Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент групи КН-20-1* _____ *Олександр КЕНС*
Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Керівник: *док.філ., ст. викл. каф. КН* _____ *Павло РАДЮК*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

Нормоконтроль: *к.т.н., доц. каф. КН* _____ *Руслан БАГРІЙ*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри КН, д.т.н., професор _____ *Олександр БАРМАК*
Підпис | Ім'я, ПРІЗВИЩЕ

20 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор Олександр БАРМАК
«*16*» *02* *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних»*

2. Завдання видано студенту *Олександру КЕНСУ*
(ім'я, прізвище)

3. Керівник роботи *старший викладач кафедри КН Павло РАДЮК*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «*15*» *02* *2024* р. № *8*

5. Дата видачі завдання студенту: «*16*» *02* *2024* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Метою кваліфікаційної роботи бакалавра є покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу. Досягнення мети роботи передбачало розв'язання таких завдань: проведено аналіз наявних методів, засобів та підходів до управління часом та планування розпорядку дня; розроблено метод покращення планування розпорядку дня засобами інтелектуального аналізу даних; спроєктовано вебсервіс для покращення розпорядку дня на основі розробленого методу; виконано програмну реалізацію вебсервісу за розробленим методом покращення планування розпорядку дня; проведено тестування розробленого методу покращення планування розпорядку дня та створеного на його основі вебсервісу; продемонстровано покращення планування розпорядку дня через впровадження розробленого методу.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2024*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2024*	*виконано*
3	*Робота над розділом 1 – Аналітичний оглядів методів, засобів та підходів до процесу планування розпорядку дня*	*лютий 2024*	*виконано*
4	*Робота над розділом 2 – Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних*	*березень 2024*	*виконано*
5	*Робота над розділом 3 – Програмна реалізація вебсервісу для покращення планування розпорядку*	*квітень 2024*	*виконано*
6	*Написання пояснювальної записки, урахування зауважень керівника, оформлення роботи, відповідно до вимог*	*травень 2024*	*виконано*
7	*Розроблення презентаційних матеріалів та попередній захист кваліфікаційної роботи*	*травень 2024*	*виконано*
8	*Захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*

Виконавець: *студент групи КН-20-1* (Група виконання) _____ (Підпис) *Олександр КЕНС* (Ім'я, ПРІЗВИЩЕ)

Керівник: *док.філ., ст. викл. каф. КН* (Науковий ступінь, посада) _____ (Підпис) *Павло РАДЮК* (Ім'я, ПРІЗВИЩЕ)

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-20-1 Олександр КЕНС*

Керівник кваліфікаційної роботи бакалавра: *доктор філософії, старший викладач кафедри КН Павло РАДЮК*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
68	36	1	26	4

Метою кваліфікаційної роботи бакалавра є покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу. Для розроблення вебсервісу було використано різні програмні засоби: REST, Spring Boot, Spring Data, Spring Security, PostgreSQL та MongoDB. За допомогою інтелектуального аналізу даних система генерує індивідуалізовані рекомендації для оптимального використання часу, враховуючи унікальні потреби користувача.

Створений вебсервіс може бути корисним користувачам, щоби підвищити їхню продуктивність, зменшити стрес та досягти кращого балансу між професійним та особистим життям.

Ключові слова: тайм-менеджмент, розпорядок дня, інтелектуальний аналіз даних, генетичний алгоритм, вебсервіс.

Виконавець: *студент групи КН-20-1* (Група виконання) _____ (Підпис) *Олександр КЕНС* (Ім'я, ПРІЗВИЩЕ)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
ІАД	Інтелектуальний аналіз даних
ІС	Інформаційна система
CSS	Cascading Style Sheets
DLA	Deep Learning Algorithms
DTO	Data Transfer Object
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
MILP	Mixed Integer Linear Programming

Вступ

Кваліфікаційна робота бакалавра присвячена покращенню планування розпорядку дня через розроблення методу покращення планування розпорядку дня засобами інтелектуального аналізу даних та створення на його основі вебсервісу.

Актуальність. На сьогодні, сучасний світ насичений різноманітними викликами та завданнями, тому ефективне управління часом стає важливою умовою успішності будь-якої людської діяльності. Ріст потреб щодо покращення розпорядку дня та планування різноманітних діяльностей породжує необхідність у впровадженні до розв'язання цього завдання інноваційних підходів та інформаційних технологій. Тому актуальним є розроблення та впровадження методу, що дасть можливість покращити планування розпорядку дня за допомогою інтелектуального аналізу даних.

Об'єкт дослідження – процес планування розпорядку дня засобами інтелектуального аналізу даних.

Предмет дослідження – методи інтелектуального аналізу даних для планування розпорядку дня.

Мета кваліфікаційної роботи бакалавра – покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

Завдання кваліфікаційної роботи бакалавра – Провести ретельний аналіз наявних методів, засобів та підходів інтелектуального аналізу даних до управління часом та планування розпорядку дня засобами інтелектуального аналізу даних. Розробити метод покращення планування розпорядку дня засобами інтелектуального аналізу даних. Спроєктувати вебсервіс для покращення розпорядку дня на основі розробленого методу. Виконати програмну реалізацію вебсервісу за розробленим методом покращення планування розпорядку дня. Здійснити тестування розробленого методу покращення планування розпорядку дня та створеного на його основі вебсервісу. Продемонструвати покращення планування розпорядку дня через впровадження розробленого методу.

Розділ 1 Аналітичний оглядів методів, засобів та підходів до процесу планування розпорядку дня

1.1 Аналіз проблеми управління часом

У сучасному світі, де швидкість та складність інформаційних потоків стають невід'ємною частиною повсякденного життя, ефективне планування розпорядку дня [1] стає особливо важливим аспектом. Зокрема, питання поліпшення тайм-менеджменту стає важливим, особливо коли маєш справу з безліччю інформації та завдань. Наразі, ефективне управління часом – це ключовий компонент успіху, будь то у професійній сфері чи в особистому житті.

Головна мета вдосконалення тайм-менеджменту [2] полягає в тому, щоб грамотно використовувати часові ресурси для досягнення своїх конкретних цілей. Це охоплює багато різних аспектів [3], таких як планування робочого дня, встановлення пріоритетів та забезпечення належного відпочинку.

Бажання поліпшити тайм-менеджмент постає з усвідомлення того, що ефективне управління часом безпосередньо впливає на продуктивність та задоволення від нашої роботи чи життя [4]. Це особливо важливо в умовах високого темпу життя та постійних викликів. Існує кілька факторів, які підкреслюють важливість поліпшення тайм-менеджменту. Зокрема, швидкі технологічні зміни та ринкові умови вимагають від нас постійного оновлення та вивчення нового. Також, зростання ролі технологій у нашому повсякденному житті [5] призводить до збільшення обсягу інформації, яку нам доводиться опрацьовувати кожен день.

Не менш важливим є вплив соціальних мереж на розхитування уваги. Часті візити на соціальні мережі чи постійні повідомлення можуть вплинути на наш робочий графік та сконцентрованість [6], що, в свою чергу, впливає на ефективність роботи.

Важливо розуміти, що завдання поліпшення тайм-менеджменту не обмежується лише сферою роботи. Воно охоплює всі аспекти нашого життя, включаючи час для відпочинку, освіти, творчості та особистих відносин. Баланс

між цими різними галузями людської діяльності потребує ретельного планування та управління часом.

Аналіз наявних методів для поліпшення тайм-менеджменту може допомогти зрозуміти, які підходи вже використовуються та як вони впливають на ефективність управління часом. Розглянемо деякі з них. Традиційне планування робочого дня – це один із основних методів, де завдання визначаються та розподіляються на конкретні години чи періоди. Цей підхід допомагає встановлювати пріоритети та структурувати час, але вимагає дисципліни та може бути менш гнучким у змінних умовах. Ще однією популярною стратегією є встановлення пріоритетів, де люди визначають найважливіші завдання та акцентують свою увагу на їх вирішенні. Цей метод дає змогу фокусуватися на ключових завданнях, але може призвести до відступу від менш важливих, але також значущих справ. Техніки управління часом, такі як метод Помодоро [7], базуються на ідеї розбиття робочого часу на короткі періоди активності та відпочинку для підвищення продуктивності.

Інші підходи включають делегування завдань, використання матриць управління часом, аутсорсинг рутинних завдань та використання принципів Lean-управління для усунення зайвого витрати часу.

Важливо пам’ятати, що кожен метод має свої плюси та мінуси, і вибір конкретного підходу залежить від особистих уподобань, типу робіт та індивідуальних характеристик. Комбінування різних методів [8] може бути ефективним способом забезпечити гнучкість та оптимальне використання часу в різних ситуаціях.

Один із основних аспектів критики наявних методів тайм-менеджменту – це їхня недостатня гнучкість [9]. Звичайні підходи, такі як розпланування робочого дня чи встановлення пріоритетів, можуть виявитися неефективними у змінних умовах або при несподіваних змінах пріоритетів. Така жорсткість може викликати стрес і ускладнювати адаптацію до змін у робочому середовищі.

Ще однією серйозною критикою є недооцінка індивідуальних особливостей користувачів. Багато методів передбачають, що всі люди реагують

на різні стратегії однаково, але різниці в особистості, стилях роботи та типах завдань можуть значно вплинути на ефективність цих методів.

Технічні засоби, такі як застосунки для планування та відстеження часу, також піддаються критиці через свої недоліки. Зокрема, може виникати проблема перевантаження інформацією [10], коли обсяг даних стає надто великим для ефективного використання. Також, ці технічні засоби не завжди враховують індивідуальні особливості користувачів і можуть викликати відволікання або збільшення стресу.

Сучасний світ потребує від нас постійного адаптивного підходу до управління часом. Наявні методи нерідко неспроможні ефективно впоратися з нестабільністю сучасного життя, де робота може розподілятися між різними часовими зонами, а робоче середовище постійно зазнає змін. Крім того, традиційні стратегії, спрямовані на просте розподілення часу, можуть обмежувати можливості розвитку та досягнення високих результатів.

В загальному, аналіз та критика наявних методів поліпшення тайм-менеджменту підкреслює необхідність нових, більш гнучких та індивідуалізованих стратегій. Розвиток таких підходів може значно покращити ефективність використання часу в умовах сучасного світу і задовольнити різноманітні потреби користувачів з різними особистими та професійними характеристик

1.2 Огляд методів та підходів до розв’язку задачі управління часом

Перед розробленням методу для покращення планування розпорядку дня було розглянуто різні підходи, які можуть бути використані для вирішення цього завдання, проаналізовано їхні переваги та недоліки, а також обрано найкращий.

Алгоритми є потужним інструментом у сфері штучного інтелекту [11], який здатний до вивчення складних залежностей у даних. У контексті покращення планування розпорядку дня можна використати Deep Learning Algorythms (DLA), вони можуть відігравати важливу роль у вирішенні завдань

автоматичного аналізу та оптимізації [12]. Переваги використання DLA для цієї задачі можуть бути наступними: автоматизація та індивідуалізація. Ці алгоритми можуть адаптуватися до унікальних потреб кожного користувача, аналізуючи його звички, пріоритети та обмеження. Вони автоматично аналізують великі обсяги даних, щоб визначити найефективніші рішення для планування розпорядку дня, і можуть надавати індивідуалізовані рекомендації, що відповідають конкретним потребам користувача. Однак варто враховувати, що для успішного використання DLA необхідна велика кількість даних для навчання і час на їхню підготовку. Також важливо враховувати можливість виникнення перевантаження системи або недостатньо точних результатів внаслідок недостатньої якості вихідних даних або неправильної настройки параметрів алгоритму.

Методи змішаних цілочислових лінійних програм (MILP) [13] є математичними інструментами, які використовуються для оптимізації складних систем з обмеженнями та цільовими функціями (рисунок 1.1).

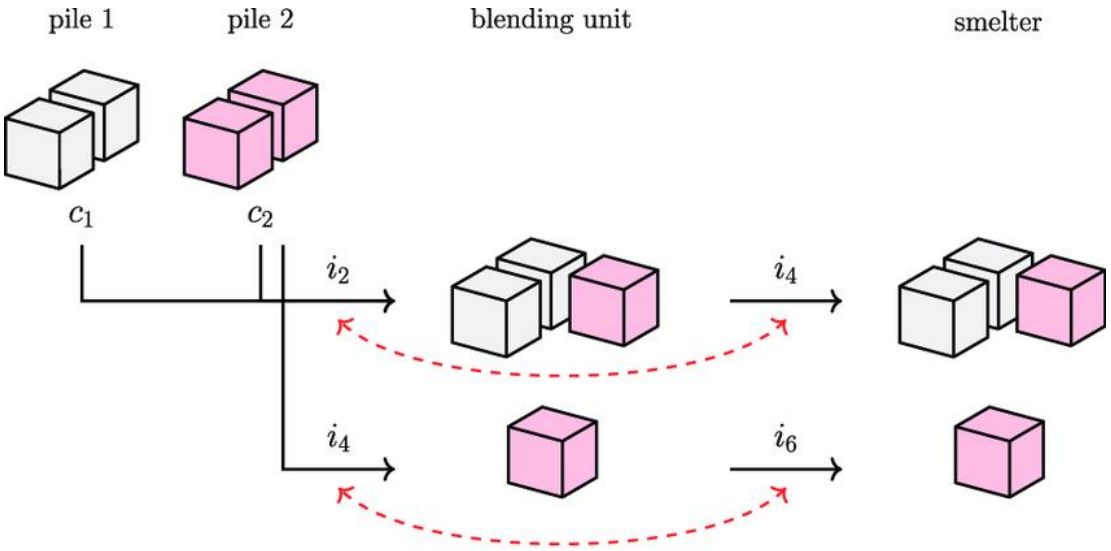


Рисунок 1.1 – Приклад архітектури методу змішаних цілочислових лінійних програм [14]

Ці методи можуть бути застосовані і до задачі покращення планування розпорядку дня шляхом формалізації цієї проблеми у вигляді математичної

моделі та її розв’язанням для отримання оптимального розкладу. Однією з головних переваг використання методів MILP є їх універсальність та здатність до вирішення різноманітних задач оптимізації, включаючи планування розпорядку дня. Завдяки можливості встановлення цілочислових обмежень на змінні, ці методи можуть моделювати різні аспекти планування, такі як час роботи, пріоритети завдань та обмеження ресурсів. Крім того, методи MILP можуть враховувати різні типи обмежень, такі як часові, кількісні та категорійні, що дає змогу їм моделювати різні аспекти реального життя та враховувати індивідуальні потреби користувачів.

Однак важливо відзначити, що вирішення складних задач MILP може вимагати значних обчислювальних ресурсів та часу, особливо при великій кількості змінних та обмежень. Також результати розв’язання можуть бути залежні від якості початкових даних та правильного вибору параметрів моделі. У підсумку, методи змішаних цілочислових лінійних програм можуть бути корисним інструментом для покращення планування розпорядку дня, забезпечуючи систематичний та оптимальний підхід до управління часом та завданнями. Однак їхня продуктивність може залежати від різних факторів, і важливо ретельно аналізувати та налаштовувати модель для досягнення найкращих результатів.

