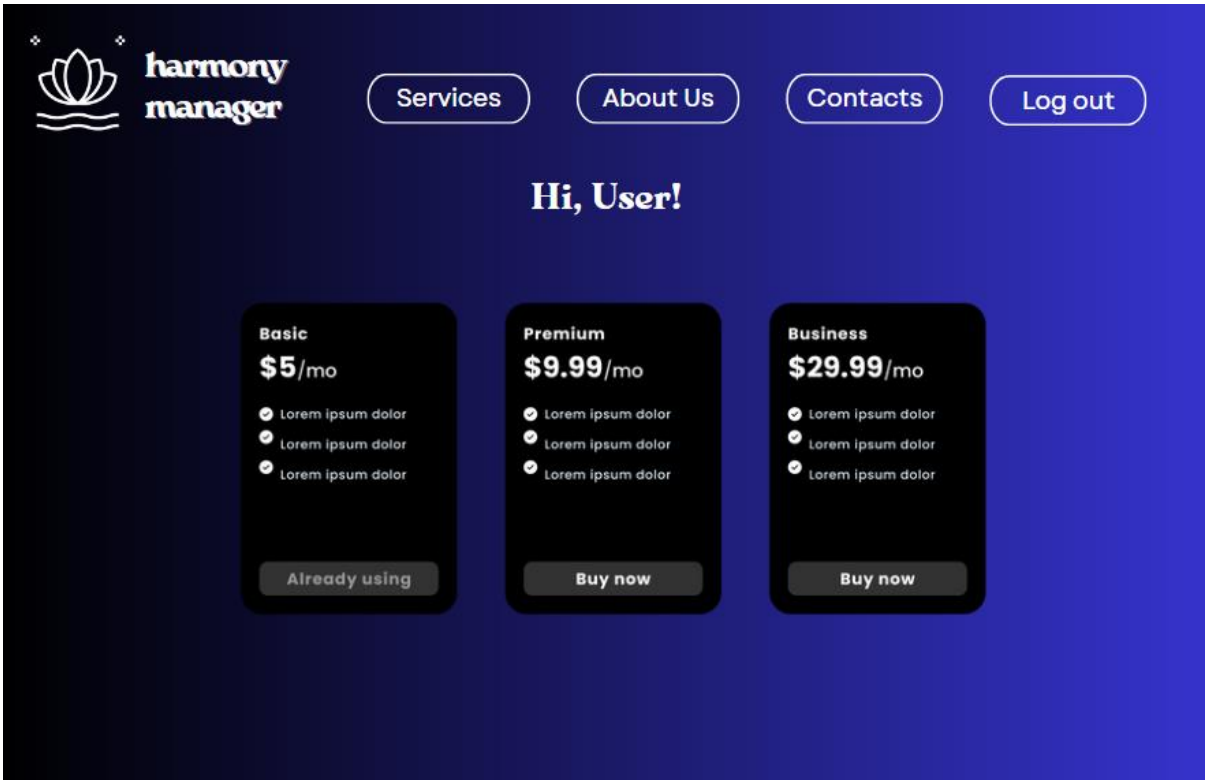


Рисунок 3.11 – Дизайн тарифного плану (виконано самостійно)

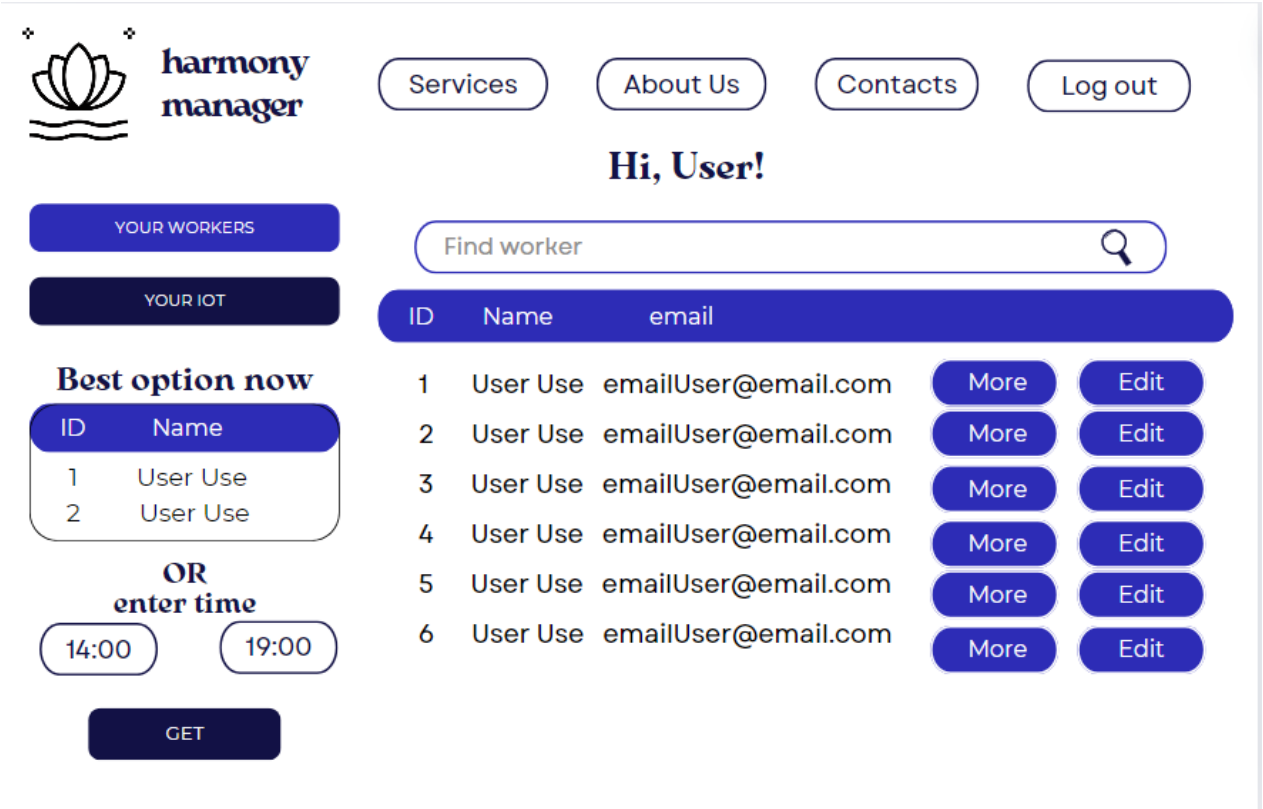


Рисунок 3.12 – Дизайн головної сторінки власника (виконано самостійно)

Також було розроблено дизайн для мобільного застосунку за допомогою Canva.

На рисунку зображені три сторінки для користувача з правами доступу «власник», а саме: сторінка авторизації, і дві сторінки, які ілюструють функціональні можливості на домашній сторінці користувача (див. рис. 3.13).

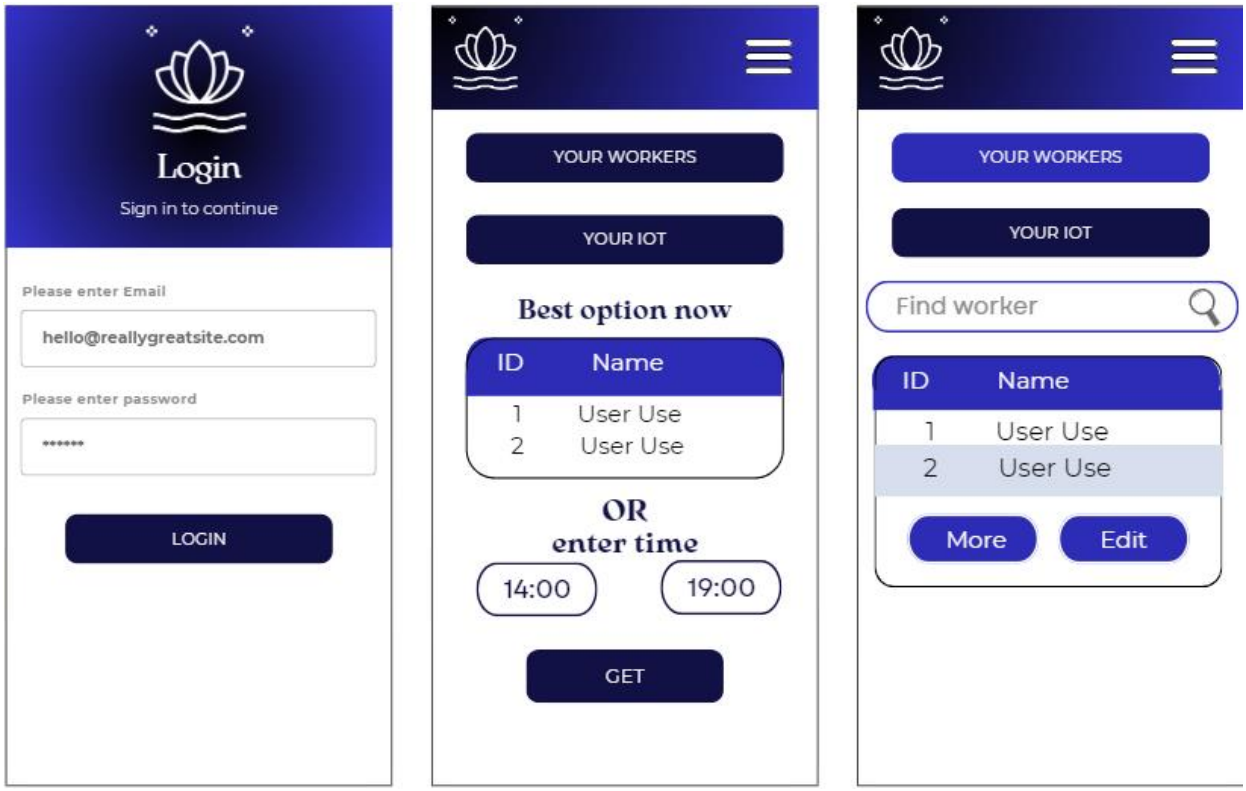


Рисунок 3.13 – Дизайн мобільного застосунку (виконано самостійно)

Було вирішено зберігати дизайн код для обох інтерфейсів, щоб користувач з легкістю міг орієнтуватися в програмній системі. Розроблений дизайн є мінімалістичним, функціональним, простим та зрозумілим. Обрані кольори і логотип асоціюються зі спокоєм і балансом, що відповідає меті проєктованої системи.