Методи оптимізації за допомогою пропагації зворотного поширення [15] (backpropagation) є ключовими для навчання нейронних мереж, підгонки їх параметрів так, щоб мінімізувати помилку прогнозування. Ці методи стали ключовими для багатьох застосувань штучного інтелекту, включаючи розпізнавання образів, оброблення природної мови та, звісно, планування розпорядку дня. Коли ми говоримо про покращення планування розпорядку дня, нейронні мережі, оптимізовані за допомогою пропагації зворотного поширення, можуть бути застосовані для розроблення моделей, які враховують різноманітні аспекти життя користувача: від робочих завдань до особистих занять та відпочинку. Однією з ключових переваг такого підходу є його гнучкість. Нейронні мережі можуть навчитися адаптуватися [16] до різноманітних умов та

індивідуальних потреб кожного користувача. Наприклад, вони можуть враховувати звички, пріоритети, особисті перерви та обмеження робочого часу. Крім того, ці методи дають змогу моделювати складні залежності між різними факторами, такими як час доби, тип завдань та їх вплив на ефективність користувача. Наприклад, можна розробити модель, яка оптимізує розподіл часу між різними видами діяльностей для максимізації продуктивності та задоволення користувача. Проте важливо враховувати, що тренування нейронних мереж може бути витратним за обчислювальними ресурсами та часом. Також результати можуть бути чутливими до якості вихідних даних та правильності вибору архітектури та параметрів моделі. Отже, використання методів оптимізації за допомогою пропагації зворотного поширення для покращення планування розпорядку дня потребує уважного аналізу та налагодження, але може принести значні переваги в організації та ефективності робочого та особистого часу.

Генетичні алгоритми представляють собою потужний інструмент для розв’язання складних задач оптимізації [17], які включають великий простір пошуку та численні обмеження (рисунок 1.2).

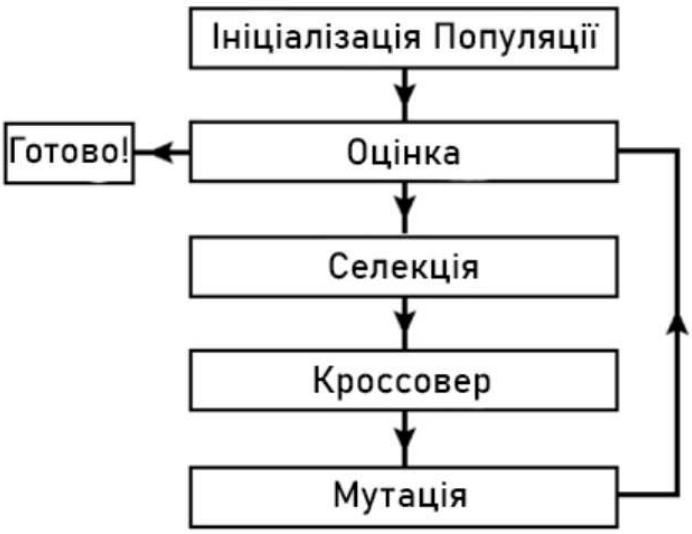


Рисунок 1.2 – Приклад архітектури генетичного алгоритму [18]

Їхніми основними перевагами є здатність працювати з різноманітними типами даних і функціями вартості, що робить їх відмінним інструментом для планування розпорядку дня. Генетичні алгоритми базуються на ідеї природного відбору та генетики, які спостерігаються в природі. У природі, організми з певними характеристиками мають більшу ймовірність виживання та передачі своїх генетичних матеріалів на наступне покоління. ГА використовують аналогічний принцип: вони оперують популяцією індивідів, кожен з яких представляє потенційне рішення задачі.

Один із перших етапів роботи генетичного алгоритму – це створення початкової популяції. Початкова популяція складається зі випадково створених індивідів, які відображають можливі рішення задачі планування розпорядку дня. Кожен індивід має свій генотип, який представляється у вигляді хромосоми. У контексті планування розпорядку дня, ці хромосоми можуть кодувати розклад, щоденні завдання, пріоритети тощо. Після створення початкової популяції, кожен індивід в оцінюється за допомогою функції пристосованості. Ця функція визначає, наскільки добре кожен розклад відповідає вимогам та критеріям ефективності планування. Індивіди з більшим значенням функції пристосованості мають більше шансів бути вибраними для подальшого розмноження.

Далі відбувається процес селекції, де індивіди з кращою пристосованістю мають більше шансів на виживання та розмноження. Вибір індивідів для подальшого розмноження може відбуватися різними методами, такими як турнірна селекція, рулетковий відбір або вибір за рангом. Після селекції, індивіди здійснюють обмін своїми генетичними матеріалами за допомогою процесу схрещування. Це може бути виконано різними способами, наприклад, одноточковим або багато точковим схрещуванням. Після схрещування, можуть відбуватися мутації, коли деякі генетичні характеристики індивідів випадково змінюються.

Цей процес повторюється протягом декількох ітерацій або поки не буде досягнуто критерію закінчення. Критерієм закінчення може бути досягнення

заданої кількості ітерацій, досягнення певного рівня пристосованості або інші умови, встановлені користувачем. Застосування генетичних алгоритмів у вирішенні задач планування розпорядку дає змогу знаходити оптимальні рішення в складних ситуаціях та враховувати багато факторів, таких як пріоритети, обмеження часу, особисті уподобання тощо.

Проаналізувавши різні підходи до вирішення завдання, було обрано для розроблення генетичний алгоритм, адже він є потужним інструментом для планування розпорядку дня, проте його ефективність може бути забезпечена тільки при правильному налаштуванні та використанні відповідно до конкретних вимог та умов завдання.

1.3 Аналіз наявних програмних засобів для покращення управління часом

Проведення огляду наукових робіт та публікацій з теми покращення планування розпорядку дня з використанням інформаційого аналізу даних (ІАД) є наступним ключовим моментом у розумінні та аналізі проблеми. Дана область здобула значну увагу дослідників та практиків, оскільки ефективне управління часом стає важливим чинником для досягнення особистих та професійних цілей.

Багато досліджень приділяють значну увагу теоретичним аспектам планування розпорядку дня, використовуючи ІАД для покращення цього процесу. Наприклад, у роботі “Time Management: A Review” [19] автори детально аналізують сучасні стратегії управління часом та досліджують можливості застосування ІАД для вдосконалення цих стратегій. Вони розглядають різні методи та підходи, що використовуються для ефективного планування часу, і наголошують на необхідності впровадження новітніх технологій аналізу даних у цей процес. За їхніми висновками, інтеграція таких технологій дає змогу значно підвищити продуктивність та ефективність роботи. Автори також акцентують увагу на важливості систематичного збору та оброблення даних для отримання точних і корисних рекомендацій щодо

покращення планування розпорядку дня, що, у свою чергу, сприяє більш ефективному використанню часу і досягненню кращих результатів у різних сферах діяльності.

Дослідження [20] детально розглядає використання ІАД для ідентифікації та аналізу особистих звичок і патернів у використанні часу. Автори пропонують комплексну методологію, яка ґрунтується на зборі та аналізі великих обсягів даних про активності користувача. Ця методологія включає в себе різні етапи, починаючи від збору даних за допомогою сенсорів та застосунків, до їх подальшої оброблення і аналізу з використанням алгоритмів машинного навчання. Отримані результати дають змогу розробляти персоналізовані рекомендації, що враховують індивідуальні особливості кожного користувача. Завдяки такому підходу можна досягти значного покращення у плануванні розпорядку дня, що, в свою чергу, сприяє підвищенню загальної продуктивності та ефективності.

Автори дослідження підкреслюють, що такий індивідуальний підхід до планування часу є надзвичайно важливим у сучасному світі, де швидкий темп життя і велика кількість завдань вимагають максимальної оптимізації кожного робочого дня.

Щодо аналізу наявних програмних засобів для покращення планування розпорядку дня, можна виділити деякі популярні інструменти та застосунки, які вже використовуються в даній області.

Trackabi – це інноваційний вебсервіс [21], призначений для обліку робочого часу та керування відпустками (рисунок 1.3). Цей застосунок розроблений спеціально для фрілансерів, консультантів, а також для малих і середніх компаній, які потребують ефективного інструменту для моніторингу робочого часу. Trackabi не лише забезпечує точний облік відпрацьованих годин, але й надає можливість керувати відпустками та іншими видами відсутностей. Крім того, цей вебсервіс пропонує користувачам зручну платформу для автоматизації рутинних процесів, що значно полегшує адміністрування та управління персоналом.

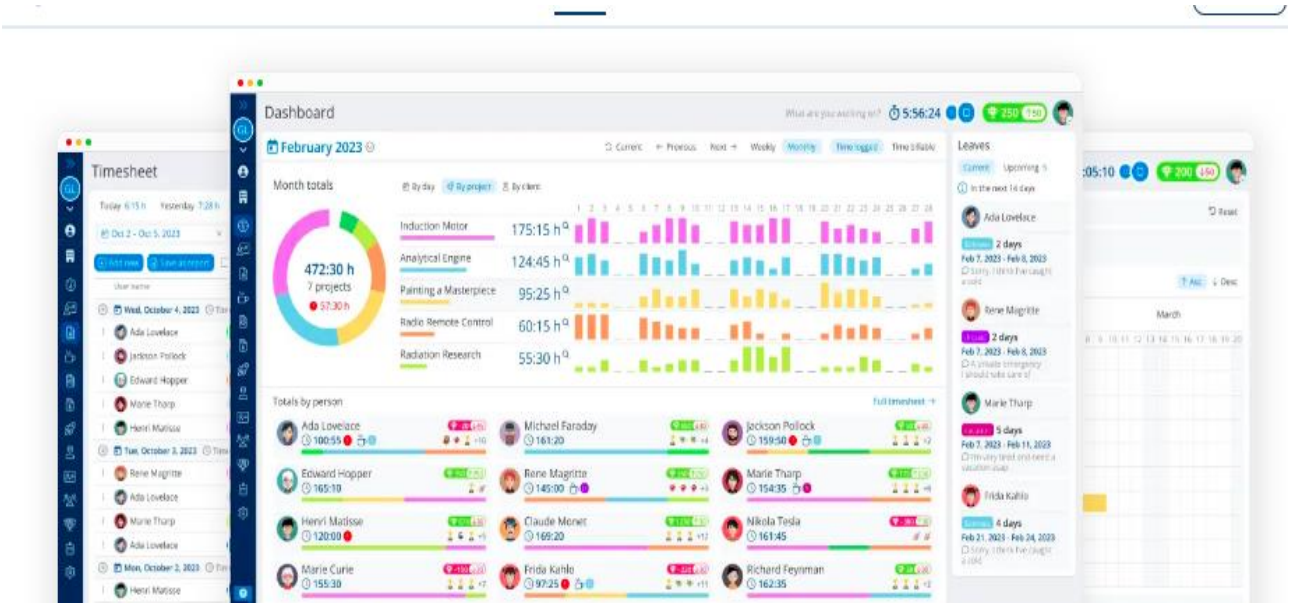


Рисунок 1.3 –Вигляд вебсервісу Trackabi [22]

Також Trackabi являє собою комплексну платформу, яка поєднує в собі набір зручних інструментів для збирання та оброблення даних. Ці інструменти дають змогу бізнесам легко та ефективно аналізувати дані, які щодня генеруються в процесі діяльності. Завдяки цьому, компанії можуть отримувати цінні інсайти щодо продуктивності працівників, оптимізувати використання робочого часу та приймати обґрунтовані рішення на основі реальних даних. Рисунок 1.3 демонструє різноманітні функції та можливості Trackabi, які роблять цей вебсервіс незамінним інструментом для сучасних бізнесів, що прагнуть підвищити свою ефективність і продуктивність. Trackabi дає змогу бізнесам отримувати детальні звіти про використання робочого часу, що допомагає ідентифікувати найбільш ефективні процеси та виявляти області, де можливо покращити продуктивність. Зокрема, компанії можуть аналізувати, як витрачається час у різних проєктах, відстежувати прогрес у виконанні завдань та визначати, які співробітники потребують додаткової підтримки чи навчання. Завдяки цим функціям, Trackabi стає потужним інструментом для управління ресурсами та оптимізації робочих процесів у будь-якому бізнесі.

Be Focused Pro – застосунок, який ґрунтується на методиці Pomodoro, розділяючи час на періоди активності та короткі перерви для покращення концентрації та продуктивності [23] (рисунок 1.4).

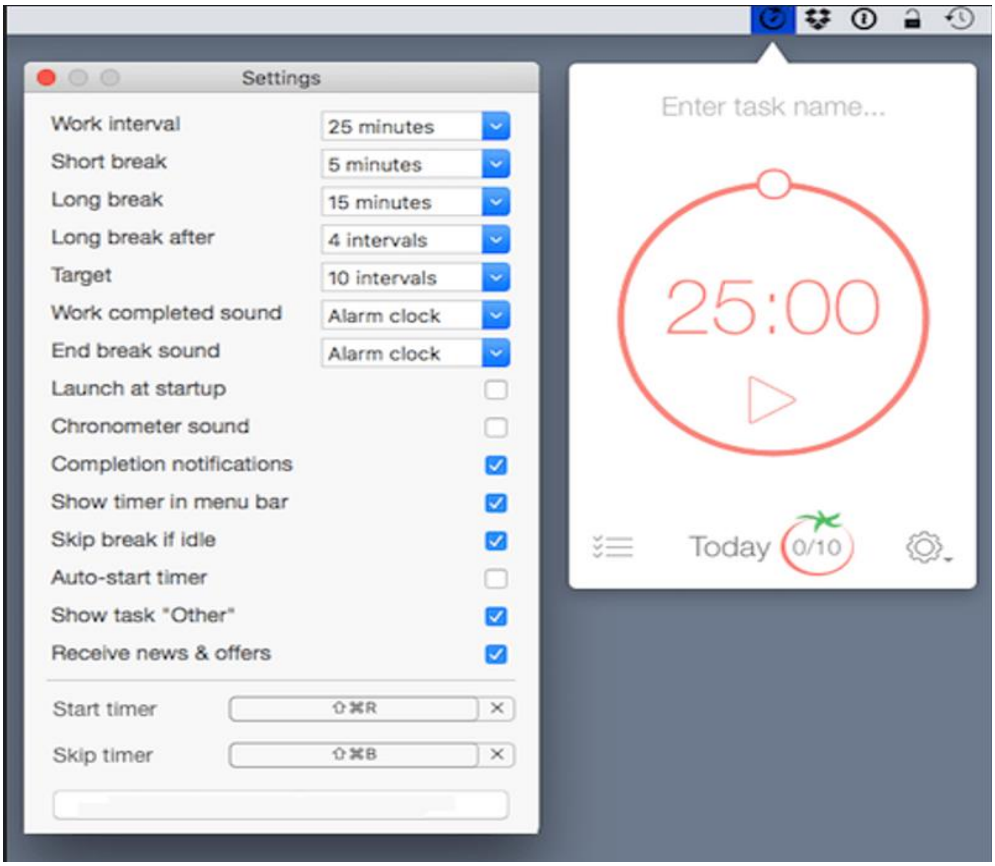


Рисунок 1.4 – Вигляд застосунку Be Focused Pro [24]

Додатково, варто звернутися до досліджень, які акцентують увагу на психологічних та фізіологічних аспектах ефективного планування розпорядку дня. У статті [25] розглядається вплив ефективного управління часом на академічні досягнення студентів. Дослідження показує, що правильне планування розпорядку дня може впливати на загальний успіх та здоров’я студентів. ІАД може бути використаний для виявлення патернів ефективного часового управління серед студентів.

Також важливим є розгляд додаткових аспектів, таких як кібербезпека та конфіденційність даних у програмах для планування розпорядку дня. У роботі [26] детально досліджується проблема забезпечення безпеки та конфіденційності даних користувачів.

Автори підкреслюють, що забезпечення надійного захисту даних є критично важливим для збереження довіри користувачів до цих програм. Зокрема, дослідження аналізує різноманітні загрози, з якими можуть стикатися користувачі мобільних застосунків для планування часу, а також розглядає методи захисту від них. Розуміння цих аспектів дає змогу розробникам впроваджувати ефективні механізми безпеки, що захищають особисту інформацію користувачів від несанкціонованого доступу і витоку даних.

Зведення наукових досліджень та аналіз наявних програмних рішень підкреслює необхідність пошуку новаторських підходів, які враховують сучасні виклики управління часом та сприяють покращенню продуктивності та задоволення користувачів. Створення нового методу у вигляді вебсервісу для планування розпорядку дня, заснованого на ІАД, виправдовується потребою в більш ефективних та персоналізованих інструментах, а також врахуванні особливостей користувачів.