Для заголовків було обрано досить акцентний ретро шрифт Arsenica Antiqua, який звертає увагу на важливий функціонал. Основним для програмної системи є шрифт Montserrat, який добре читається.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка клієнтської частина для веб-застосунку

Проєкт побудований на основі фреймворку React, який є популярним інструментом для створення користувацьких інтерфейсів.

React дозволяє розробляти інтерфейси у вигляді компонентів, що легко повторно використовуються. Це особливо важливо для проєктів з великим і складним інтерфейсом користувача, як у випадку з системою, що розробляється [28].

React має свій мобільний аналог - React Native, що дозволяє створювати мобільні додатки з тією ж логікою, що і веб-додатки, забезпечуючи єдиний кодовий базис для різних платформ.

Структура стандартного React проєкту виглядає так:

- public/: містить статичні файли та HTML-шаблон.
- src/: містить усі файли з кодом програми.
- components/: компоненти React.
- assets/: файли зі стилями, зображеннями та шрифтами.
- services/: сервіси для роботи з API.
- utils/: утиліти та допоміжні функції.

При розробці клієнту для веб застосунку було виділено основні компоненти, а саме: підвал, шапка, налаштування, тарифи на підписку, головна сторінка, реєстрація, авторизація, модальні вікна для редагування інформації про робітників, власників, адміністраторів, початкова сторінка, сторінка сплати.

Було розроблено компонент Header (шапку сайту), яка використовується на більшості сторінок для полегшення навігації користувачів по веб-застосунку та забезпечення доступу до основних розділів системи (див. рис. 4.1).

Цей компонент забезпечує єдину точку входу для навігації по основних розділах веб-застосунку. Логотип та заголовок надають користувачам чітке уявлення про бренд, а навігаційне меню дозволяє швидко перемикатися між сторінками. Використання бібліотеки react-router-dom дозволяє організувати

маршрутизацію без перезавантаження сторінки, що підвищує швидкодію та покращує користувацький досвід.

Наступні компоненти також відіграють ключову роль у функціональності веб-застосунку:

- Footer (підвал): містить інформацію про контактні дані та посилання на політику конфіденційності та умови використання.
- Settings (налаштування): дозволяє користувачам налаштовувати персональні параметри використання системи.
- SubscriptionPlans (тарифи на підписку): відображає різні плани підписки та дозволяє користувачам вибрати відповідний план.
- HomePage (головна сторінка): забезпечує загальний огляд системи та її основні функції.
- SignUp (реєстрація): форма для створення нового облікового запису користувача.
- Login (авторизація): форма для входу до системи.
- EditModal (модальні вікна для редагування інформації): дозволяють редагувати інформацію про робітників, власників та адміністраторів.
- PaymentPage (сторінка сплати): забезпечує процес оплати та управління підписками.

Ця структура компонентів дозволяє легко розширювати функціональність веб-застосунку, підтримувати чистий та зрозумілий код.

```
const Header = () => {
  return (
    <header>
      <h1 className="title">harmony<br />manager</h1>
      <nav>
        <img src={logo} alt="Logo" className="logo" />
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/services">Services</Link></li>
          <li><Link to="/pay">Pay</Link></li>
          <li><Link to="/ singup"> Sing up</Link></li>
          <li><Link to="/login"> Log in</Link></li>
        </ul>
      </nav>
    </header>
  );
};
```

Рисунок 4.1 – Компонент Header (виконано самостійно)

Цей компонент забезпечує єдину точку входу для навігації по основних розділах веб-застосунку. Логотип та заголовок надають користувачам чітке уявлення про бренд, а навігаційне меню дозволяє швидко перемикатися між сторінками. Використання бібліотеки react-router-dom дозволяє організувати маршрутизацію без перезавантаження сторінки, що підвищує швидкодію та покращує користувацький досвід.

4.2 Розробка серверної частини

Серверна частина проєкту реалізована з використанням Node.js та Express.js, що забезпечує простоту та гнучкість у створенні RESTful API для взаємодії з клієнтською частиною. Основні функції серверної частини включають створення, читання, оновлення та видалення (CRUD) даних про робітників, власників та інші сутності [29].

Серверна частина проєкту організована наступним чином:

- db/: модулі для роботи з базою даних;
- controllers/: логіка обробки запитів;
- routes/: маршрутизація запитів;
- config/: конфігураційні файли, включаючи налаштування для аутентифікації;
- public/: статичні файли.

Контролери відповідають за обробку запитів та взаємодію з базою даних. Наприклад, контролер для робітників (workerController) реалізує основні CRUD-операції (див. Додаток А).

Маршрутизація налаштовується з використанням Express Router, що дозволяє організувати логіку обробки запитів за окремими шляхами, нижче зображено приклад коду (див. рис. 4.2).

```
const express = require('express');
const passport = require('passport');
const cors = require('cors');
const path = require('path');

const workerRouter = require('./routes/worker.routes');
const ownerRouter = require('./routes/owner.routes');
const iotRouter = require('./routes/iot.routes');
const workerdataRouter = require('./routes/workerdata.routes');
const sensordataRouter = require('./routes/sensordata.routes');

const PORT = process.env.PORT || 8080;

const app = express();
app.use(express.json());
require('./config/passport')(passport);
app.use(express.static(path.join(__dirname, 'public')));
app.use(cors({ origin: 'http://localhost:3000' }));
app.use(passport.initialize());
app.use('/api', sensordataRouter);
app.use('/api', ownerRouter);
app.use('/api', iotRouter);
app.use('/api', workerRouter);
app.use('/api', workerdataRouter);

app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Рисунок 4.2 – Маршрутизація з використанням Express Router

Для взаємодії з базою даних використовується модуль db, який забезпечує виконання SQL-запитів. Це дозволяє ефективно управляти даними та забезпечувати необхідну функціональність додатку.

Сервер використовує Middleware для обробки запитів, таких як cors для забезпечення політики CORS, passport для аутентифікації, та інші. Конфігураційні файли містять налаштування для підключення до бази даних, конфігурації Passport та інші параметри (див. рис.4.3)

```
function checkRole(role) {
    return (req, res, next) => {
        console.log('checkRole req.user:', req.user);
        if (req.user.role.includes(role)) {
            return next();
        }

        res.status(403).json({ error: 'Forbidden' });
    };
}
module.exports = { checkRole };
```

Рисунок 4.3 – Middleware для перевірки ролі корстувача (виконано самостійно)

Таким чином, серверна частина забезпечує надійну та гнучку платформу для обробки запитів клієнтської частини, управління даними та забезпечення необхідного рівня безпеки та продуктивності.

4.3 Розробка клієнтської частина для мобільного застосунку

Розробка клієнтської частини для мобільного застосунку на React Native включає створення інтерфейсу користувача, налаштування навігації, взаємодію з серверною частиною та забезпечення оптимального користувацького досвіду. Реалізація цих компонентів забезпечує ефективну та інтуїтивну взаємодію з системою для кінцевих користувачів [30].

Для розробки мобільного застосунку були обрані наступні технології та інструменти:

- React Native: основний фреймворк для розробки застосунків;
- Expo: платформа для швидкої розробки та тестування застосунків;
- React Navigation: бібліотека для маршрутизації та навігації;
- Axios: бібліотека для виконання HTTP-запитів;
- Redux: для управління станом застосунку.

Структура стандартного проєкту на React Native виглядає наступним чином:

- assets/: містить ресурси, такі як зображення, шрифти та інші медіафайли;
- components/: компоненти React Native;
- navigation/: файли, що стосуються навігації між екранами;
- redux/: управління станом застосунку;
- screens/: екрани застосунку;
- services/: сервіси для роботи з API;
- App.js: головний файл застосунку.

Основні компоненти та екрани мобільного застосунку включають:

- LoginScreen: екран для входу користувача до системи;
- HomeScreen: головний екран, що надає загальний огляд системи та основні функції;
- WorkerListScreen: екран для перегляду списку робітників;

– WorkerDetailsScreen: екран для перегляду та редагування інформації про конкретного робітника;
– SettingsScreen: екран для налаштувань користувача.

Однією з важливих функціональних можливостей застосунку є інтеграція з TensorFlow.js для надання рекомендацій користувачам. Наприклад, програмний продукт, використовуючи TensorFlow.js, може давати рекомендації у повідомленнях щодо оптимального часу для сну та пробудження на основі циркадних ритмів кожного користувача. Він також може нагадувати про необхідність регулярних перерв у роботі для збереження продуктивності та психічного здоров'я.

Для взаємодії з серверною частиною застосунку використовується бібліотека Axios. Вона дозволяє виконувати HTTP-запити для отримання та надсилання даних (див. рис. 4.4).

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:8080/api',
});

export const fetchWorkers = async () => {
  const response = await api.get('/workers');
  return response.data;
};

export const createWorker = async (workerData) => {
  const response = await api.post('/workers', workerData);
  return response.data;
};

// Інші функції для взаємодії з API...
```

Рисунок 4.4 – Приклад коду взаємодії з серверною частиною за допомогою Axios(виконано самостійно)

Для управління станом застосунку використовується Redux. Це дозволяє централізовано зберігати та управляти станом усіх компонентів застосунку (див. рис. 4.5).

```
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import rootReducer from './redux/reducers';

const store = createStore(rootReducer);

const App = () => {
  return (
    <Provider store={store}>
      <NavigationContainer>
<Stack.Navigator initialRouteName="Login">
<Stack.Screen name="Login" component={LoginScreen} />
<Stack.Screen name="Home" component={HomeScreen} />
<Stack.Screen name="WorkerList" component={WorkerListScreen} />
</Stack.Navigator>
      </NavigationContainer>
    </Provider>
  );
};

export default App;
```

Рисунок 4.5 – приклад коду з використанням Redux для керування станами (виконано самостійно)

Розробка клієнтської частини для мобільного застосунку на React Native включає створення інтерфейсу користувача, налаштування навігації, взаємодію з серверною частиною та забезпечення оптимального користувацького досвіду. Інтеграція з TensorFlow.js дозволяє надавати користувачам персоналізовані рекомендації для покращення їхнього здоров'я та продуктивності. Реалізація цих компонентів забезпечує ефективну та інтуїтивну взаємодію з системою для кінцевих користувачів.

4.4 Інтеграція IoT

Інтеграція IoT (Internet of Things) дозволяє отримувати дані з різних сенсорів, відправляти їх на сервер для подальшої обробки та надання рекомендацій. В даному проєкті використовується модуль ESP8266 для збору даних з сенсорів та відправки їх на сервер через Wi-Fi [31].

Для відправки даних на сервер використовується модуль ESP8266. Він з'єднується з Wi-Fi мережею та відправляє дані про шум, освітлення, вологість та температуру на сервер через HTTP POST запит. Нижче представлено код, де зчитуються дані з відповідних сенсорів (див. рис. 4.6).

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <SoftwareSerial.h>
#include <DHT.h>
#define DHTPIN D2     // Пін для підключення датчика DHT22
#define DHTTYPE DHT22 // Тип датчика DHT22

SoftwareSerial mySerial(4, 5); // RX, TX
const char* ssid = "Double_Rose";
const char* password = "Pili50grim";
WiFiClient wifiClient;

DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  dht.begin();

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop () {
  if(Serial.available()){
    int noise = Serial.parseInt();
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    int illumination = analogRead(A0);
    int id_iot = 1;
    if(WiFi.status()== WL_CONNECTED){
      HTTPClient http;
      http.begin(wifiClient, "http://16.171.2.89/api/sensorData");
      http.addHeader("Content-Type", "application/json");
      String json = "{";
      json += "\"humidity\":" + String(humidity) + ",";
      json += "\"temperature\":" + String(temperature) + ",";
      json += "\"noise\":" + String(noise) + ",";
      json += "\"illumination\":" + String(illumination) + ",";
      json += "\"id_iot\":" + String(id_iot) ;
      json += "}";

      Serial.println(json);
      int httpCode = http.POST(json);
      if (httpCode > 0) {
          String payload = http.getString();
          Serial.println(httpCode);
          Serial.println(payload);
      } else {
          Serial.println("Error on HTTP request");
      }
      delay(60000);
    }
  }
}
```

Рисунок 4.6 – код зчитування показників світла, вологості, температури і шуму з датчиків (виконано самостійно)

Підключення бібліотек:

- ESP8266WiFi.h: бібліотека для роботи з Wi-Fi на ESP8266.
- ESP8266HTTPClient.h: бібліотека для виконання HTTP запитів.
- SoftwareSerial.h: бібліотека для роботи з програмним послідовним портом.
- DHT.h: бібліотека для роботи з датчиком DHT22.

Цей код використовується для зчитування показників шуму з аналогового датчика, підключеного до піну A0, та відправки цих показників через програмний серійний порт. Програмний серійний порт налаштований для передачі даних з піну 5 (TX) на інший пристрій, який може приймати ці дані для подальшої обробки або виводу.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(4, 5); // RX, TX

void setup() {
  mySerial.begin(9600);
}

void loop() {
  int noise = analogRead(A0);
  mySerial.println(noise);
  delay(1000);
}
```

Рисунок 4.7 – Код для зчитування показників шуму за допомогою датчика шуму (виконано самостійно)

Інтеграція IoT в даному проєкті дозволяє ефективно збирати та відправляти дані з різних сенсорів на сервер для подальшої обробки. Це забезпечує можливість отримання актуальної інформації про умови навколишнього середовища, що може бути використано для надання рекомендацій користувачам щодо оптимального часу для сну та пробудження, а також для підтримання їх продуктивності та психічного здоров'я.

4.5 Інтеграція машинного навчання

Машинне навчання (ML) відіграє ключову роль у розробці програмної системи, орієнтованої на підвищення продуктивності та покращення

психоемоційного стану працівників. Використовуючи алгоритми машинного навчання, система може аналізувати дані, зібрані з IoT пристроїв та інших джерел, для надання персоналізованих рекомендацій користувачам [32].

Для інтеграції машинного навчання в проєкт були обрані наступні технології та інструменти:

- TensorFlow.js: бібліотека для машинного навчання, яка дозволяє запускати моделі прямо у браузері.
- Python: використовується для тренування складних моделей машинного навчання, які потім експортуються для використання у клієнтській частині.
- Jupyter Notebook: інструмент для інтерактивного аналізу даних та тренування моделей.

Процес інтеграції машинного навчання включає кілька етапів:

1. Збір та підготовка даних: дані зібрані з IoT пристроїв, включаючи рівень шуму, освітлення, вологість та температуру, показники, що впливають на продуктивність та психоемоційний стан працівників.

2. Тренування моделей: використовуючи Python та бібліотеки для машинного навчання (наприклад, TensorFlow), тренування моделі на зібраних даних. Моделі можуть включати класифікатори, регресійні моделі та нейронні мережі, які передбачають оптимальні умови для продуктивної роботи та здорового способу життя.

3. Експорт моделей: після тренування моделі експортуються у форматі, що підтримується TensorFlow.js. Це дозволяє використовувати моделі безпосередньо у клієнтській частині веб-застосунку або мобільного застосунку.

4. Інтеграція з клієнтською частиною: використовуючи TensorFlow.js, інтегруємо моделі у клієнтську частину застосунку. Це дозволяє виконувати передбачення та надавати рекомендації користувачам у реальному часі.

Нижче наведено приклад використання TensorFlow.js для передбачення оптимального часу для сну на основі циркадних ритмів користувача:

5. Тренування моделі у Python (див. рис. 4.8)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
# Приклад даних про режим сну
data = np.array([[22, 6], [23, 7], [19, 6], [18, 7]])
labels = np.array([6, 7, 5, 6, 7])
# Створення моделі
model = Sequential()
model.add(Dense(10, input_dim=2, activation='relu'))
model.add(Dense(1))
# Компіляція моделі
model.compile(optimizer='adam', loss='mean_squared_error')
# Тренування моделі
model.fit(data, labels, epochs=100)

# Збереження моделі
model.save('sleep_model.h5')
Інтеграція моделі у клієнтську частину:
import * as tf from '@tensorflow/tfjs';
```

Рисунок 4.8 – Тренування моделі на циркадних ритмах (виконано самостійно)

Інтеграція машинного навчання в програмну систему дозволяє надати персоналізовані рекомендації для користувачів, підвищуючи їх продуктивність та покращуючи психоемоційний стан, збирати та аналізувати великі обсяги даних у реальному часі, використовувати передові технології для передбачення та оптимізації умов роботи.

Інтеграція машинного навчання у наш проєкт забезпечує глибший аналіз даних та надання цінних рекомендацій користувачам. Використання TensorFlow.js дозволяє реалізувати ці можливості безпосередньо у клієнтській частині, забезпечуючи інтерактивний та ефективний користувацький досвід.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є критично важливим етапом у розробці, оскільки воно дозволяє виявити та виправити помилки, забезпечити стабільну роботу та високу якість кінцевого продукту

Для тестування API було використано системне тестування. Системне тестування (System Testing) включає повну перевірку всієї системи на відповідність вимогам. Це включає тестування всіх функціональних та нефункціональних вимог. У якості інструменту було використано Postman [33].

Postman є потужним інструментом для тестування RESTful API. Він дозволяє створювати та автоматизувати тести для перевірки коректної роботи API.

1. GET-запит для отримання списку робітників:

Метод: GET.

URL: http://localhost:8080/api/worker.

Перевірка: Статус код 200 та наявність поля workers у відповіді.

2. POST-запит для створення нового робітника:

Метод: POST.

URL: http://localhost:8080/api/worker.

Тіло запиту (JSON):