1.4 Мета та задачі кваліфікаційної роботи

У результаті проведеного аналізу наявних методів розв’язання задачі планування розпорядку дня встановлено переваги та недоліки традиційних стратегій управління часом. Виявивши, що, хоча багато з них є працездатними, вони можуть бути недостатньо гнучкими та адаптивними до різноманітних умов та особистих особливостей кожного користувача.

Було розглянуто теоретичні підходи до використання ІАД для покращення тайм-менеджменту. Зокрема, методи ІАД надають можливості для прогнозування та адаптації до індивідуальних потреб користувача. Зокрема використання алгоритмів ІАД мови може поліпшити здатність системи розуміти контекст та інтегрувати текстову інформацію для точніших рекомендацій. Як наслідок, сформовано задачу покращення тайм-менеджменту засобами ІАД у вигляді вебсервісу, що дасть змогу успішно управляти часом в умовах сучасного жвавого ритму життя.

Метою кваліфікаційної роботи бакалавра є покращення планування розпорядку дня засобами ІАД у вигляді вебсервісу. Для досягнення мети кваліфікаційної роботи необхідно виконати такі завдання:

1. Розробити метод покращення планування розпорядку дня засобами ІАД.

2. Спроєктувати вебсервіс для покращення розпорядку дня на основі розробленого методу.

3. Виконати програмну реалізацію вебсервісу за розробленим методом покращення планування розпорядку дня.

4. Здійснити тестування розробленого методу покращення планування розпорядку дня та створеного на його основі вебсервісу.

5. Продемонструвати покращення планування розпорядку дня через впровадження розробленого методу.

Розв’язання вище вказаних завдань є ключовим для успішного розроблення методу покращення планування розпорядку дня та створення на його основі вебсервісу.

Розділ 2 Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних

2.1 Загальна схема методу покращення планування розпорядку

Розглянувши різні можливі підходи для покращення планування розпорядку дня, проаналізувавши їх переваги та недоліки, було вирішено використати генетичний алгоритм для покращення планування розпорядку дня. Основна перевага генетичних алгоритмів полягає в їхній здатності до пошуку оптимальних рішень у складних просторах можливих варіантів. Вони можуть ефективно працювати як і з великими об’ємами даних так і з малими, і враховувати багато факторів, що впливають на прийняття рішень.

В межах розробленого методу було реалізовано такий генетичний алгоритм для покращення планування розпорядку дня та впроваджено у вебсервіс. Основні кроки методу для покращення планування розпорядку дня зображено на рисунку 2.1.

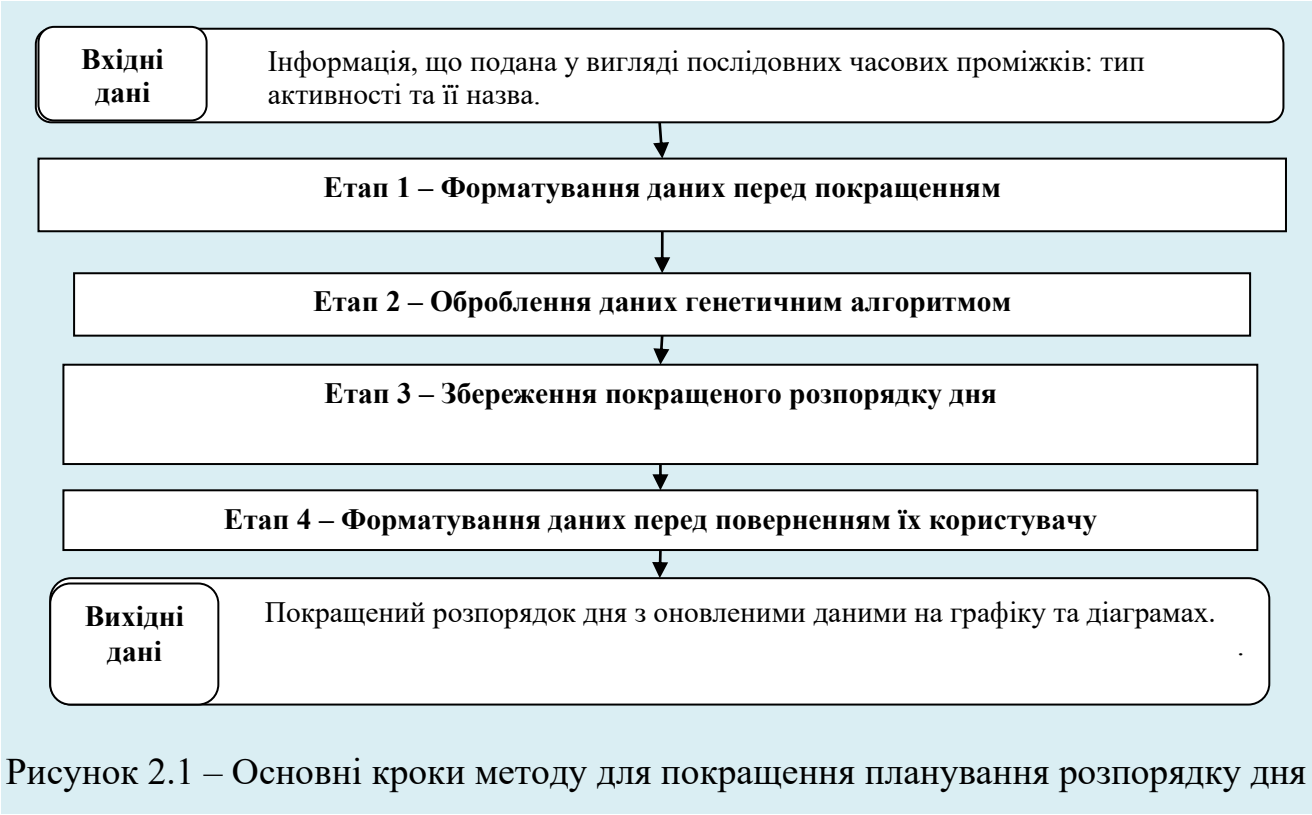


Рисунок 2.1 – Основні кроки методу для покращення планування розпорядку дня

Далі детально описано кожен з етапів.

Етап 1 – Форматування даних перед покращенням. На цьому етапі вхідні дані, які містять інформацію про поточний розклад користувача, перетворюються у формат, придатний для оброблення генетичним алгоритмом. Вхідні дані включають часові проміжки та типи активностей, наприклад, робота, відпочинок, фізичні вправи тощо. Кожен індивід, що представляє один день, отримує свій генотип. Генотип – це структура, де кожен ген відповідає за певну активність, а його значення визначає конкретний час для цієї активності. Наприклад, ген може означати “роботу” з 9:00 до 11:00, або “відпочинок” з 14:00 до 15:00. Це форматування забезпечує стандартизацію даних, що робить їх придатними для подальшого оброблення генетичним алгоритмом.

Етап 2 – Оброблення даних генетичним алгоритмом. Процес покращення розкладу починається зі створення початкової популяції розкладів. Кожен індивід у популяції представляє можливий варіант розпорядку дня. Спочатку генетичний алгоритм генерує розклади з різними часовими проміжками активностей наданих користувачем, які становлять початкову популяцію. Кожен розклад (індивід) оцінюється за допомогою функції пристосованості. Функція пристосованості враховує різні критерії, такі як загальна енергія та ефективність дня, з метою визначення якості кожного розкладу. Наприклад, високоефективні розклади отримують вищі оцінки, тоді як менш ефективні – нижчі. Після оцінки всіх індивідів популяції відбувається процес селекції, де відбираються найкращі розклади для переходу до наступного покоління. Відібрані індивіди здійснюють обмін генетичним матеріалом через кросовер. Після кросовера інколи відбуваються мутації, де деякі гени можуть випадково змінювати свої значення, що додає різноманітності у популяцію та допомагає уникнути застою на локальних максимумах. Цей цикл створення нових поколінь продовжується до тих пір, поки не буде досягнуто критеріїв зупинки. Це може бути досягнення заданої кількості ітерацій або досягнення певного рівня пристосованості, коли розклади більше не покращуються значною мірою. Результатом цього етапу є

найкращий можливий розпорядок дня, що забезпечує максимальну ефективність та енергію.

Етап 3 – Збереження покращеного розпорядку дня. Коли генетичний алгоритм знаходить оптимальний розпорядок дня, цей розклад зберігається в базі даних (БД). Збереження оновленого розкладу гарантує, що користувач має доступ до найефективнішого плану дня, створеного алгоритмом. Це також дає змогу користувачеві повернутися до цього розкладу в майбутньому або використовувати його як основу для подальшого налаштування і покращення.

Етап 4 – Форматування даних перед поверненням їх користувачу. На останньому етапі дані оновленого розпорядку дня перетворюються у формат, який забезпечує їх коректне відображення на веб-сторінці. Цей етап включає підготовку даних для зручного перегляду та взаємодії з ними користувачем. Оновлений розклад представляється у вигляді, що легко читається, наприклад, у формі таблиці або календаря. Це дає змогу користувачу бачити всі свої заплановані активності на день, ефективно планувати час та керувати своїми завданнями. Завдяки такому підходу користувач отримує інструмент для більш ефективного управління своїм часом, що сприяє підвищенню продуктивності та загальної задоволеності життям.

Отже, було розроблено метод покращення планування розпорядку дня засобами ІАД у вигляді покрокового виконання основних етапів.

2.2 Особливості отримання вхідної інформації для покращення планування розпорядку дня

Отримання вхідної інформації для покращення планування розпорядку дня відіграє ключову роль у створенні ефективної системи. Процес отримання вхідної інформації для покращення планування розпорядку дня включає додавання бажаних активностей протягом дня користувачем, які потім формуються у список та надсилаються на оброблення на бекенд.

Спочатку варто розглянути принцип роботи обміну даними між сервером та клієнтом. Обмін даними між сервером та клієнтом здійснюється через JavaScript Object Notation (JSON) (рисунок 2.2).

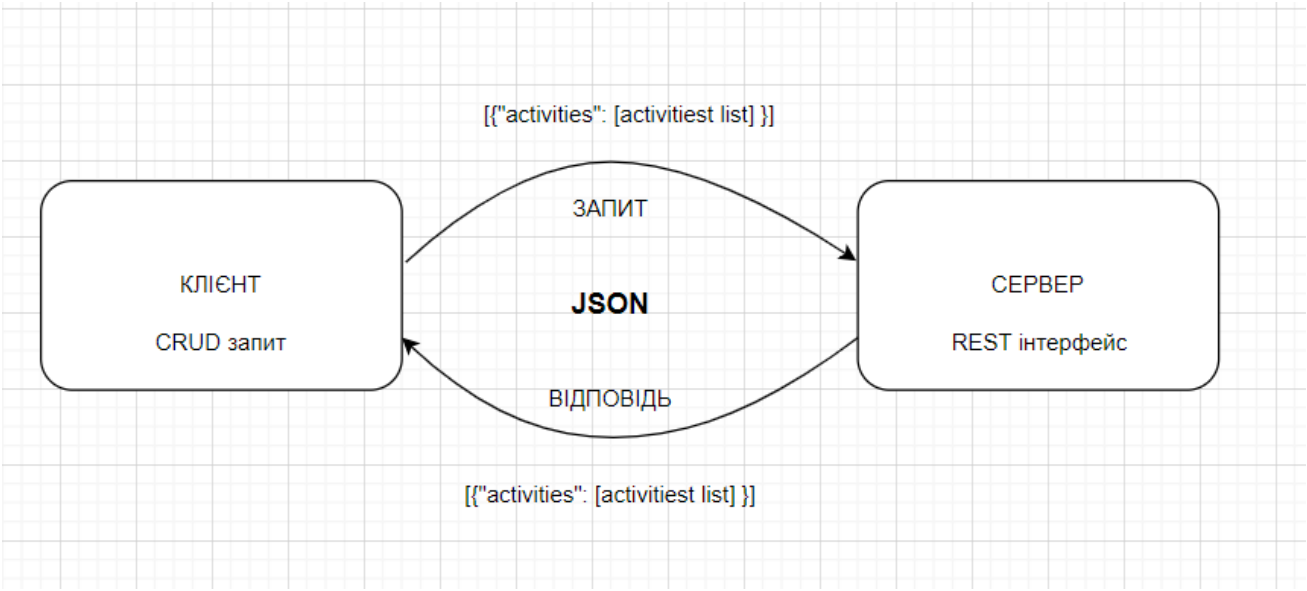


Рисунок 2.2 – Принцип обміну даними між клієнтом і сервером за допомогою JSON формату

JSON є легким текстовим форматом, який використовується для представлення структурованих даних на основі синтаксису об’єктів JavaScript. Завдяки своїй простоті та читабельності, JSON став стандартом для обміну даними між сервером і клієнтом. Принцип обміну даними через JSON полягає в наступному: спочатку клієнтська частина формує запит до сервера, структурований у вигляді JSON-об’єкта. Цей об’єкт може містити різноманітні дані, необхідні для виконання запиту, такі як інформація про користувача, параметри активностей тощо. Далі клієнтська частина використовує CRUD HTTP-запити (GET, POST, PUT, DELETE) для відправлення JSON-даних на сервер. Найчастіше використовується метод POST для надсилання даних, оскільки він дає змогу відправляти великі об’єми даних у тілі запиту. Серверна частина приймає запит, розпаковує JSON-дані та обробляє їх відповідно до логіки застосунку. Це може включати виконання алгоритмів обчислення, аналізу, збереження даних у БД тощо. Сервер, отримавши JSON-дані, проводить

необхідні обчислення, наприклад, оптимізацію розкладу дня користувача за допомогою генетичного алгоритму. Після оброблення даних сервер формує відповідь також у форматі JSON і надсилає її назад клієнту. Відповідь може містити оновлені дані, результати обчислень, статус виконання запиту тощо. Клієнтська частина приймає JSON-відповідь від сервера, розпаковує її та оновлює інтерфейс користувача відповідно до отриманих даних. Це може включати оновлення графіків, таблиць, текстових полів тощо. Таким чином, обмін даними між сервером та клієнтом через JSON забезпечує ефективну та зручну передачу інформації, дозволяючи легко інтегрувати різні частини вебсервісу та забезпечити їхню взаємодію.

Користувач заходить на сайт, де йому пропонується ввести інформацію про свої щоденні активності (рисунок 2.3).

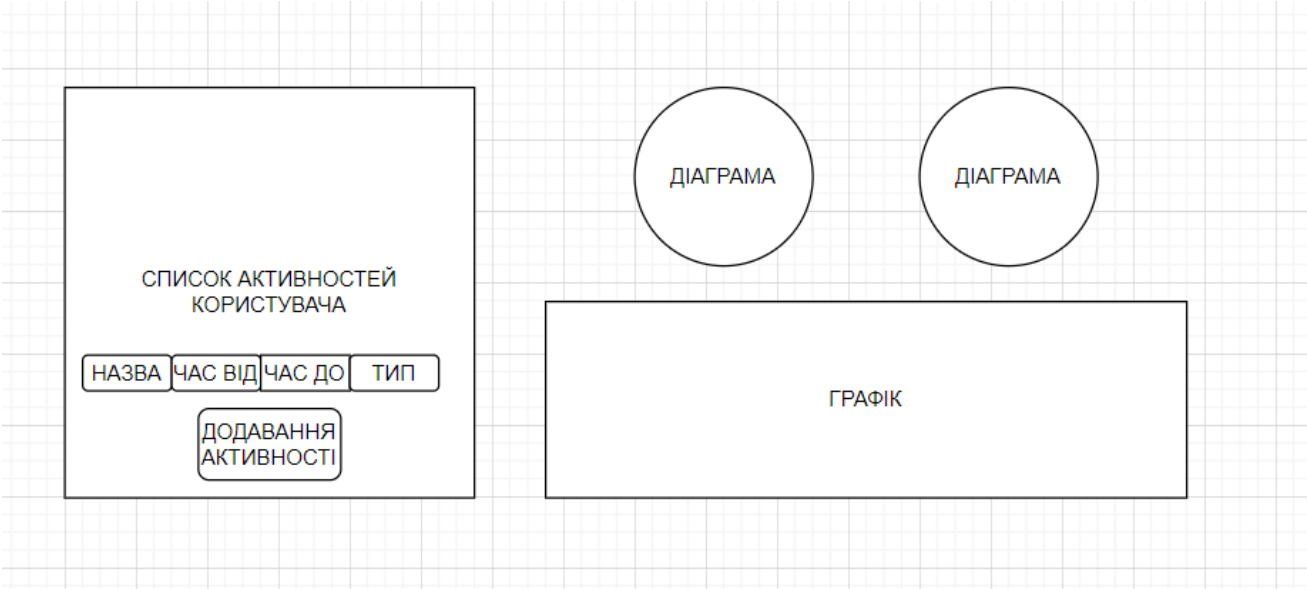


Рисунок 2.3 – Приклад отримання даних від користувача

Кожна активність, яку він вводить, може містити деталі, такі як назва активності, час початку та закінчення, а також тип активності, а саме – сон, робота, спорт, відпочинок та інше. Ці дані є ключовими для розуміння структури розпорядку дня користувача. Вони дають змогу системі аналізувати режим його дня, визначати періоди активності та відпочинку, а також розрізняти між різними видами діяльності.

Коли користувач натискає кнопку "update" (рисунок 2.4), відбувається наступне: всі введені користувачем активності, надсилаються на сервер для подальшого оброблення.

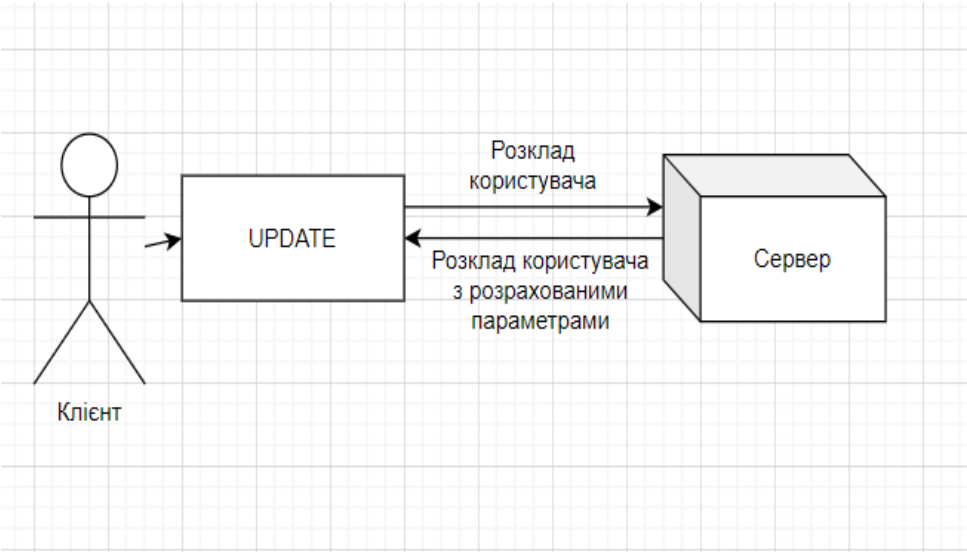


Рисунок 2.4 – Маніпуляція даними користувача при натисканні кнопки update

Сервер приймає ці дані від клієнта та починає необхідні обчислення, аналізуючи кожен часовий проміжок різних категорій активностей, таких як сон, робота, спорт, відпочинок тощо. Кожна з цих категорій ретельно аналізується для визначення їхнього впливу на загальний розпорядок дня користувача. Далі сервер проводить розрахунки, щоб визначити загальні параметри енергії та ефективності дня. Це включає обчислення, які враховують взаємозв’язок між різними видами діяльності та їх вплив на енергію користувача. Наприклад, оптимальний час для роботи чи відпочинку може змінюватися залежно від часу початку та закінчення інших активностей. Розрахунки також враховують рекомендації щодо тривалості сну та фізичної активності, щоб забезпечити максимальну продуктивність і здоров’я користувача. Після завершення оброблення сервер надсилає клієнту оновлені дані щодо розкладу дня, а також оновлені параметри енергії та ефективності.

Клієнтська частина вебсервісу оновлюється, щоб відобразити ці оновлені дані та результати аналізу. Графіки та діаграми також оновлюються, забезпечуючи користувачу візуальне представлення свого оптимізованого

розпорядку дня. Це дає змогу користувачу легко зрозуміти, які зміни були внесені і як вони впливають на загальну ефективність і рівень енергії протягом дня. Користувач може переглядати ці дані в реальному часі, вносячи подальші корективи в свій розпорядок дня для досягнення найкращих результатів.

Коли користувач натискає кнопку "improve" (рисунок 2.5), відбувається наступне: всі введені користувачем активності надсилаються на сервер для подальшої оброблення та оптимізації. Сервер приймає ці дані та починає використовувати генетичний алгоритм для покращення розпорядку дня користувача.

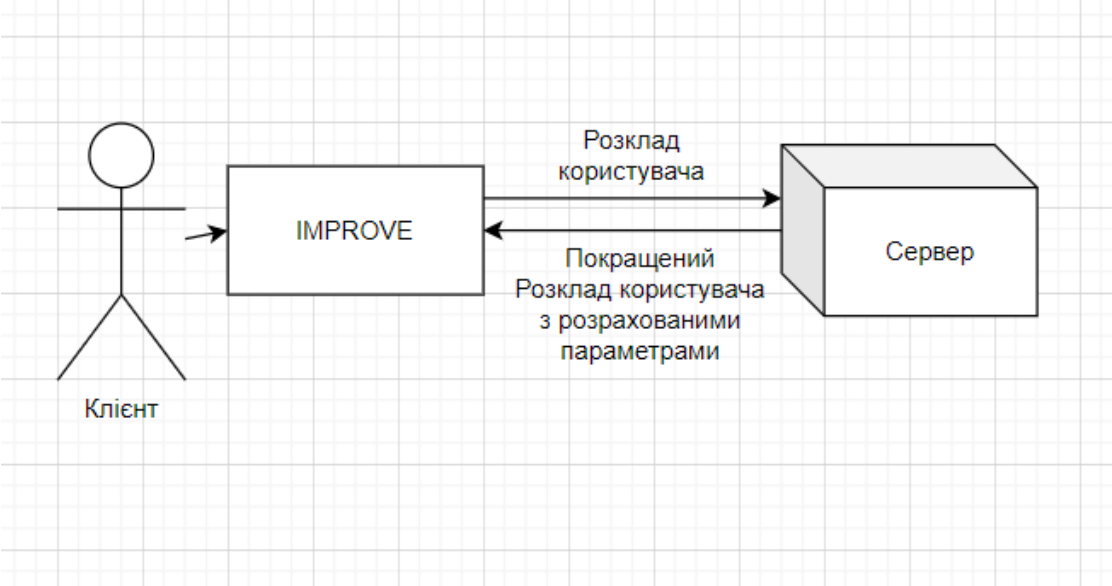


Рисунок 2.5 – Маніпуляція даними користувача при натисканні кнопки improve

Процес оптимізації включає кілька ключових етапів. Спочатку алгоритм генерує початкову популяцію можливих розкладів, які потім піддаються оцінці на основі заданих критеріїв, таких як продуктивність та енергія. Далі відбувається відбір найкращих розкладів для створення нових поколінь шляхом комбінування та мутації, що дає змогу алгоритму досліджувати різні варіанти і знаходити оптимальні рішення. Кожне нове покоління розкладів аналізується і порівнюється з попередніми, щоб визначити, які зміни призводять до покращення ефективності. Після завершення оптимізації сервер повертає назад користувачеві оновлені дані щодо їхнього розкладу дня. Ці дані включають

змінені часові проміжки активностей, які були визначені як найефективніші для підвищення продуктивності та енергії користувача. Клієнтська частина вебсервісу оновлюється, щоб відобразити ці оновлені дані та результати оптимізації. Новий розклад дня з'являється на екрані користувача, забезпечуючи візуальне представлення оптимізованого плану. Це дає змогу користувачу легко переглядати та аналізувати зміни, зроблені алгоритмом, а також наочно бачити, як ці зміни покращують ефективність та енергетичний рівень протягом дня. Користувач може інтерактивно взаємодіяти з новим розкладом, вносячи додаткові корективи. Важливою перевагою такого підходу є можливість персоналізації планування розпорядку дня відповідно до індивідуальних потреб та вимог кожного користувача. Система може враховувати різні типи активностей та їх вплив на енергію та продуктивність, а також враховувати особисті уподобання щодо часу проведення певних видів діяльності.

Діаграма на рисунку 2.5(зліва) відображає загальні частки часу, відведені на кожен тип активності протягом дня. Кожен сегмент пирога представляє відсоткове співвідношення часу, який був відведений на певний тип активності, порівняно з загальним часом дня. Діаграма на рисунку 2.6(справа) відображає загальні значення енергії та ефективності користувача протягом дня. Вона може включає дві заокруглених полоси різного кольору: одна для енергії та інша для ефективності. Кожна полоса показує значення цих загальних параметрів розпорядку дня.

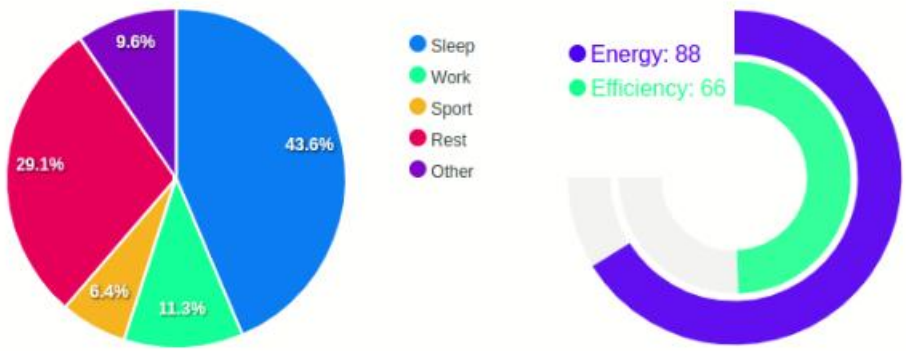


Рисунок 2.6 – Діаграми розпорядку дня користувача

Графік енергії та ефективності для кожного часового проміжку (рисунок 2.7) є ключовим інструментом для візуалізації продуктивності користувача. Кожна точка на графіку представляє значення енергії та ефективності для певного проміжку часу. Ці значення розраховуються на сервері на основі даних про активності користувача і повертаються у вигляді послідовного списку значень. Користувач може бачити, як змінюються їхні рівні енергії та ефективності протягом дня, що допомагає краще розуміти, коли вони є найбільш продуктивними і енергійними.

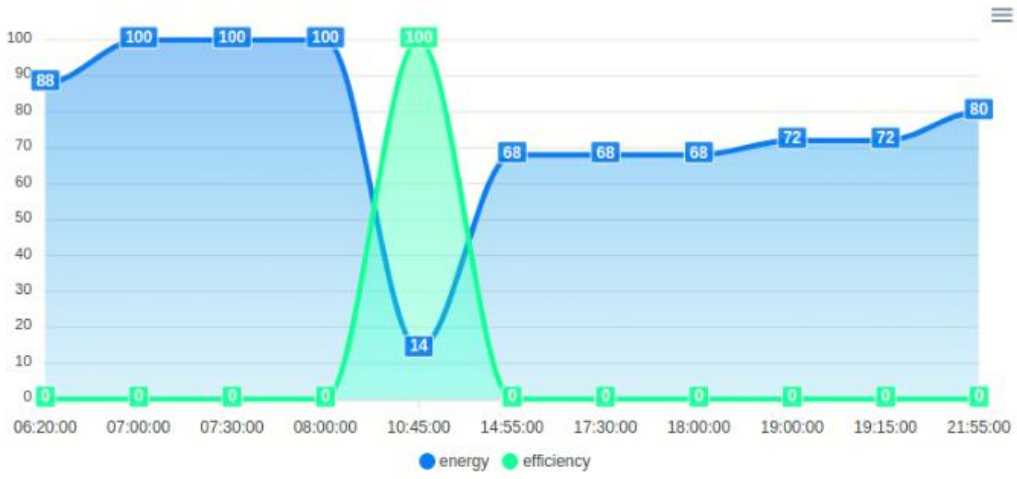


Рисунок 2.7 – Графік параметрів для кожного часового проміжку розпорядку дня користувача

Для кращої візуалізації всі ці точки поєднуються різнокольоровими лініями, де одна позначає рівень енергії, а інша – рівень ефективності. Ці лінії дають змогу легко відслідковувати динаміку змін протягом дня. Наприклад, користувач може побачити, як певні активності, такі як робота, спорт або відпочинок, впливають на їхню енергію та ефективність. Це дає змогу виявити найбільш ефективні періоди для виконання певних завдань і, відповідно, краще планувати свій розпорядок дня.

Підсумовуючи, отримана інформація про активності користувача подається у формі списку, графіка та діаграм (рисунок 2.5) для кращого сприйняття даних користувачем. Після отримання даних про активності

користувача, вони обробляються генетичним алгоритмом для пошуку оптимального розпорядку дня. Генетичний алгоритм використовуються для аналізу та оптимізації розкладу, враховуючи індивідуальні потреби користувача. Він шукає найкращі комбінації розмірів часових проміжків активностей, які забезпечують оптимальний баланс між роботою, відпочинком та іншими аспектами життя. Такий підхід дає змогу покращити організацію часу та забезпечити більш ефективне використання ресурсів людини.

2.3 Особливості реалізації методу для покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу

Реалізація методу для покращення планування розпорядку дня з використанням ІАД у формі вебсервісу вже є однією з особливостей інформаційної системи (ІС). Адже таке рішення є перспективним напрямом, що відкриває широкі можливості для користувачів у впорядкуванні їхнього часу та підвищенні продуктивності. Такий підхід дає змогу зручно та ефективно користуватися системою з будь-якого пристрою, що має доступ до Інтернету, надаючи гнучкість та доступність для користувачів у будь-який час.

Наступною особливістю реалізації методу для покращення планування розпорядку дня у формі вебсервісу є його безпека, яка забезпечується за допомогою Spring Security. Безпека є критичним аспектом будь-якої веб-системи, особливо коли йдеться про оброблення особистої інформації користувачів та конфіденційних даних. Spring Security – це потужний інструмент для забезпечення безпеки вебзастосунків у середовищі Java. Він надає різноманітні можливості для автентифікації, авторизації, захисту від атак та управління доступом, що забезпечує надійний рівень захисту для вебсервісу.

Однією з ключових функцій Spring Security є можливість налаштування автентифікації користувачів. За допомогою різних методів автентифікації, таких як форма входу, базова автентифікація або OAuth, користувачі можуть безпечно

авторизуватися в системі, забезпечуючи захист від несанкціонованого доступу. Крім того, Spring Security надає можливості для налаштування авторизації, що дає змогу обмежити доступ до різних частин системи для різних груп користувачів. Це дає змогу забезпечити приватність та конфіденційність даних, запобігаючи несанкціонованому доступу до важливої інформації. Spring Security також забезпечує захист від різних видів атак, таких як перехоплення сесій, внедрення коду та переповнення буфера. Він надає механізми для фільтрації та перевірки вхідних даних, а також захисту від небезпечних запитів, що забезпечує безпеку системи навіть у випадку непередбачених сценаріїв використання.