```
{
  "name_worker": "Test Worker",
  "mail_worker": "test@worker.com",
  "usually_sleep_start_time": "22:00",
  "usually_sleep_end_time": "06:00"
}
```

Було протестовано роботу токена для авторизації і захисту даних користувачів, для цього було згенеровано токен і додано в Header GET запиту на вивід списку власників (див. рис. 5.1).

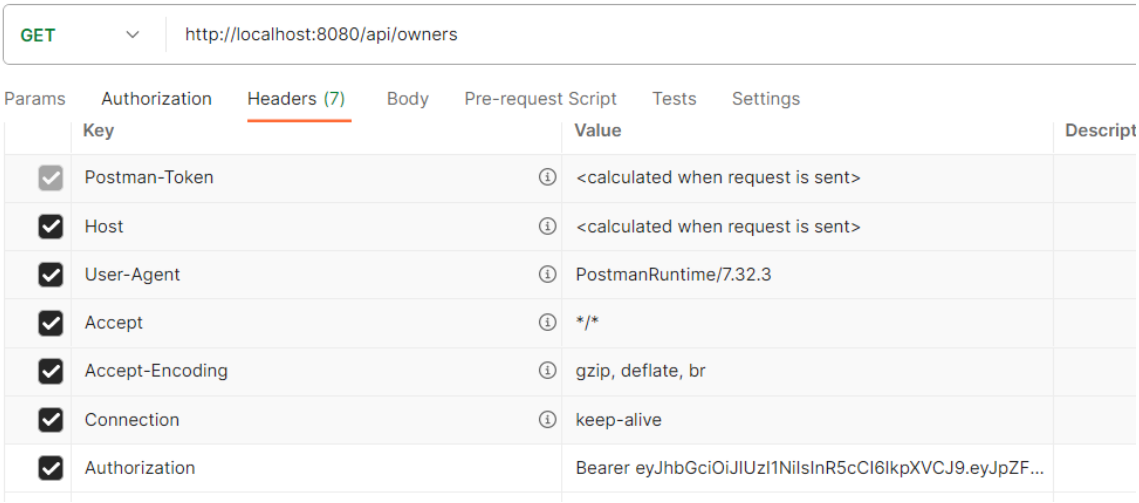


Рисунок 5.1 – Приклад використання Postman для тестування API (виконано самостійно)

Для тестування клієнтської частини веб-застосунку і мобільного застосунку було використано мануальне тестування для перевірки інтуїтивності, доступності та коректності відображення елементів на сторінках веб-застосунку та мобільного застосунку.

Мануальне тестування є ключовою складовою процесу валідації програмного забезпечення, оскільки воно дозволяє виявити помилки, які можуть не бути виявлені автоматизованими тестами. Цей вид тестування включає перевірку користувацького інтерфейсу, бізнес-логіки, інтеграції компонентів системи, безпеки, а також загальної коректності поведінки програмного забезпечення [34].

Мануальне тестування користувацького інтерфейсу виконується для перевірки зручності користування та відповідності інтерфейсу призначеним для користувача завданням.

Навігація по додатку:

- перевірка, що всі кнопки та посилання працюють;
- перевірка на наявність та коректність зображення всіх візуальних елементів.

Форми вводу даних:

- ведення даних в форми та перевірка валідації та збереження даних;
- тестування повідомлень про помилки та підказок для користувача;

Звіти та вивід даних:

- перевірка відображення списків, таблиць та інших засобів візуалізації даних;
- перевірка точності та актуальності виведених даних.

Реакція системи на некоректні дії користувача:

- тестування системи на стійкість до помилок користувача;
- перевірка обробки виняткових ситуацій.

Тестування зміни конфігурації:

- перевірка системи при зміні налаштувань користувача;
- тестування реакції системи на зміни параметрів.

Сценарій мануального тестування для веб-застосунку.

Вхід в систему з коректними обліковими даними:

- відкрити сторінку авторизації.
- ввести коректний логін та пароль.
- натиснути кнопку "вхід".
- переконатися, що користувача перенаправлено на головну сторінку.

Спроба входу з некоректними обліковими даними та перевірка повідомлення про помилку:

- відкрити сторінку авторизації.
- ввести некоректний логін або пароль.
- натиснути кнопку "вхід".
- переконатися, що з'явилося повідомлення про помилку.

Реєстрація нового користувача та перевірка активації акаунта:

- відкрити сторінку реєстрації.
- заповнити всі обов'язкові поля коректною інформацією.
- натиснути кнопку "зареєструватися".
- переконатися, що акаунт створено і користувач може увійти в систему.

Додавання інформації про працівників:

- відкрити сторінку "додати працівника";
- заповнити всі обов'язкові поля коректною інформацією;

- натиснути кнопку "зберегти";
- переконатися, що новий працівник з'явився у списку працівників.

Редагування інформації про працівників:

- відкрити сторінку "список працівників";
- обрати працівника для редагування;
- натиснути кнопку "редагувати";
- внести зміни до будь-якого з полів;
- натиснути кнопку "зберегти";
- переконатися, що зміни збережені та відображаються у списку.

Видалення інформації про працівників:

- відкрити сторінку "список працівників";
- обрати працівника для видалення;
- натиснути кнопку "видалити";
- переконатися, що працівник видалений зі списку.

Використання функцій пошуку для знаходження специфічних даних:

- відкрити сторінку "список працівників";
- ввести частину імені або електронної адреси працівника у поле пошуку;
- переконатися, що у списку відображаються тільки ті працівники, які відповідають критеріям пошуку.

Протягом мануального тестування були виявлені та успішно усунені декілька помилок, що стосувалися навігації по додатку та обробки даних форм. Особливу увагу було приділено тестуванню поведінки системи при введенні користувачами некоректних даних, і відповідні помилки були оброблені з виведенням зрозумілих повідомлень про помилки.

Завершення тестового циклу підтвердило високий рівень стабільності та коректності програмного забезпечення, забезпечивши повну відповідність всім вимогам та специфікаціям проєкту. Ретельне тестування забезпечило впевненість у тому, що програмне забезпечення функціонує належним чином, що дозволило забезпечити високий рівень задоволеності кінцевих користувачів кінцевим продуктом.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було розроблено програмну систему, спрямовану на організацію робочого процесу з урахуванням циркадних ритмів та психоемоційного стану працівників. Під час аналізу предметної галузі було виявлено, що проблеми перевтоми, недосипання та вигорання працівників є актуальними та критичними на загальному рівні, але універсальних рішень для їх подолання ще не існує. У цьому контексті були розроблені конкретні алгоритми, спрямовані на потенційне покращення даної ситуації.

У ході роботи було опубліковано наукову публікацію на тему «Моделювання ПЗ для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників».

Під час проєктування програмної системи були сформульовані чіткі вимоги до функціональності та характеристик системи, а також визначено оптимальні технології для досягнення поставлених цілей. Процес проєктування включав у себе чотири ключових етапи: розробку UML-діаграм для визначення структури та взаємозв'язків компонентів системи, створення архітектури програмного забезпечення, проєктування структури зберігання даних та розробку інтерфейсу користувача.

В процесі розробки програмної системи було ретельно проаналізовано потреби користувачів та особливості їх робочого середовища, що дозволило забезпечити максимальну користь та зручність використання системи. У результаті, були розроблені інтуїтивно зрозумілі і ергономічні інтерфейси, як для мобільного додатку, так і для веб-застосунку, що сприяє зручності користування та підвищує загальну продуктивність користувачів.

Вибір технології TensorFlow.js для впровадження машинного навчання в систему відкрив широкі можливості для аналізу та оптимізації робочого процесу. Використання цієї бібліотеки дозволило створювати потужні та ефективні моделі машинного навчання, використовуючи JavaScript.

Такий підхід дозволив систематизувати та узгодити всі елементи системи, забезпечивши її ефективну та надійну роботу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Frieda-MariÃ de Jager. 81+ troubling workplace stress statistics [Q1 2024 data] - SSR. SelectSoftware Reviews - Reviews of The Best HR and Recruiting Software. URL: https://www.selectsoftwarereviews.com/blog/workplace-stress-statistics (дата звернення: 01.05.2024).

2. State of the global workplace report. Gallup.com. URL: https://www.gallup.com/workplace/349484/state-of-the-global-workplace.aspx (дата звернення: 02.05.2024).