Ще однією особливістю реалізації методу для покращення планування розпорядку дня у формі вебсервісу є використання генетичного алгоритму для оптимізації розкладу користувача. Генетичні алгоритми є потужним інструментом для пошуку оптимальних рішень у складних задачах оптимізації, таких як планування розпорядку дня. У контексті планування розпорядку дня, генетичний алгоритм використаний для пошуку найбільш ефективного розподілу часу між різними видами діяльностей користувача. Однією з переваг використання генетичного алгоритму є його здатність до адаптації та еволюції. Він може враховувати змінні умови, уподобання користувача та нові обмеження, пристосовуючи розклад до них та надаючи користувачеві найоптимальніший розподіл часу в будь-який момент. Генетичний алгоритм також може враховувати різні критерії оптимальності, такі як максимізація продуктивності, мінімізація стресу чи забезпечення балансу між різними видами діяльностей. Це дає змогу користувачеві вибрати той розклад, який найбільш відповідає його потребам та цілям.

Додатковою особливістю є можливість використання технологій управління ресурсами, таких як Docker для контейнеризації або Kubernetes для оркестрації. Це дає змогу зручно масштабувати та керувати ресурсами вебсервісу в залежності від навантаження.

Важливою особливістю є також можливість моніторингу та аналізу роботи вебсервісу. Збір та аналіз логів, моніторинг метрик продуктивності та

роботи моделі дають змогу операторам системи своєчасно виявляти проблеми та вдосконалювати роботу системи.

Узагальнюючи перераховані вище особливості, можна сказати, що запропонований метод покращення розпорядку дня у вигляді вебсервісу забезпечує доступність, безпеку, масштабованість та ефективність, що робить його привабливим для широкого кола користувачів.

2.4 Проєктна архітектура системи та взаємозв'язок компонентів

Метод для покращення планування розпорядку дня, реалізований у формі вебсервісу, використовує мікросервісну архітектуру для створення розгалуженої та ефективної системи, яка забезпечує гнучкість, масштабованість та швидкість реакції на зміни. Мікросервісна архітектура дає змогу розбити весь функціонал системи на невеликі, самостійні сервіси, кожен з яких відповідає за виконання певної обмеженої функціональності.

Вебсервіс для покращення планування розпорядку дня складається з трьох основних мікросервісів: fe-wizard, be-wizard та be-planner (рисунок 2.8).

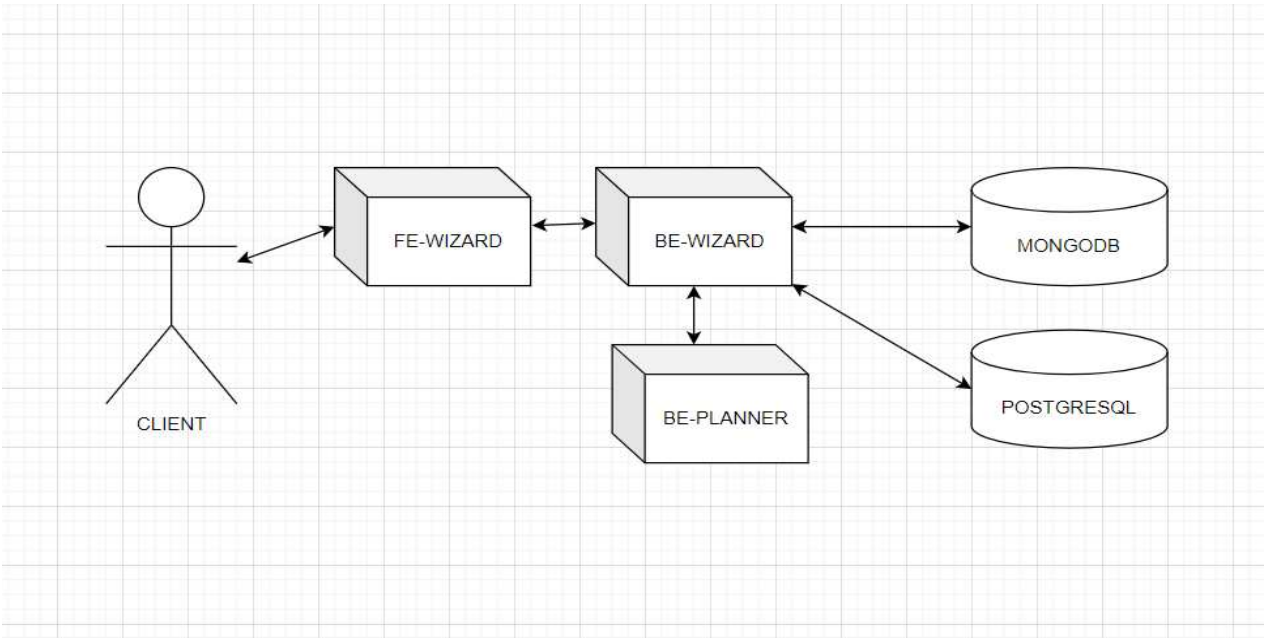


Рисунок 2.8 – Мікросервісна архітектура методу покращення планування розпорядку дня у вигляді вебсервісу

1. fe-wizard (фронтенд). Цей мікросервіс представляє собою клієнтську частину вебсервісу, яка відповідає за інтерфейс користувача. Використовуючи фреймворк React JS, fe-wizard надає користувачам можливість реєструватися, увійти в свій акаунт та взаємодіяти зі своїм розпорядком дня. Крім того, через цей мікросервіс користувач може надсилати свій розпорядок дня на бекенд для подальшого покращення.

2. be-wizard (бекенд). Цей мікросервіс приймає запити від fe-wizard та відповідає за оброблення цих запитів. Використовуючи Java Spring Boot та Spring Security, be-wizard здійснює операції авторизації, взаємодії з БД та надсилає запит на мікросервіс be-planner для покращення розпорядку дня користувача. Після оброблення отриманого від be-planner відповіді, be-wizard повертає її на фронтенд.

3. be-planner (бекенд). Цей мікросервіс відповідає за покращення розпорядку дня користувача. Використовуючи Java Spring Boot, be-planner реалізує генетичний алгоритм, який аналізує інформацію про активності користувача та генерує оптимальний розпорядок дня. Крім того, be-planner приймає запити від be-wizard та надсилає йому відповідь із покращеним розпорядком дня.

Кожен з цих мікросервісів відповідає за свої функціональності та взаємодіє з іншими компонентами системи, утворюючи комплексний та ефективний вебсервіс для покращення планування розпорядку дня.

Одним із ключових аспектів взаємозв’язку компонентів є механізми комунікації між мікросервісами. У архітектурі вебсервісу це відбувається за допомогою Hypertext Transfer Protocol (HTTP) та RESTful API. Кожен мікросервіс має свої власні API точки доступу, через які він надсилає та отримує дані від інших сервісів. Це дає змогу створювати легко розширювані та модульні системи, в яких окремі компоненти можуть бути замінені або модифіковані без зміни інших компонентів.

Крім того, важливою частиною архітектури є забезпечення масштабованості та надійності системи. Для цього використовується контейнеризація з Docker (рисунок 2.9).

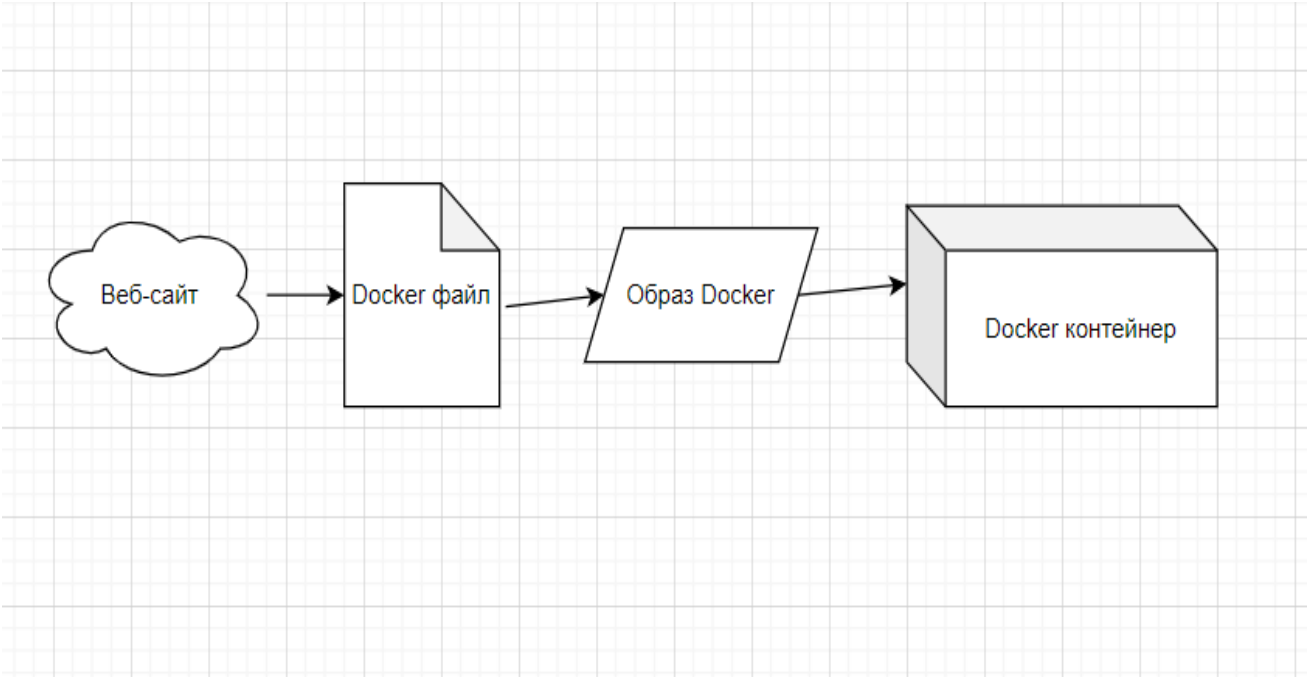


Рисунок 2.9 – Процес контейнеризації в Docker

Контейнеризація дає змогу ізолювати кожен мікросервіс у власному середовищі, що сприяє більш ефективному використанню ресурсів та забезпечує консистентність у розгортанні та масштабуванні. Такий підхід дає змогу легко керувати розподіленою інфраструктурою та швидко відгукуватися на зміни у навантаженні. Крім того, Docker забезпечує стандартизацію середовища виконання, що спрощує розробку, тестування та розгортання нових версій мікросервісів. Такий підхід дає змогу забезпечити стабільну та надійну роботу системи навіть при великому обсязі трафіку та рості числа користувачів.

Також у веб сервіс було впроваджено Spring Security (рисунок 2.10). При отриманні HTTP-запиту від клієнта сервер спершу направляє його до Security Filter Chain, що складається з ряду фільтрів, що обробляють такі запити. Ці фільтри виконують різноманітні операції, такі як перевірка наявності користувача, перенаправлення на сторінку входу чи перевірка прав доступу.

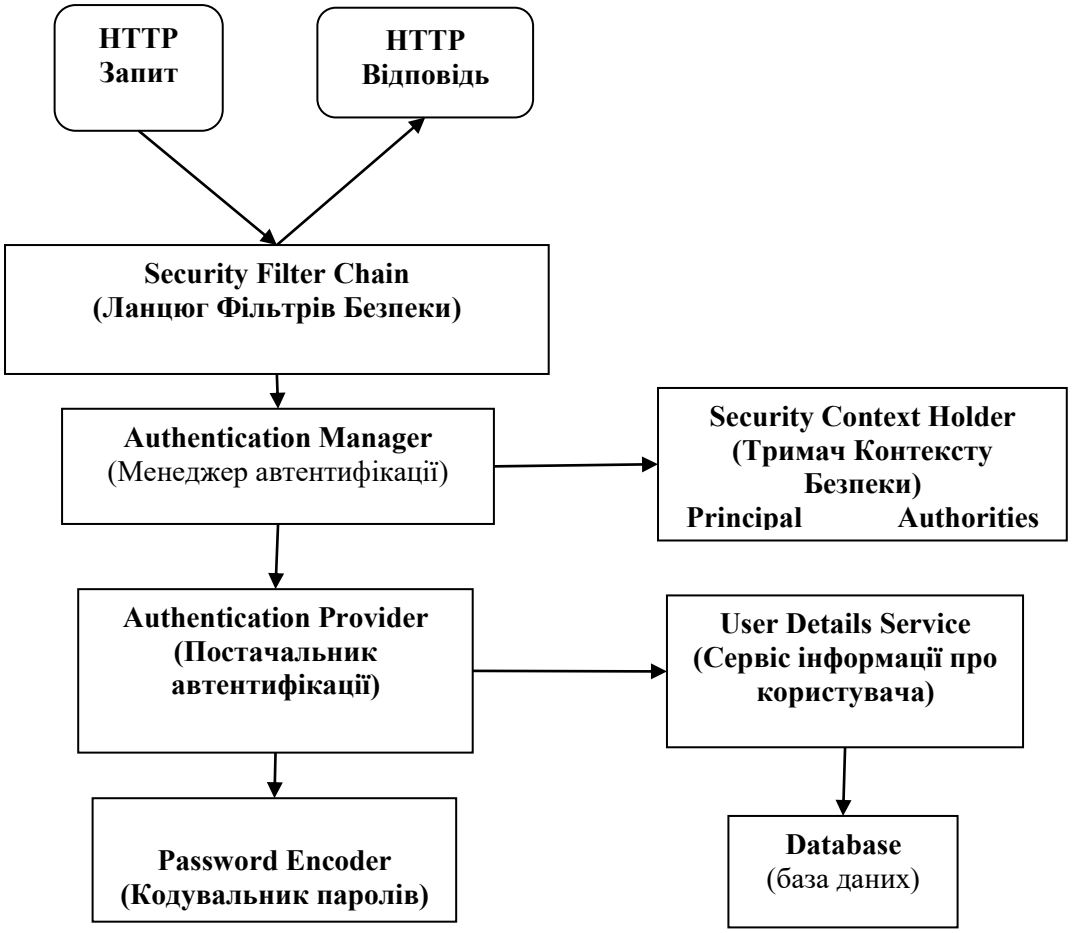


Рисунок 2.10 – Архітектура Spring Security вебсервісу для покращення планування розпорядку дня

Основним компонентом, який відповідає за перевірку автентифікації, є Authentication Manager. Він приймає дані автентифікації та намагається підтвердити їхню вірність. Authentication Manager користується Authentication Provider для перевірки цих даних, можливо, шляхом порівняння їх з різними джерелами, такими як БД іншими системи.

З метою забезпечення безпеки паролів, Spring Security рекомендує використовувати кодувальники паролів, що зберігають паролі користувачів у зашифрованому вигляді для забезпечення безпеки в разі витоку даних. Authentication Provider може також звертатися до User Details Service для отримання додаткової інформації про користувачів. Після успішної автентифікації користувача, його основна інформація, така як ідентифікатор та

список ролей, зберігається в SecurityContextHolder і може використовуватися для подальшої авторизації та доступу до ресурсів. Загалом, Spring Security надає гнучкі та потужні інструменти для забезпечення безпеки вебзастосунків, включаючи автентифікацію, авторизацію та захист від різноманітних видів атак.

Вебсервісу для покращення планування розпорядку дня використовує дві БД для забезпечення ефективної роботи та зручного зберігання важливої інформації. PostgreSQL містить інформацію про користувачів, включаючи їхні унікальні ідентифікатори, імена, електронні адреси та захешовані паролі (рисунок 2.11).

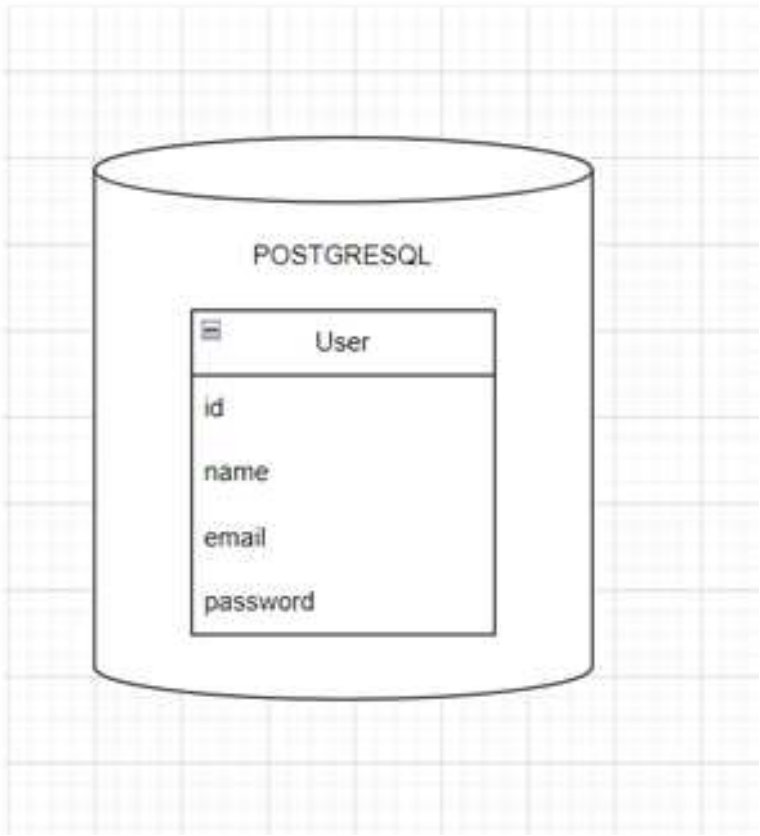


Рисунок 2.11 – БД PostgreSQL вебсервісу

Ім’я користувача використовується для персоналізації взаємодії з системою, а електронна адреса є основним ідентифікатором для здійснення автентифікації та взаємодії з системою. Захешований пароль забезпечує захист облікових записів користувачів.

MongoDB використовується для зберігання розкладів користувачів та деталей щоденної активності (рисунок 2.12). Кожен день користувача представлений документом, що містить його унікальний ідентифікатор (електронну адресу), список активностей на день, параметри енергії та ефективності для кожної активності, а також загальну ефективність та енергію користувача протягом дня.

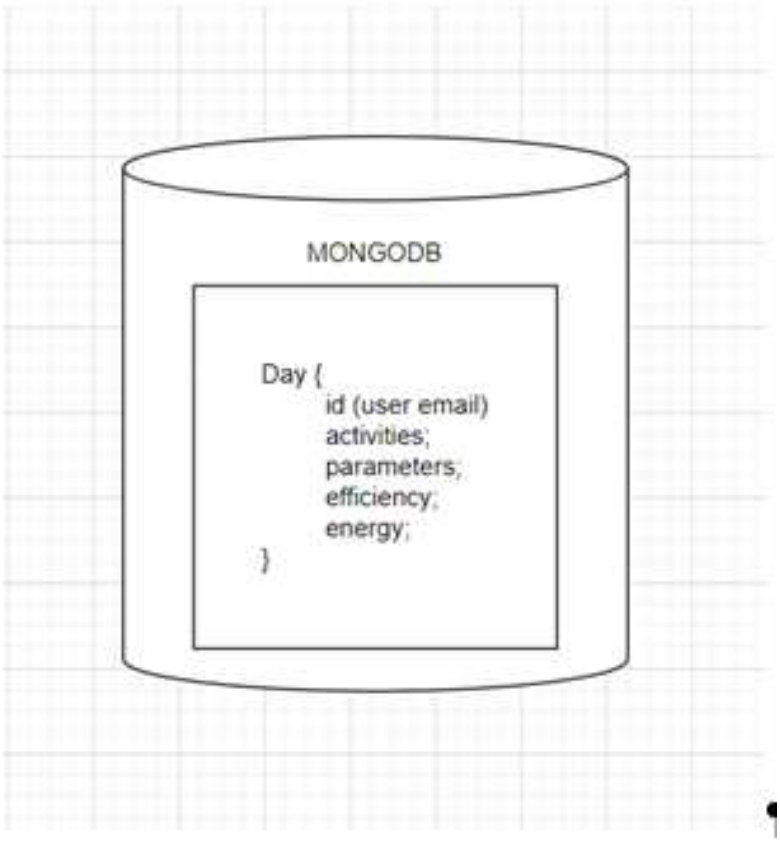


Рисунок 2.12 – БД MongoDB вебсервісу.

У MongoDB дані про розклади користувачів зберігаються у вигляді документів, що дає змогу зберігати детальну інформацію про кожен день. Такий підхід дає змогу зберігати та аналізувати дані про активності користувачів з високою точністю та ефективно використовувати їх для покращення планування розпорядку дня.

У підсумку, було розроблено архітектуру вебсервісу для покращення планування розпорядку дня. Оптимізована комунікація між мікросервісами,

ефективні методи оброблення даних, а також моніторинг та керування взаємодією сприяють високій продуктивності системи.

2.5 Висновки до розділу 2

У розділі 2 було розроблено метод покращення планування розпорядку дня на основі використаного генетичного алгоритму. Наведено детальний опис особливостей отримання вхідної інформації від користувача для подальшого покращення розпорядку дня. Важливою складовою є збір інформації про активності користувача, таких як тривалість сну, роботи, відпочинку та інших видів діяльності. Ці дані дають змогу розробленому рішенню аналізувати звички та пріоритети користувача та пропонувати індивідуалізовані рекомендації для покращення його розпорядку дня.

Також у розділі спроєктовано архітектуру системи, яка призначена для програмної реалізації методу покращення розпорядку дня. Вебсервіс був побудований на мікросервісній архітектурі, що дало змогу забезпечити гнучкість та масштабованість системи. Кожен компонент, від фронтенду до бекенду, ретельно розроблений та взаємодіє з іншими частинами системи для забезпечення швидкого та надійного функціонування.

Розділ 3 Програмна реалізація вебсервісу для покращення планування розпорядку

3.1 Визначення шляхів дослідження та засобів створення програмного забезпечення

Під час роботи над методом покращення планування розпорядку дня було визначено підходи до дослідження та розроблення програмного забезпечення. Для успішного розроблення програмного забезпечення необхідно було вибрати відповідну методологію дослідження, яка дасть змогу здійснити ретельний аналіз функціональності та ефективності планування розпорядку дня. Основний акцент було зроблено на генетичних алгоритмах.

Щодо вибору програмної системи та її функціоналу, основним критерієм було забезпечення потрібного рівня функціональності та зручності використання для користувачів. Для цього було вирішено реалізувати вебсервіс, в якому основні функції включають реєстрацію користувача, та введення активностей користувача з параметрами (назва, час початку, час закінчення, тип), оновлення розкладу та його покращення за допомогою генетичного алгоритму. Такий вибір технологій дає змогу забезпечити необхідний рівень продуктивності та зручність використання.

Щодо стратегії тестування та перевірки функцій програмного забезпечення були використані ручні методи тестування. Ручні тести проводилися для оцінки інтерфейсу користувача та загальної зручності використання програми.

У контексті дослідження параметрів розпорядку дня, вимірювалися загальні ефективність та енергія користувача на основі різних видів активностей та їх часових проміжків. Для цього були використані відповідні метрики, які дозволяли об’єктивно оцінити результати дослідження та внести необхідні корективи для подальшого покращення програми.

3.2 Вибір засобів програмної реалізації розробленого методу

Для розроблення програмного забезпечення було обрано комплекс інструментів, що забезпечують ефективну розробку на всіх рівнях системи. Нижче наведено огляд вибраних засобів розроблення та їх роль у процесі створення програмного забезпечення.

Для розроблення бекенду системи, який ґрунтується на Java Spring Boot, було обрано середовище розроблення IntelliJ IDEA. Це потужне середовище для розроблення, яке надає широкі можливості для програмістів Java. Використання IntelliJ IDEA дає змогу підвищити продуктивність розробників завдяки розширеному набору інструментів для рефакторингу, автоматизації та налагодження коду.

Для фронтенду, розробленого на основі JavaScript, HTML і CSS, використовувався текстовий редактор Visual Studio Code. Це легкий, але потужний інструмент, який надає широкі можливості для редагування коду, автоматичного доповнення, інтеграції з системами керування версіями та розширеннями для підтримки різних технологій.

Для зберігання даних було обрано дві системи управління БД: MongoDB та PostgreSQL. MongoDB використовується для зберігання структурованих даних, таких як розклади користувачів та їх активності. Його гнучка схема дає змогу ефективно зберігати дані з різних типів активностей. PostgreSQL використовується для зберігання додаткової інформації, яка потребує складних зв’язків та транзакційної підтримки, наприклад, дані про користувачів та їх автентифікацію.

Підсумовуючи, було обрано такі засоби для розроблення програмного забезпечення: IntelliJ IDEA для розроблення бекенду на Java Spring Boot та Visual Studio Code для фронтенду на основі JavaScript, HTML і CSS. Для зберігання даних використовуються MongoDB та PostgreSQL, кожна з яких має свої переваги в залежності від характеру даних. Така комбінація забезпечує

ефективний процес розроблення програмного забезпечення з урахуванням його потреб та особливостей.

3.3 Структура та функціональне призначення програмних складових системи

У програмній архітектурі системи “Покращення планування розпорядку дня” використовуються різні модулі та компоненти, які спільно працюють для забезпечення функціональності продукту. На рисунку 3.1 можна побачити архітектуру та основні компоненти бекенд мікросервісів.

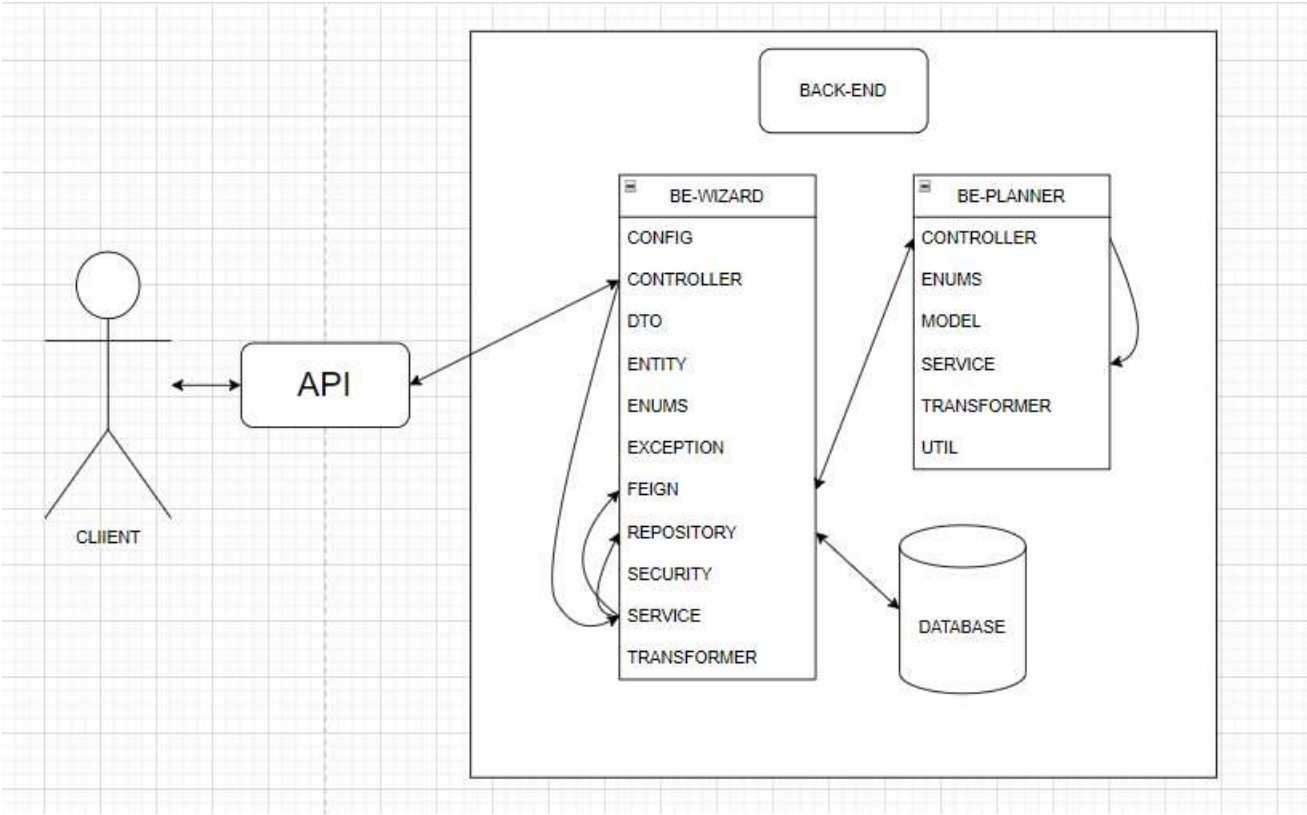


Рисунок 3.1 – Архітектура та зв’язок компонентів бекенд мікросервісів be-wizard та be-planner

Компонент config відіграє важливу роль у налаштуванні та конфігурації бекенд мікросервісів. Він містить конфігураційні файли та класи, які визначають різноманітні параметри, такі як налаштування з’єднання з БД, параметри

безпеки, порти, на яких слухаються HTTP-запити, та інші налаштування, необхідні для правильної роботи мікросервісів. Наприклад, в config можуть бути визначені параметри підключення до БД MongoDB та PostgreSQL, а також параметри безпеки, такі як ключі JSON Web Token (JWT) або налаштування.

Компонент controller відповідає за оброблення HTTP-запитів, які надходять до бекенд мікросервісів від фронтенду або інших клієнтів. Кожен контролер містить методи, що відповідають на різні типи запитів та виконують певні дії відповідно до цих запитів. Наприклад, контролер може мати метод для отримання списку активностей користувача за певний період часу або для додавання нової активності до розкладу користувача. Контролери також взаємодіють з іншими компонентами, такими як сервіси та репозиторії, для оброблення та збереження даних.

Data transfer object (DTO) – це класи, які використовуються для передачі даних між компонентами системи. Вони представляють об’єкти, які містять дані, які можуть бути передані через мережу або збережені у БД. DTO зазвичай використовуються для серіалізації даних у формати JSON або XML для передачі через HTTP-запити. Наприклад, DTO може представляти об’єкт активності з полями, такими як назва, час початку, час закінчення та тип.

Компонент entity представляє сутності або об’єкти, які зберігаються у БД. Вони відображають структуру даних та взаємозв’язки між ними. Класи entity відображають таблиці у БД та мають поля, що відповідають стовбцям у цих таблицях. Наприклад, сутність “Активність” може мати поля, такі як “назва”, “час початку”, “час закінчення” та “тип”.

Enums – це перерахування, які використовуються для визначення фіксованих наборів значень. Вони використовуються для представлення деяких обмежених або фіксованих наборів даних, таких як типи активностей (сон, робота, спорт, відпочинок тощо). Enums забезпечують зручний спосіб визначення та використання цих наборів значень у програмі.

Exception – це компонент, який відповідає за оброблення винятків та помилок в програмі. Він включає класи та компоненти, які допомагають

програмі реагувати на помилкові стани та виконувати відповідні дії для їхнього оброблення. Наприклад, виняток може бути викинути, якщо під час збереження активності в БД сталася помилка.

Feign – це компонент, який використовується для взаємодії з іншими мікросервісами або зовнішніми системами за допомогою HTTP-запитів. Він надає зручний спосіб викликати методи інших сервісів, використовуючи анотації та інші зручні засоби. Feign дає змогу зробити взаємодію між мікросервісами більш простою та ефективною.

Repository – це компонент, який відповідає за взаємодію з БД. Він містить методи для зчитування, запису, видалення та оновлення даних у БД. Repository використовується для виконання операцій з БД без прямої взаємодії з SQL-запитами, що робить роботу з БД більш простою та зручною.

Security – це компонент, який відповідає за забезпечення безпеки системи. Він включає в себе різні класи та компоненти, які відповідають за автентифікацію, авторизацію, захист від несанкціонованого доступу та інші аспекти безпеки. Security забезпечує захист даних та ресурсів системи від небажаних атак та доступу.