3. Burnout syndrome as an occupational disease in the European Union: an exploratory study. PubMed Central (PMC). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5889935/ (дата звернення: 02.05.2024).

4. 2022 pulse of talent: competing for talent takes more than pay. Paper Library: Training for State and Local Government. URL: https://papers.governing.com/234121071.html/2022-Pulse-of-Talent-Competing-for-Talent-Takes-More-Than-Pay-140981.html (дата звернення: 02.05.2024).

5. Morris N. Brits are 'more exhausted than ever' - over a third on the brink of burnout. Metro. URL: https://metro.co.uk/2022/04/07/brits-are-more-exhausted-than-ever-a-third-on-the-brink-of-burnout-16421027/ (дата звернення: 01.05.2024).

6. Women are more likely to suffer work burnout than men, study finds. The Independent. URL: https://www.independent.co.uk/life-style/work-burnout-men-women-positions-power-self-esteem-family-balance-study-montreal-a8377096.html (дата звернення: 02.05.2024).

7. Bright Horizons Global Homepage. Welcome to Bright Horizons | Bright Horizons. URL: https://www.brighthorizons.com/ (дата звернення: 12.05.2024).

8. Mental Health UK's Burnout Report. Mental Health UK. URL: https://mentalhealth-uk.org/burnout/ (дата звернення: 26.04.2024).

9. Costa-Font J., Fleche S., Pagan R. The labour market returns to sleep. Journal of Health Economics. 2024. Vol. 93. P. 102840. URL: https://doi.org/10.1016/j.jhealeco.2023.102840 (дата звернення: 26.04.2024).

10. Important Burnout Stats 2024 [Trends and Facts to Know]. Learn Digital Marketing. URL: https://thrivemyway.com/burnout-stats/ (дата звернення: 26.04.2024).

11. Million Brits pull sickie | MetLife. MetLife. URL: https://www.metlife.co.uk/about-us/media-centre/media-centre-archive/2022/march/10-Million-Brits-pull-sickie/ (дата звернення: 26.04.2024).

12. . Borysenko K. How Much Are Your Disengaged Employees Costing You?. Forbes. URL: https://www.forbes.com/sites/karlynborysenko/2019/05/02/how-much-are-your-disengaged-employees-costing-you/?sh=781ad92e3437#358339234376 (дата звернення: 26.04.2024).

13. Mraovic J. Career burnout and its effect on health. Clockify Blog. URL: https://clockify.me/blog/productivity/career-burnout/ (дата звернення: 26.04.2024).

14. Half of UK staff would leave job for better burnout support. Employee Benefits. URL: https://employeebenefits.co.uk/half-uk-staff-would-leave-job-better-burnout-support/ (дата звернення: 26.04.2024).

15. Which jobs are most likely to cause a burn-out? - Growth Business. Growth Business. URL: https://growthbusiness.co.uk/jobs-likely-cause-burn-out-15914/ (дата звернення: 26.04.2024).

16. Which jobs cause burn-outs? Crowdsourced Pay Data - Reliable Salary Comparison - Emolument. URL: https://www.emolument.com/career_advice/jobs_most_likely_cause_burnout#gsc.tab=0 (дата звернення: 26.04.2024).

17. Williams S. The industries still overworking their people to the point of burnout. – Hrgrapevine. URL: https://www.hrgrapevine.com/content/article/2022-10-03-the-industries-still-overworking-their-people-to-the-point-of-burnout (дата звернення: 26.04.2024).

18. Cook J. Majority of UK doctors are burnt out, study shows. GPonline | Primary care news, education, CPD, careers and jobs. URL: https://www.gponline.com/majority-uk-doctors-burnt-out-study-shows/article/1584800 (дата звернення: 26.04.2024).

19. 2022 Annual Review. Deloitte United Kingdom. URL: https://www2.deloitte.com/uk/en/pages/annual-review-2022/home.html (дата звернення: 26.04.2024).

20. Wilding M. Am I Burned Out? How To Recognize The 12 Stages Of Burnout. Forbes. URL: https://www.forbes.com/sites/melodywilding/2023/02/21/am-i-burned-out-how-to-recognize-the-12-stages-of-burnout/?sh= 7044a3b9157b (дата звернення: 26.04.2024).

21. Booth J. Facts About Burnout That Can Help You Avoid Exhaustion. Redbook. URL: https://www.redbookmag.com/life/g32072571/facts-about-burnout/ (дата звернення: 26.04.2024).

22. The Relationship Between Technology and Burnout | The Burnout Gamble by Hamza Khan. Burn Bright, Not Out | The Burnout Gamble by Hamza Khan. URL: https://www.theburnoutgamble.com/blog/relationship-between-technology-and-burnout (дата звернення: 26.04.2024).

23. Association between insomnia symptoms, job strain and burnout syndrome: a cross-sectional survey of 1300 financial workers. BMJ Open. URL: https://bmjopen.bmj.com/content/7/1/e012816 (дата звернення: 26.04.2024). Effects of Shift Work on Cognitive Performance, Sleep Quality, and Sleepiness among Petrochemical Control Room Operators. Journal of Circadian Rhythms. 2016. Vol. 14, no. 1. URL: https://doi.org/10.5334/jcr.134 (date of access: 29.03.2024).

24. K S., Venkataraman R., Al-Hamadi H. Trustworthy Data Provisioning for IoT-Based Circadian Health Evaluation Systems. Kuwait Journal of Science. 2023. URL: https://doi.org/10.1016/j.kjs.2023.02.028 (date of access: 29.03.2024).

25. Physiological Responses among Shift Workers and General Population / M. Ahmad et al. International Journal of Environmental Research and Public Health. 2020. Vol. 17, no. 19. P. 7156. URL: https://doi.org/10.3390/ijerph17197156 (date of access: 29.03.2024).

26. Vitale J. A., Weydahl A. Chronotype, Physical Activity, and Sport Performance: A Systematic Review. Sports Medicine. 2017. Vol. 47, no. 9. P. 1859–1868. URL: https://doi.org/10.1007/s40279-017-0741-z (date of access: 29.03.2024).

27. Machine Learning Analyses Reveal Circadian Features Predictive of Risk for Sleep Disturbance / R. Overton et al. Nature and Science of Sleep. 2022. Volume 14. P. 1887–1900. URL: https://doi.org/10.2147/nss.s379888 (date of access: 29.03.2024).

28. Zammetti F. Modern Full-Stack Development: Using TypeScript, React, Node. js, Webpack, Python, Django, and Docker. Apress L. P., 2023.

29. Herron D. Node. js Web Development: Server-Side Web Development Made Easy with Node 14 Using Practical Examples, 5th Edition. Packt Publishing, Limited, 2020.

30. Bershadskiy S., Villa C. React Native Cookbook. Packt Publishing - ebooks Account, 2016. 437 p.

31. Sai Prasad D. S., Taluja A., Sharma V. Feedback System using ESP8266. MR International Journal of Engineering and Technology. 2023. Vol. 10, no. 1. P. 8–11. URL: https://doi.org/10.58864/mrijet.2023.10.1.2 (дата звернення: 01.06.2024).

32. Survey: Tensorflow in Machine Learning / M. Ramchandani et al. Journal of Physics: Conference Series. 2022. Vol. 2273, no. 1. P. 012008. URL: https://doi.org/10.1088/1742-6596/2273/1/012008 (date of access: 06.06.2024).

33. API Testing Using Postman Tool / P. P. Kore et al. International Journal for Research in Applied Science and Engineering Technology. 2022. Vol. 10, no. 12. P. 841–843. URL: https://doi.org/10.22214/ijraset.2022.48030 (date of access: 06.06.2024).

34. Falk J., Nguyen H. Q., Kaner. Testing Computer Software. Wiley & Sons, Incorporated, John, 2011. 496 p.

ДОДАТОК А

Програмний код контролеру для робітників