Service – це компонент, який містить бізнес-логіку програми. Він виконує різноманітні операції з даними та обробляє бізнес-логічні правила. Сервіси викликаються контролерами для виконання певних дій та взаємодію з іншими компонентами, такими як репозиторії та утиліти.

Model – це компонент, який відповідає за визначення структури та даних об’єктів, які використовуються в системі. Він включає класи, що представляють DTO, Entity та інші об’єкти, які використовуються для передачі даних, зберігання у БД та виконання інших операцій.

Transformer – це компонент, який відповідає за перетворення даних з одного формату в інший або з одного типу об’єкту в інший. Він забезпечує можливість зручного перетворення даних між різними компонентами системи та зовнішніми джерелами.

Util – це компонент, який містить допоміжні класи та утиліти, які використовуються в різних частинах системи. Він містить різноманітні службові функції та методи, які використовуються для вирішення різних задач, таких як оброблення дат та часу, робота з рядками та інші.

У додатку А зображена діаграма класів мікросервісу be-wizard, де можна побачити всі класи, їх поля та методи, а також взаємозв'язок між ними. Далі буде детальніше розглянуто певні класи цього мікросервісу та їх методи.

Розглянемо класи компоненту controller мікросервісу be-wizard (рисунок 3.2). Метод register контролера AuthenticationController відповідає за реєстрацію нових користувачів у системі. При отриманні запиту, він пробує виконати реєстрацію, обробляючи дані, які надійшли у вигляді об'єкта RegisterRequest. Якщо реєстрація вдалася, метод поверне ResponseEntity з об'єктом AuthenticationResponse, який містить токен доступу для нового користувача. У випадку, якщо користувач з вказаною електронною поштою вже є в системі, відбудеться оброблення помилки UserAlreadyExistException.

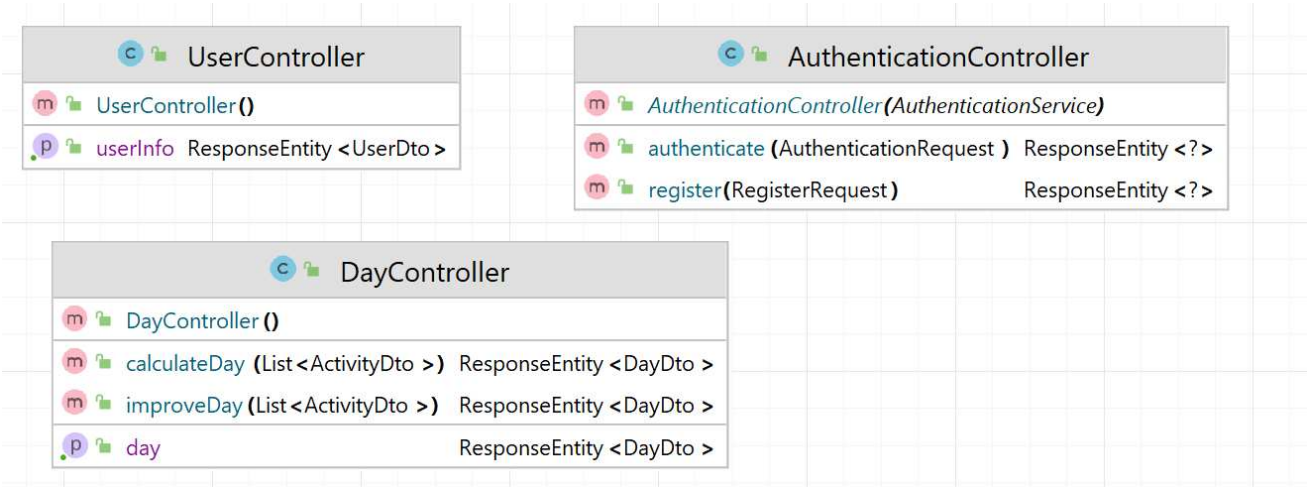


Рисунок 3.2 – Класи компоненту controller мікросервісу be-wizard

Метод authenticate контролера AuthenticationController відповідає за автентифікацію користувачів у системі. При отриманні запиту на автентифікацію, він спробує провести автентифікацію, обробляючи дані, які надійшли у вигляді об'єкта AuthenticationRequest. Якщо автентифікація пройшла успішно, метод поверне ResponseEntity з об'єктом AuthenticationResponse, який

містить токен доступу для автентифікованого користувача. У випадку неправильних облікових даних (наприклад, неправильний пароль або електронна пошта), відбудеться оброблення помилки.

Метод getDay контролера DayController призначений для отримання інформації про розпорядок дня користувача. При виклику цього методу відбувається отримання даних про активності користувача за допомогою сервісу dayService. Результат відправляється назад у вигляді ResponseEntity, який містить об’єкт DayDto з інформацією про розпорядок дня.

Метод improveDay контролера DayController призначений для покращення розпорядку дня користувача на основі отриманих активностей. При отриманні списку активностей, цей метод викликає сервіс dayService, який використовує генетичний алгоритм для оптимізації розпорядку дня. Після цього він повертає ResponseEntity з об’єктом DayDto, який містить покращений розпорядок дня користувача.

Метод calculateDay контролера DayController призначений для розрахунку розпорядку дня користувача на основі отриманих активностей. При отриманні списку активностей, цей метод викликає сервіс dayService, який виконує розрахунок розпорядку дня на основі введених даних. Після цього він повертає ResponseEntity з об’єктом DayDto, який містить розрахований розпорядок дня користувача.

Далі розглянемо класи компоненту security мікросервісу be-wizard (рисунки 3.3 та 3.4). Клас ApplicationConfig відповідає за конфігурацію різних компонентів безпеки в мікросервісі be-wizard. Його основною задачею є налаштування різних бінів, які використовуються для забезпечення автентифікації та авторизації користувачів. У даному класі визначені методи для створення бінів, які надають доступ до репозиторію користувачів, обробника автентифікації, менеджера автентифікації та кодувальника паролів.

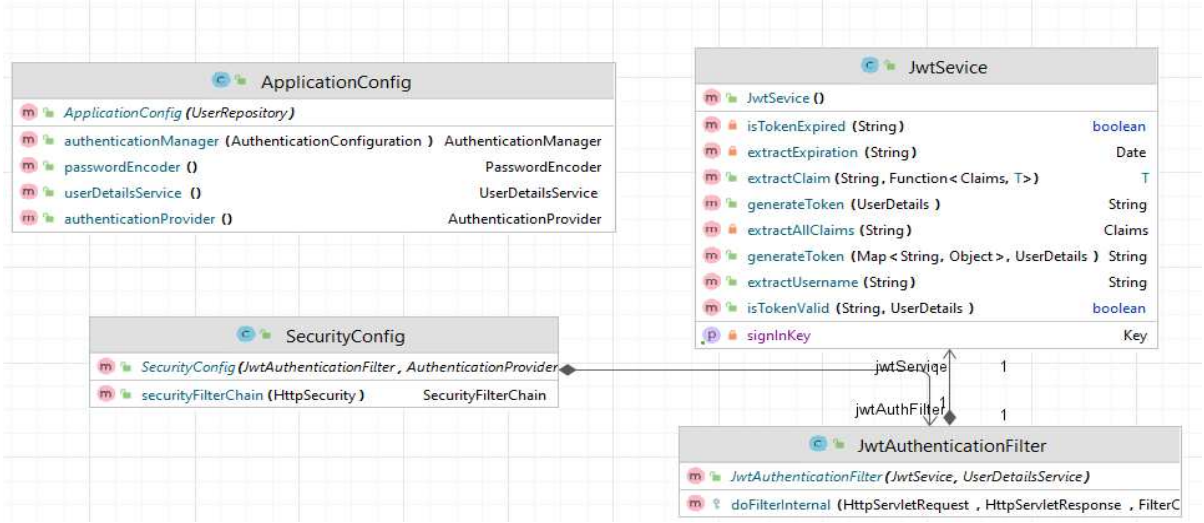


Рисунок 3.3 – Класи компоненту security мікросервісу be-wizard (частина 1)

JwtAuthenticationFilter є компонентом, що відповідає за оброблення запитів на автентифікацію з використанням JWT токенів. Він реалізований як фільтр, який перехоплює всі HTTP-запити та перевіряє наявність JWT токену у заголовку. У разі наявності токену, фільтр перевіряє його валідність та автоматично автентифікує користувача.

SecurityConfig представляє собою клас конфігурації безпеки, де визначаються правила доступу до захищених ресурсів, встановлюються параметри безпеки для HTTP-запитів, налаштовуються фільтри безпеки та визначається спосіб проведення автентифікації та авторизації.

JwtService відповідає за створення та перевірку JWT токенів. У ньому реалізовані методи для генерації та перевірки токенів, а також витягування інформації з токенів. Також він використовується для налаштування секретного ключа для підпису токенів.

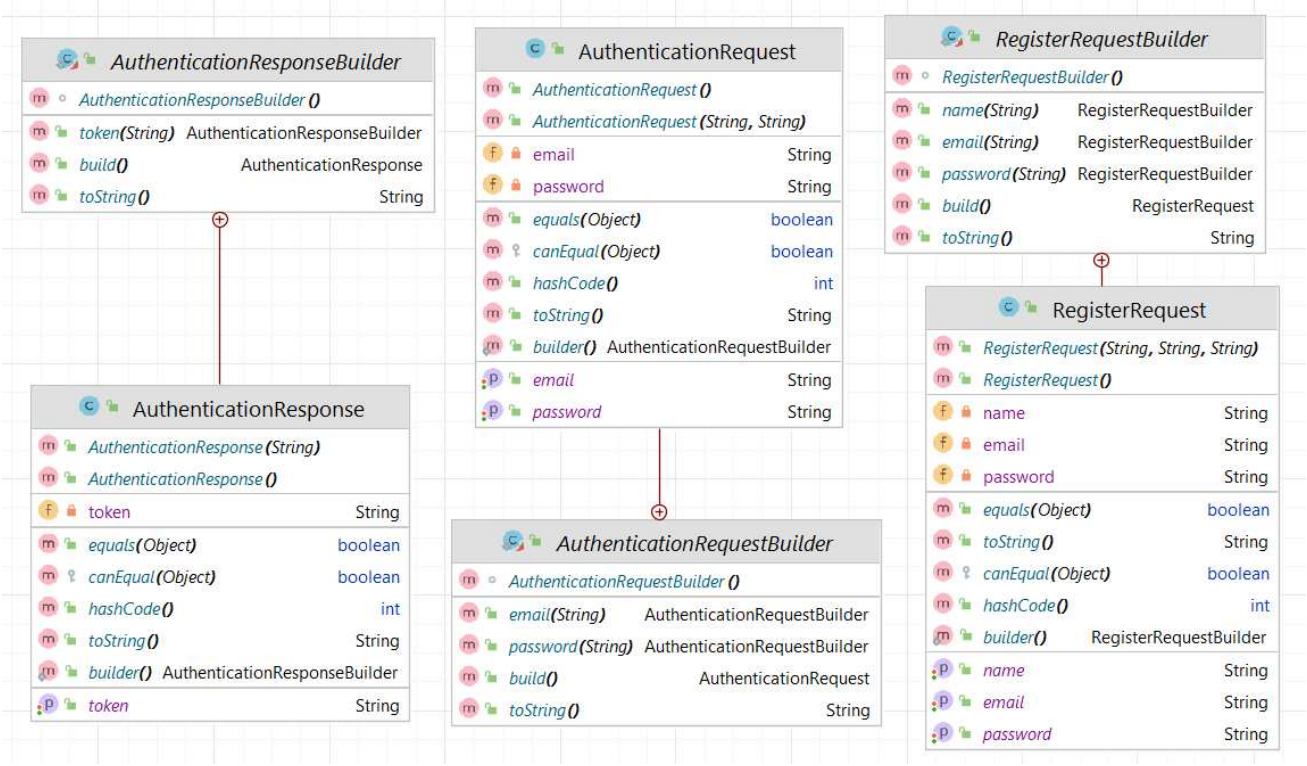


Рисунок 3.4 – Класи компоненту security мікросервісу be-wizard (частина 2)

Класи AuthenticationRequest, AuthenticationResponse та RegisterRequest представляють собою моделі даних, що використовуються для передачі інформації при автентифікації та реєстрації користувачів. Вони містять необхідні поля, такі як ім’я, електронна пошта та пароль, які використовуються для ідентифікації користувача.

FeignService (рисунок 3.5) є інтерфейсом, який використовується для взаємодії з мікросервісом be-planner за допомогою Feign. Він містить два методи: `improveDay`, який відправляє запит на покращення розпорядку дня з вказаними активностями, та `calculateDay`, який відправляє запит на обчислення розпорядку дня з вказаними активностями.

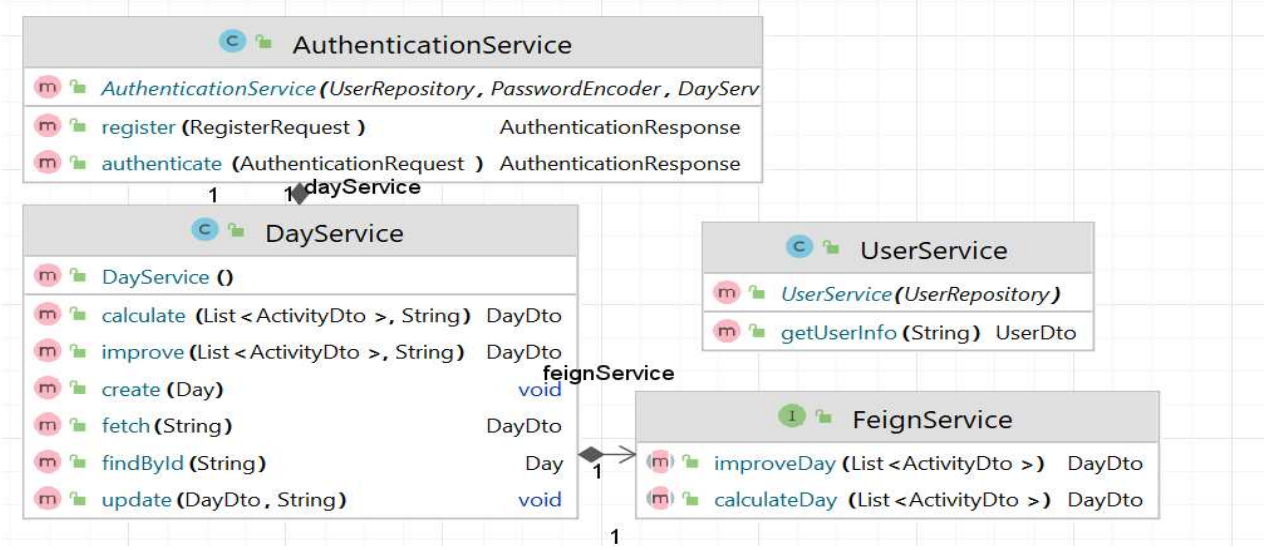


Рисунок 3.5 – Класи компонентів service та feign мікросервісу be-wizard

Клас UserService є сервісом, який надає інформацію про користувачів. Він містить метод для отримання даних користувача за його електронною поштою. AuthenticationService є сервісом, який відповідає за реєстрацію та автентифікацію користувачів. Він містить методи `register`, який реєструє нового користувача та генерує JWT токен для нього, та `authenticate`, який автентифікує користувача на основі наданих даних.

Клас DayService відповідає за оброблення даних про розпорядок дня користувача. Він містить методи для отримання, покращення та обчислення розпорядку дня, а також для оновлення цих даних у БД.

Ці компоненти разом забезпечують взаємодію та оброблення даних у мікросервісі be-wizard, зокрема здійснюють автентифікацію користувачів, керування розпорядком дня та доступ до даних користувачів.