```
const db = require('../db');

class workerController {

  async createWorker(req, res) {
    const { name_worker, mail_worker, usually_sleep_start_time, usually_sleep_end_time, usually_peak_productivity_time, usually_lowest_productivity_time, id_owner } = req.body;

    const newWorker = await db.query(
        'INSERT INTO worker (name_worker, mail_worker, usually_sleep_start_time, usually_sleep_end_time, usually_peak_productivity_time, usually_lowest_productivity_time, id_owner) VALUES ($1, $2, $3, $4, $5, $6, $7) RETURNING *',
        [name_worker, mail_worker, usually_sleep_start_time, usually_sleep_end_time, usually_peak_productivity_time, usually_lowest_productivity_time, id_owner]
    );

    res.json(newWorker.rows[0]);
  }
    async getWorker(req, res) {
        const id = req.params.id;
        const worker = await db.query('SELECT * FROM worker WHERE id_worker = $1', [id]);
        res.json(worker.rows);
    }
    async updateWorker(req, res) {
        const id = req.params.id;
        const { name_worker ,mail_worker, usually_sleep_start_time, usually_sleep_end_time, usually_peak_productivity_time, usually_lowest_productivity_time, id_owner } = req.body;
        const updatedWorker = await db.query('UPDATE worker SET name_worker= $1, mail_worker = $2, usually_sleep_start_time = $3, usually_sleep_end_time = $4, usually_peak_productivity_time = $5, usually_lowest_productivity_time = $6, id_owner = $7 WHERE id_worker = $8 RETURNING *',
        [name_worker ,mail_worker, usually_sleep_start_time, usually_sleep_end_time, usually_peak_productivity_time, usually_lowest_productivity_time, id_owner, id]);
        res.json(updatedWorker.rows[0]);
    }
    async getWorkers(req, res) {
        const workers = await db.query('SELECT * FROM worker');
        res.json(workers.rows);
    }
    async deleteWorker(req, res) {
        const id = req.params.id;
        await db.query('DELETE FROM worker WHERE id_worker = $1', [id]);
        res.sendStatus(204);
    }
}
module.exports = new workerController();
```

ДОДАТОК Б

Слайди презентації

ПРОГРАМНА СИСТЕМА ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ З УРАХУВАННЯМ ЦИРКАДНИХ РИТМІВ І ПСИХОЕМОЦІЙНОГО СТАНУ ПРАЦІВНИКІВ

Виконала:
Студентка 4 курсу
групи ПЗПІ-20-8
Писаренко Є.М.

Керівник:
ст. викл. кафедри ПІ
Онищенко К. Г.

1

Мета і цілі роботи:

Створення програмної системи для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників.

- Забезпечення ефективного розподілення завдань та ресурсів.
- Підвищення загального самопочуття та продуктивності працівників.
- Врахування фізіологічних та емоційних потреб працівників для оптимізації робочих процесів.
- Надання рекомендацій для покращення якості сну та зменшення стресу.
- Використання сучасних технологій IoT та машинного навчання для збору та аналізу даних.

2

Конкуренти та аналоги:

Microsoft MyAnalytics:

•Опис: Сервіс від Microsoft, який допомагає працівникам розуміти та покращувати свої робочі звички.

•Переваги: Інтеграція з Microsoft Office, аналіз робочих звичок.

•Недоліки: Відсутність врахування фізіологічних та психоемоційних станів.

Google Wellbeing:

•Опис: Інструменти для управління цифровим здоров'ям та добробутом користувачів Android.

•Переваги: Інтеграція з Android, моніторинг використання пристроїв.

•Недоліки: Обмежений фокус на робочому середовищі, відсутність аналізу циркадних ритмів.

Fitbit Health Solutions:

•Опис: Рішення для моніторингу фізичної активності та здоров'я працівників.

•Переваги: Детальний моніторинг фізичної активності, широкий асортимент пристроїв.

•Недоліки: Відсутність інтеграції з робочими процесами, обмежений аналіз психоемоційного стану.

Emotiv Insight:

•Опис: Нейротехнологічний пристрій для моніторингу мозкової активності та емоційного стану.

•Переваги: Точний моніторинг мозкової активності, глибокий аналіз психоемоційного стану.

•Недоліки: Висока вартість, складність у використанні для широкого кола працівників.

Актуальність

Проблеми сучасного робочого середовища:

Перевтома та вигоряння: Зростаюча кількість працівників стикається з перевтомою та професійним вигоранням.

Проблеми зі сном: Недостатній та нерегулярний сон негативно впливає на продуктивність та здоров'я працівників.

Психоемоційний стан: Погіршення психоемоційного стану знижує ефективність та збільшує кількість помилок.

Важливість врахування циркадних ритмів та психоемоційного стану:

Здоров'я працівників: Покращення загального самопочуття.

Продуктивність: Оптимізація робочих процесів підвищує ефективність.

Зниження стресу: Надання рекомендацій для зменшення стресу та покращення якості сну.

Потенціал для впровадження:

Інноваційний підхід: Використання IoT та машинного навчання.

Інтеграція з робочими процесами: Оптимізація робочих середовищ.

Підвищення конкурентоспроможності: Здорові та продуктивні працівники підвищують успіх підприємства.

Постановка задачі

Збір даних:

•Використання IoT пристроїв для збору даних про освітленість, температуру, вологість та рівень шуму.

•Моніторинг психоемоційного стану та циркадних ритмів працівників.

Аналіз даних:

•Використання алгоритмів машинного навчання для аналізу зібраних даних.

•Визначення патернів та рекомендацій для оптимізації робочого процесу.

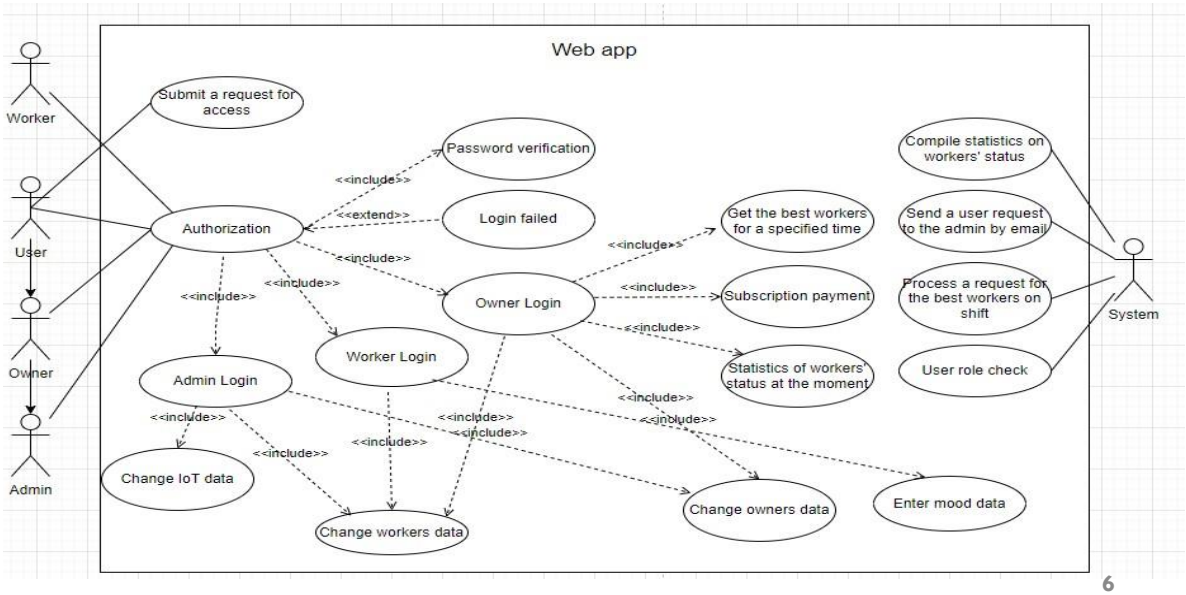
Надання рекомендацій:

•Розробка мобільного та веб-застосунку для відображення рекомендацій.

•Персоналізовані поради для покращення якості сну, зниження стресу та підвищення продуктивності.

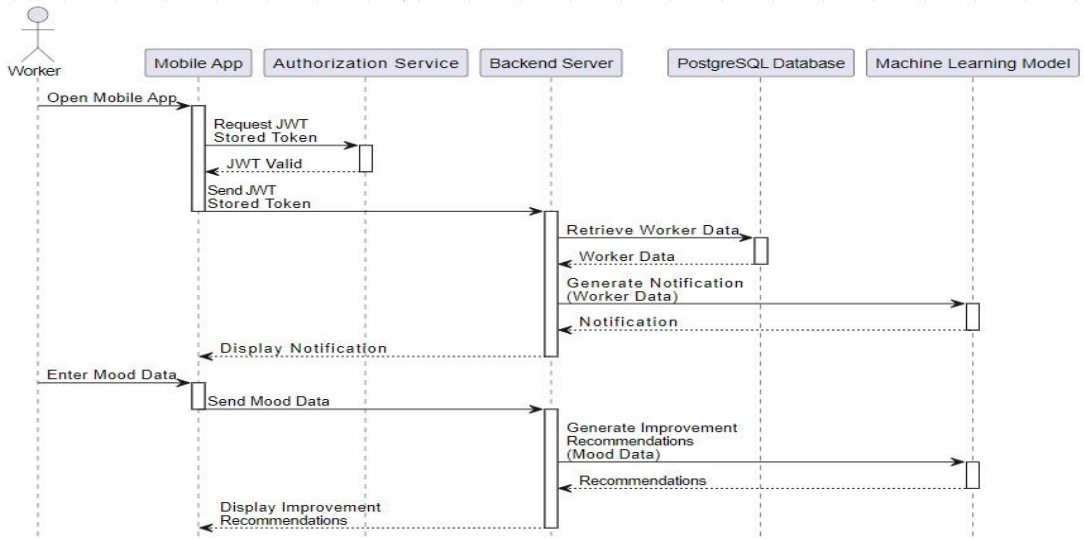
5

Моделювання ПЗ



6

Моделювання ПЗ



7

Компоненти архітектури:

Серверна частина:

Node.js: Використовується як основна серверна платформа для обробки запитів і виконання бізнес-логіки.

Express.js: Фреймворк для побудови маршрутизації та обробки HTTP-запитів.

JSON Web Tokens (JWT):Забезпечує автентифікацію та авторизацію користувачів.

Клієнтська частина:

Веб-застосунок:React.js: Створення динамічних та інтерактивних користувацьких інтерфейсів.

Компонентна структура: Поділ на багаторазово використовувані компоненти для спрощення розробки та підтримки.

Мобільний додаток:

React Native: Створення крос-платформних мобільних додатків для iOS та Android.

Інтеграція з API: Взаємодія з серверною частиною через HTTP-запити.

8

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BH1750.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
Wi-Fi const char* ssid = "SSID";
const char* password = "PASSWORD";
// Параметри сервера
const char* serverUrl = "http://aws-endpoint.com/api/sensorData";
// Ініціалізація сенсорів
Adafruit_BH1750 bh1750;
#define DHTPIN D2
#define DHTTYPE DHT22
 DHT dht(DHTPIN, DHTTYPE);
#define NOISE_PIN A0
 void setup(){
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000); Serial.println("Connecting to WiFi..."); } Serial.println("Connected to WiFi");
 // Ініціалізація BH1750
if (!bh1750.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
Serial.println("Error initializing BH1750"); }
else { Serial.println("BH1750 initialized"); }
// Ініціалізація DHT22
dht.begin();
}
```

10

База даних:

PostgreSQL: Надійне зберігання даних: Реляційна база даних для збереження інформації про працівників, їх стан та дані з IoT пристроїв.

IoT Інтеграція:

ESP8266: Передача даних

Сенсори: BH1750 - Сенсор освітленості. DHT22 - Сенсор температури та вологості. KY-037 - Модуль вимірювання рівня шуму.

Обробка даних та аналіз:

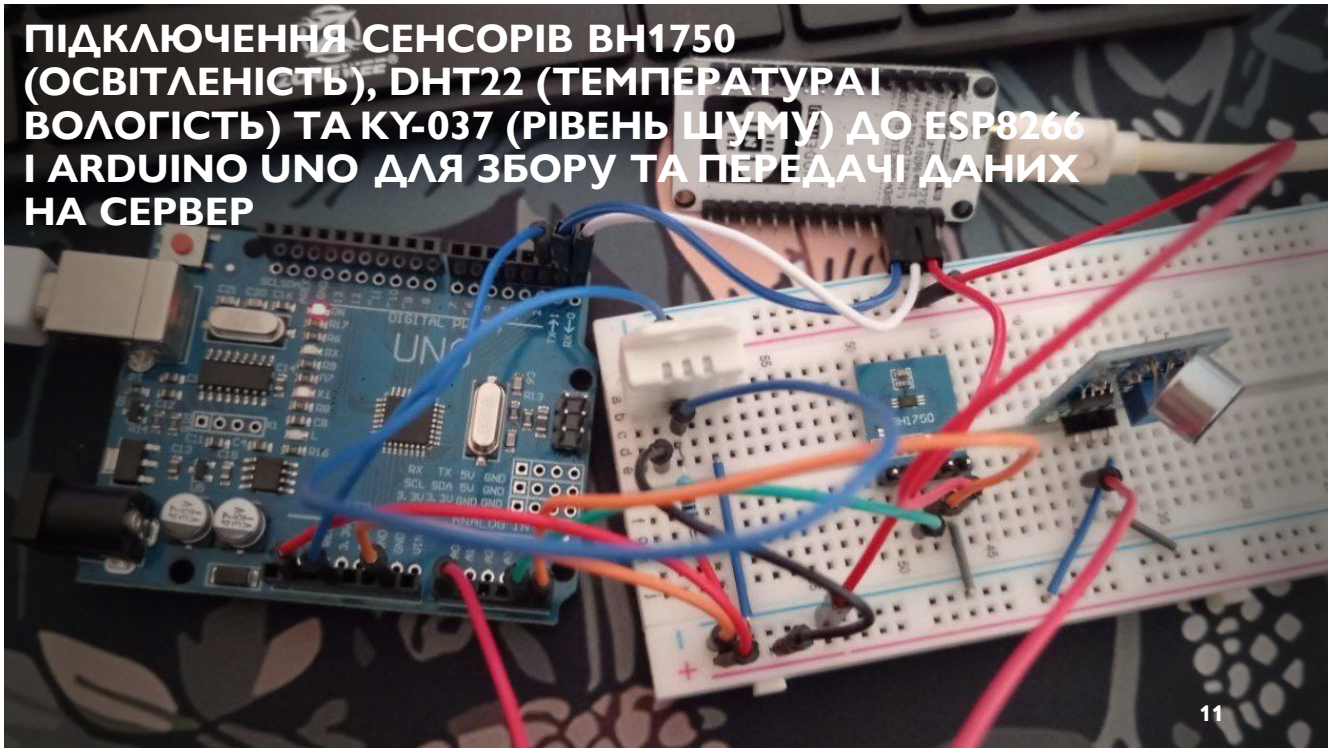
TensorFlow.js: Аналіз зібраних даних для виявлення патернів і надання персоналізованих рекомендацій.

Інтерференція моделей: Виконання моделей машинного навчання на стороні клієнта або сервера.

API RESTful сервіс: Взаємодія між клієнтською та серверною частинами через HTTP-запити.

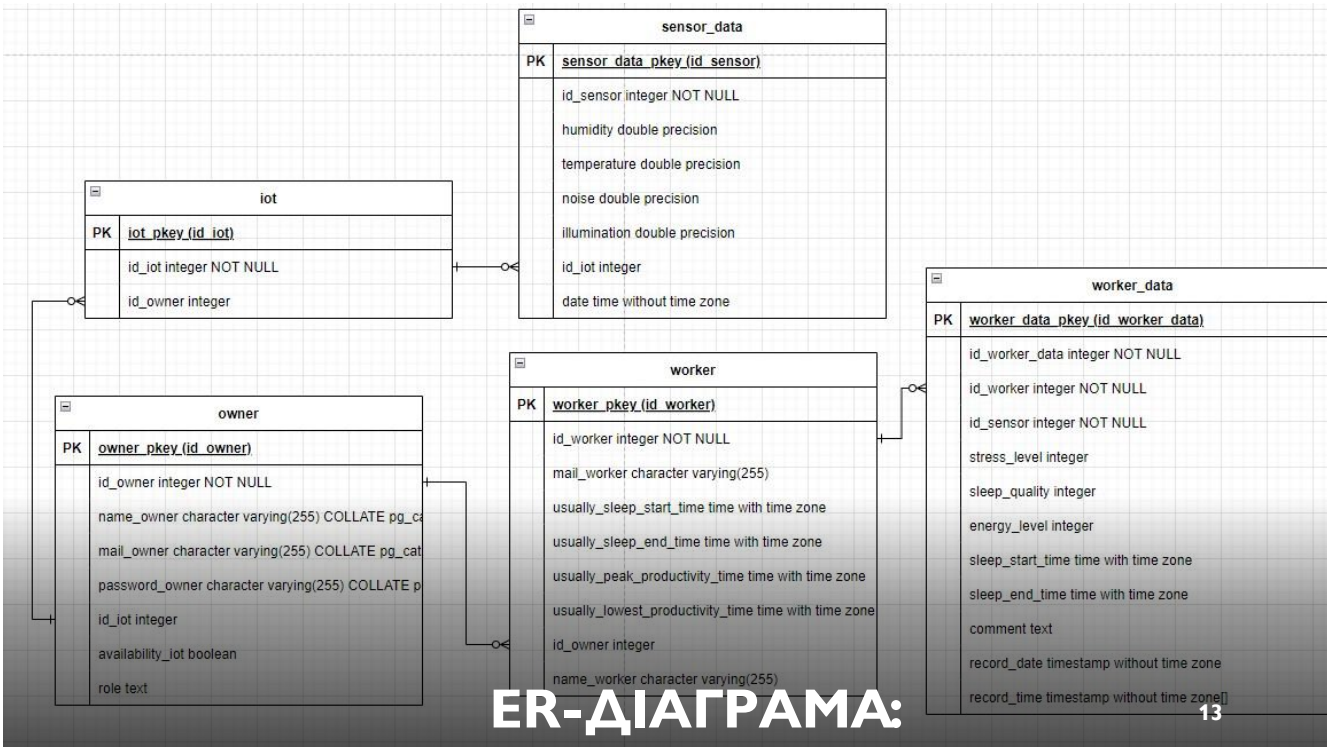
AWS: Тимчасове використання для зберігання та обробки даних, забезпечення масштабованості та надійності.

9




```
void loop() {
// Збір даних з сенсорів
float lux = bh1750.readLightLevel();
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int noise = analogRead(NOISE_PIN);
// Перевірка, чи успішно зібрано дані
if (isnan(temperature) || isnan(humidity)) {
Serial.println("Failed to read from DHT sensor!"); return; }
// Відправка даних на сервер
if (WiFi.status() == WL_CONNECTED) {
HTTPClient http; http.begin(serverUrl);
http.addHeader("Content-Type", "application/json");
String jsonData = "{";
jsonData += "\"lux\":" + String(lux) + ",";
jsonData += "\"temperature\":" + String(temperature) + ",";
jsonData += "\"humidity\":" + String(humidity) + ",";
jsonData += "\"noise\":" + String(noise);
jsonData += "}";
int httpResponseCode = http.POST(jsonData);
if (httpResponseCode > 0) { String response = http.getString();
Serial.println(httpResponseCode); Serial.println(response); }
else { Serial.print("Error on sending POST: ");
Serial.println(httpResponseCode); } http.end();
 } else { Serial.println("WiFi not connected"); }
// Затримка перед наступним вимірюванням
delay(60000); // 1 хвилина }
```

12



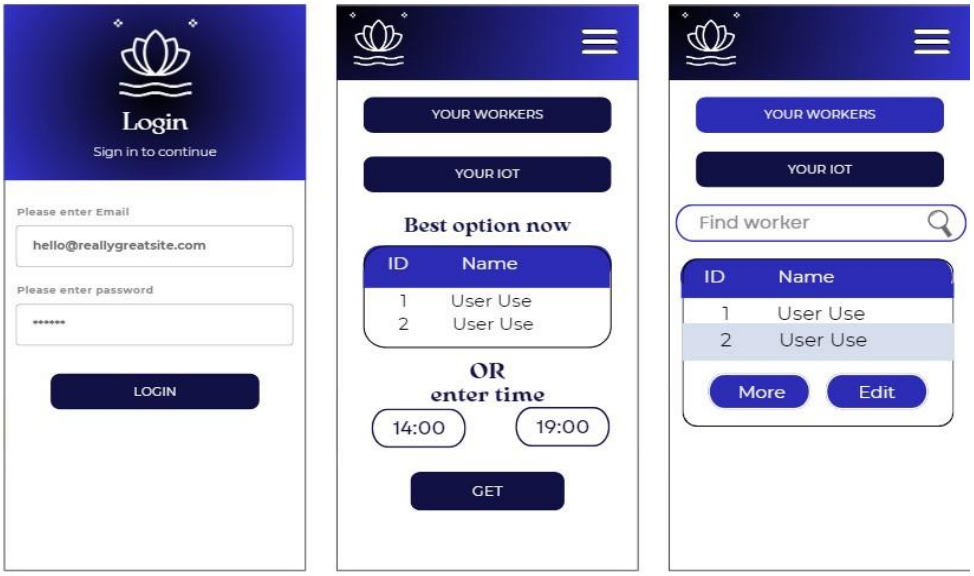
ER-ДІАГРАМА:

13

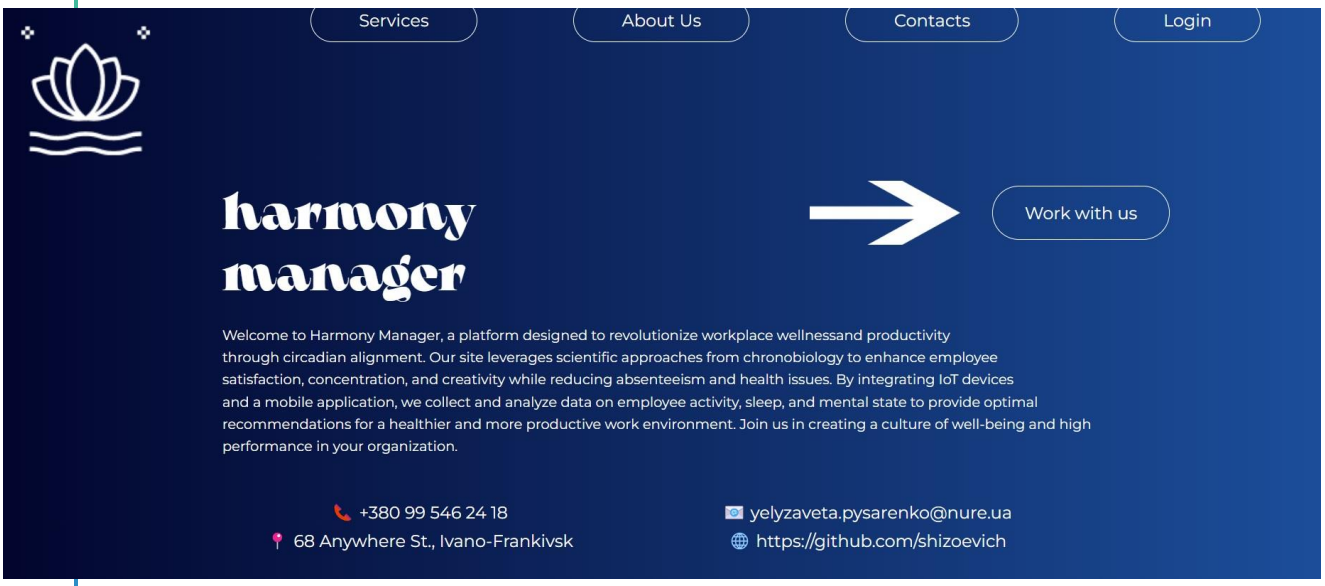
```
class iotController {
    async createIot(req, res) {
        const { id_owner } = req.body;
        const newIot = await db.query('INSERT INTO iot (id_owner) VALUES ($1) RETURNING *', [id_owner]);
        res.json(newIot.rows[0]);
    }
    async getIot(req, res) {
        const id = req.params.id;
        const iot = await db.query('SELECT * FROM iot WHERE id_iot = $1', [id]);
        res.json(iot.rows);
    }
    async test(req, res) {
        res.json({"status":true})
    }
    async updateIot(req, res) {
        const id = req.params.id;
        const { id_owner } = req.body;
        const updatedIot = await db.query('UPDATE iot SET id_owner = $1 WHERE id_iot = $2 RETURNING *',
[id_owner, id]);
        res.json(updatedIot.rows[0]);
    }
    async getIots(req, res) {
        const iots = await db.query('SELECT * FROM iot');
        res.json(iots.rows);
    }
    async deleteIot(req, res) {
        const id = req.params.id;
        await db.query('DELETE FROM iot WHERE id_iot = $1', [id]);
        res.sendStatus(204);
    }
}
```

14

Скріни системи:



15



16

17

Тестування програмного забезпечення

Тестування програмного забезпечення є критично важливим етапом розробки, що дозволяє виявити помилки та забезпечити стабільну роботу продукту.

API Тестування:

- Використання Postman для системного тестування.
- GET та POST-запити для перевірки коректної роботи API.

Мануальне тестування:

- Перевірка інтуїтивності та доступності інтерфейсу.
- Тестування введення даних, валідації та збереження.
- Ретельне тестування забезпечило відповідність вимогам і високу якість програмного забезпечення.

18

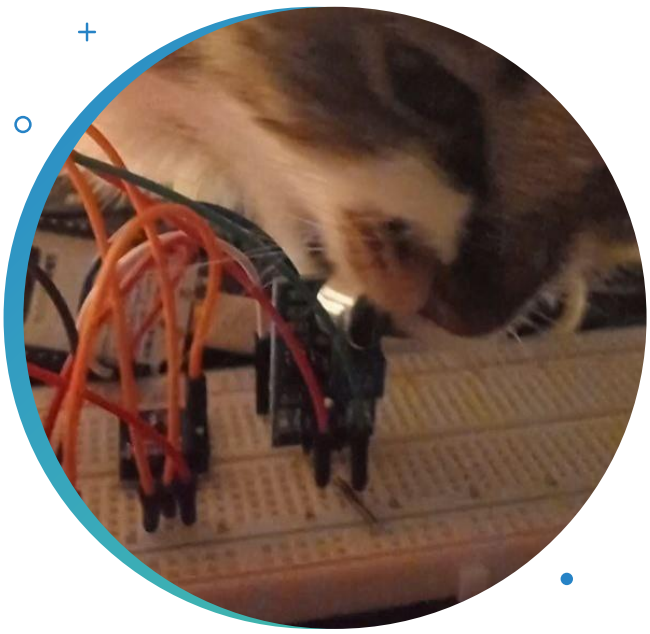
Висновки

У ході виконання роботи було розроблено програмну систему для організації роботи з урахуванням циркадних ритмів і психоемоційного стану працівників. Основними результатами стали:

1. **Аналіз предметної галузі:** Досліджено вплив циркадних ритмів і психоемоційного стану на продуктивність праці.
2. **Проєктування та розробка:** Розроблено архітектуру системи, що включає мобільний додаток, веб-застосунок та IoT пристрої для збору даних.
3. **Машинне навчання:** Впроваджено алгоритми машинного навчання для аналізу даних та надання рекомендацій працівникам.
4. **Тестування:** Проведено тестування програмного забезпечення, що включає системне та мануальне тестування, для забезпечення стабільної роботи системи.

Розроблена система сприяє підвищенню продуктивності працівників та їх загальному самопочуттю, враховуючи індивідуальні фізіологічні та психологічні потреби.

У ході виконання роботи було також підготовлено матеріали, які успішно прийняті до публікації на II Міжнародній науково-практичній конференції "PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE", яка відбулася 1-3.04.2024 у Львові, Україна.

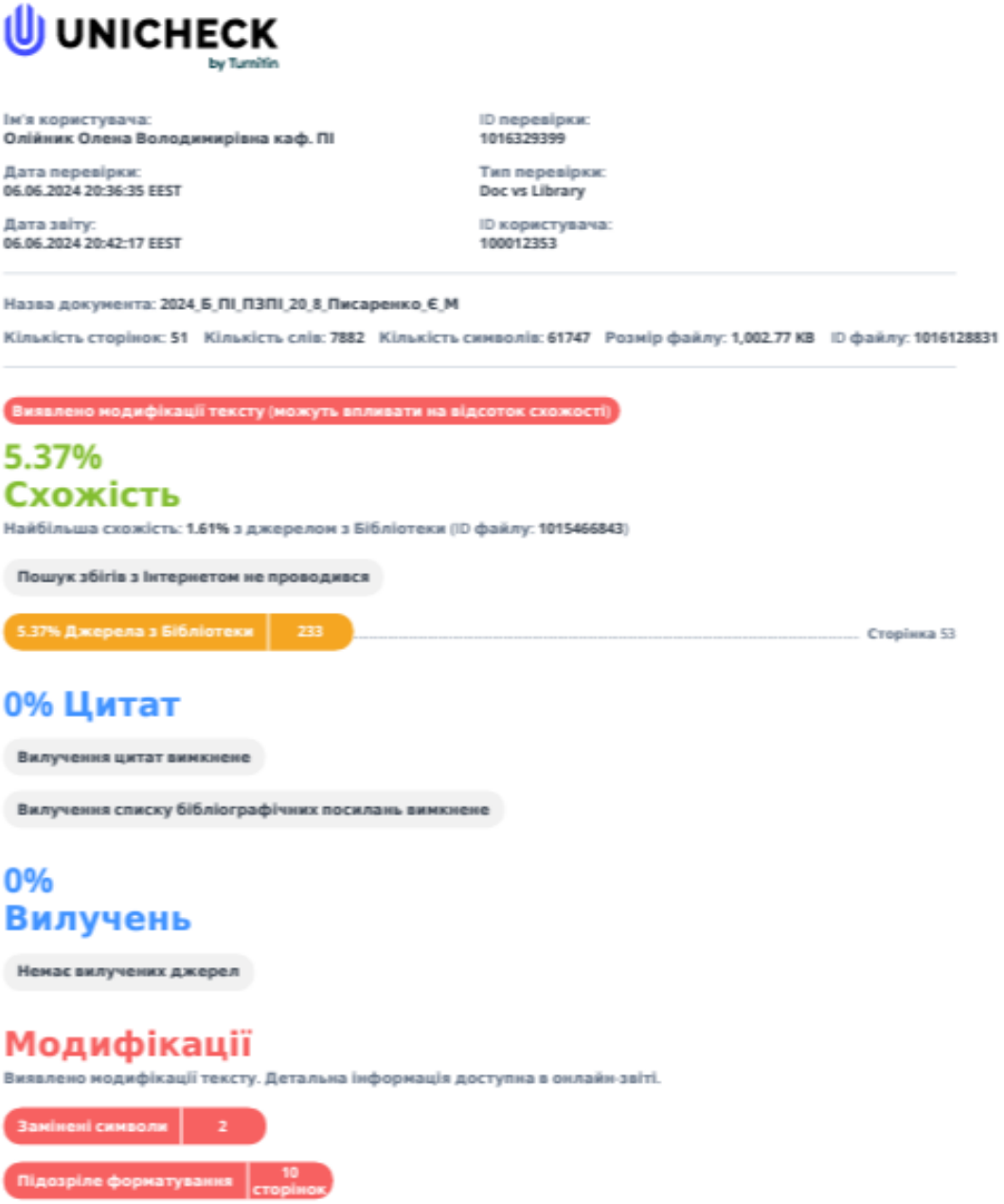


ДЯКУЮ ЗА УВАГУ!

ГОТОВА ВІДПОВІСТИ НА ВАШІ ЗАПИТАННЯ.

ДОДАТОК В

Висновок про плагіат



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016329399

Дата перевірки:
06.06.2024 20:36:35 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
06.06.2024 20:42:17 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_8_Писаренко_Є_М

Кількість сторінок: 51 Кількість слів: 7882 Кількість символів: 61747 Розмір файлу: 1,002.77 KB ID файлу: 1016128831

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.37%
Схожість

Найбільша схожість: 1.61% з джерелом з Бібліотеки (ID файлу: 1015466843)

Пошук збігів з Інтернетом не проводився

5.37% Джерела з Бібліотеки 233 Сторінка 53

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 2

Підозріле форматування 10 сторінок