У додатку Б зображена діаграма класів мікросервісу be-planner, де можна побачити всі класи, їх поля та методи, а також взаємозв’язок між ними. Далі буде детальніше розглянуто певні класи цього мікросервісу та їх методи. Загалом всі компоненти які містяться у be-planner виконують аналогічні функції як у be-wizard, проте варто виділити компонент util (рисунок 3.6).

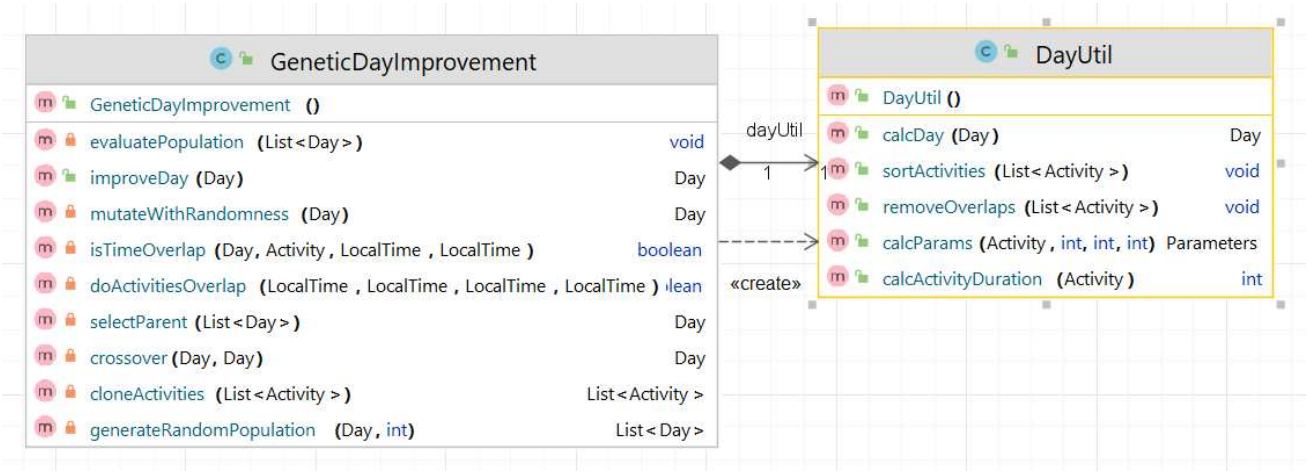


Рисунок 3.6 – Класи компоненту util мікросервісу be-planner

Клас DayUtil (рисунок 3.6) містить методи для обчислення та оптимізації параметрів розпорядку дня користувача. Метод `calcDay` визначає ефективність та енергію розпорядку дня на основі заданих активностей, а також проводить їхнє сортування та видалення наложених один на одного проміжків часу певних активностей. Метод `calcActivityDuration` обчислює тривалість активності. Методи `sortActivities` та `removeOverlaps` використовуються для сортування та видалення наложень активностей відповідно.

Клас GeneticDayImprovement (рисунок 3.6) реалізує генетичний алгоритм для покращення розпорядку дня. Метод `improveDay` приймає початковий розпорядок дня та виконує послідовність операцій для покращення його ефективності та енергії. Це включає створення випадкової популяції, оцінку та сортування популяції, а також застосування операторів кросоверу та мутації для створення нової популяції. Методи `evaluatePopulation`, `selectParent`, `crossover` та `mutateWithRandomness` відповідають за оцінку популяції, вибір батьків для розмноження, кросовер та мутацію відповідно.

Ці компоненти спільно використовуються для механізму покращення планування розпорядку дня користувача в мікросервісі be-planner. Реалізація генетичного алгоритму дає змогу автоматично покращувати розпорядок дня, забезпечуючи оптимальне розподілення часу між різними видами активності.

3.4 Тестування створеного вебсервісу на основі розробленого методу покращення розпорядку дня

У цьому пункті розглянуто процес тестування вебсервісу, щоб забезпечити його якість, надійність та безпеку. Тестування є невід'ємною частиною розроблення програмного забезпечення, оскільки воно допомагає виявити помилки та недоліки на ранніх етапах розроблення. Було проаналізовано роботу певних компонентів, а також результати проведених тестів. Це дасть змогу отримати повне уявлення про рівень підготовки вебсервісу до експлуатації та впевненість у його стабільній роботі.

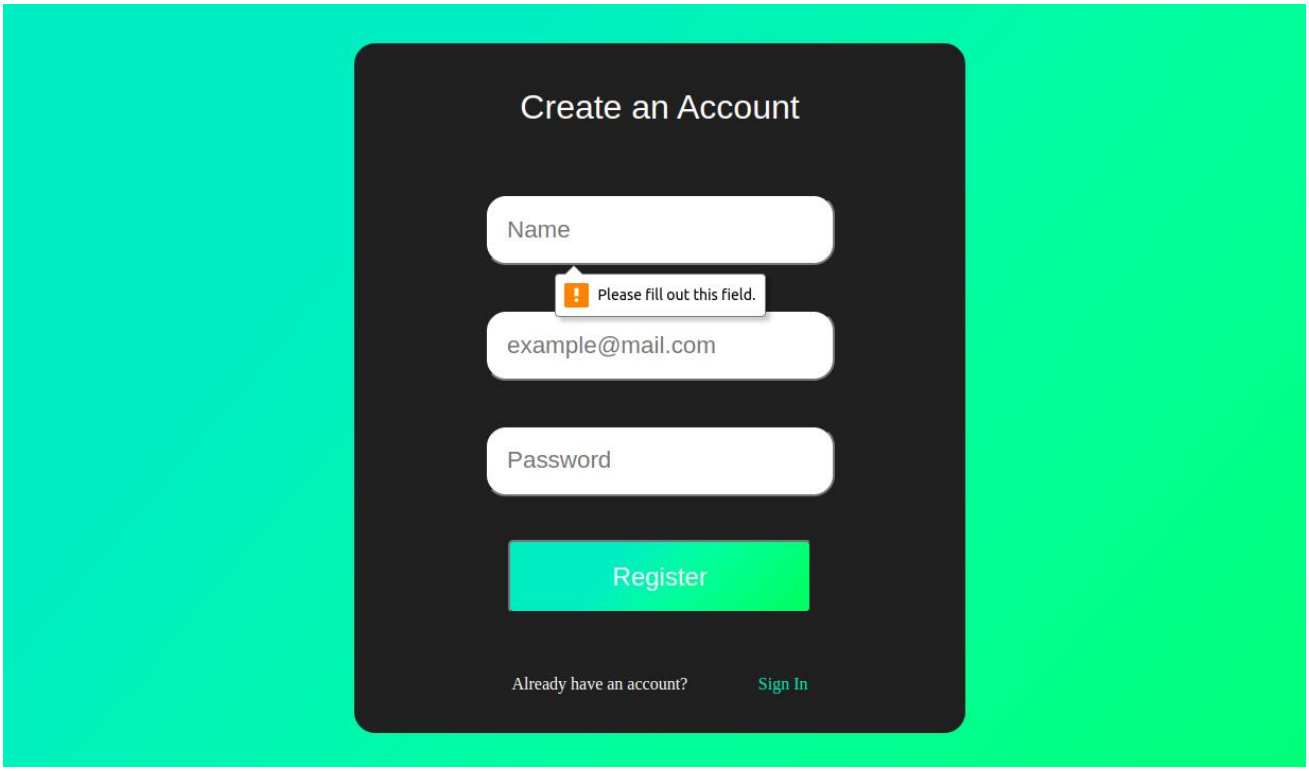


Рисунок 3.7 – Тестування форми реєстрації (пусті поля)

Тестування форми реєстрації на перевірку пустих полів (рисунок 3.7):

– відкрили сторінку реєстрації;
– залишили всі поля форми (Name, Email, Password) пустими;
– натиснули на кнопку “Register”;

– перевірили, що система відобразила повідомлення про помилку, яке вказує на необхідність заповнення всіх полів.

В результаті система успішно виявила пусті поля і відобразила відповідне повідомлення про помилку, що вказує на обов’язковість заповнення всіх полів форми для реєстрації.

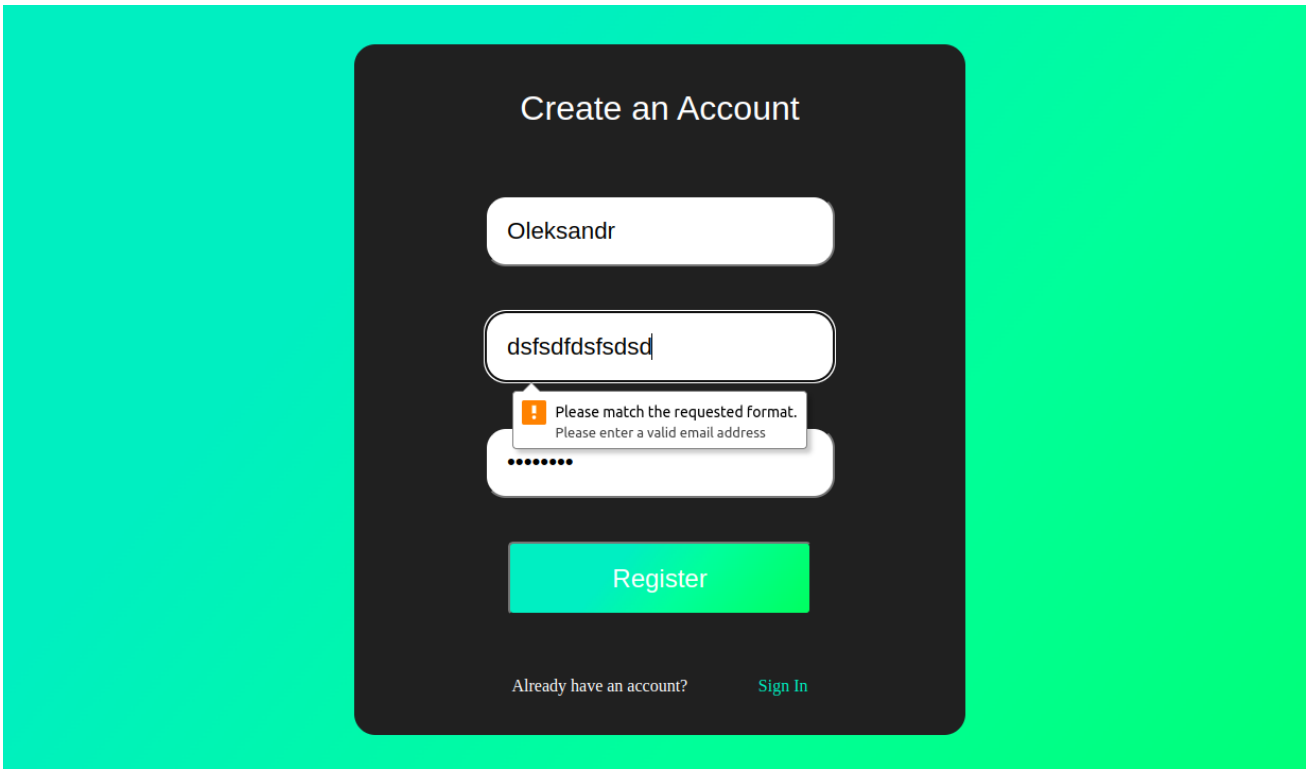


Рисунок 3.8 – Тестування форми реєстрації (некорректний формат електронної скриньки)

Тестування форми реєстрації на виявлення некоректного формату електронної скриньки (рисунок 3.8):

– відкрили сторінку реєстрації;

– заповнили поле “Name” валідним іменем;

– заповнили поле “Email” некоректною електронною скринькою;

– заповнили поле “Password” валідним паролем;

– натиснули на кнопку “Register”;

– перевірили, що система відобразила повідомлення про помилку, яке вказує на некоректний формат введеної електронної скриньки.

В результаті система успішно виявила некоректний формат електронної скриньки та відобразила відповідне повідомлення про помилку, що вказує на необхідність введення валідної адреси електронної скриньки.

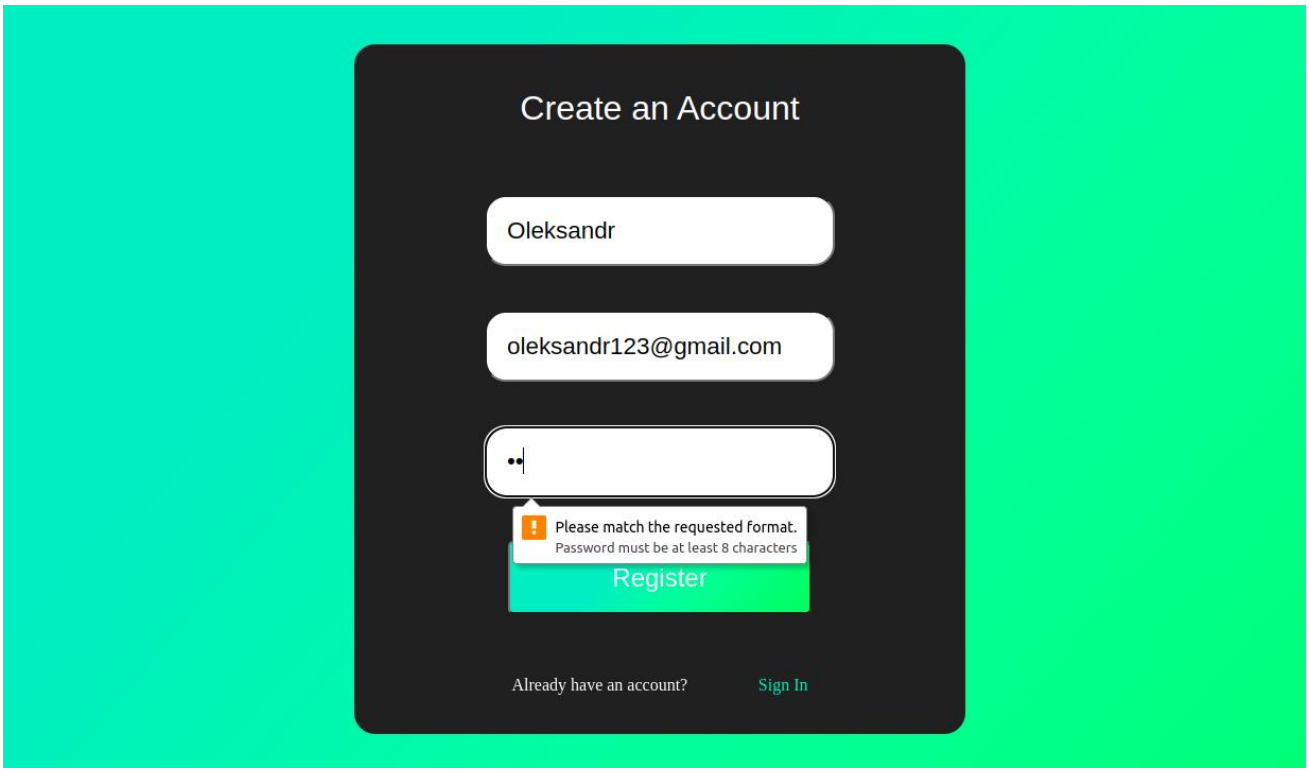


Рисунок 3.9 – Тестування форми реєстрації (неправильний формат паролю)

Тестування форми реєстрації на виявлення некоректного формату паролю (рисунок 3.9):

– відкрили сторінку реєстрації;

– заповнили поле “Name” валідним іменем;

– заповнили поле “Email” валідною адресою електронної пошти;

– заповнили поле “Password” некоректним паролем, що не відповідає вимогам;

– натиснули на кнопку “Register”;

– перевірили, що система відобразила повідомлення про помилку, яке вказує на некоректний формат введеного паролю.

В результаті система успішно виявила некоректний формат паролю і відобразила відповідне повідомлення про помилку, що вказує на необхідність

введення паролю, який відповідає встановленим вимогам (наприклад, мінімальна довжина, наявність цифр і букв).

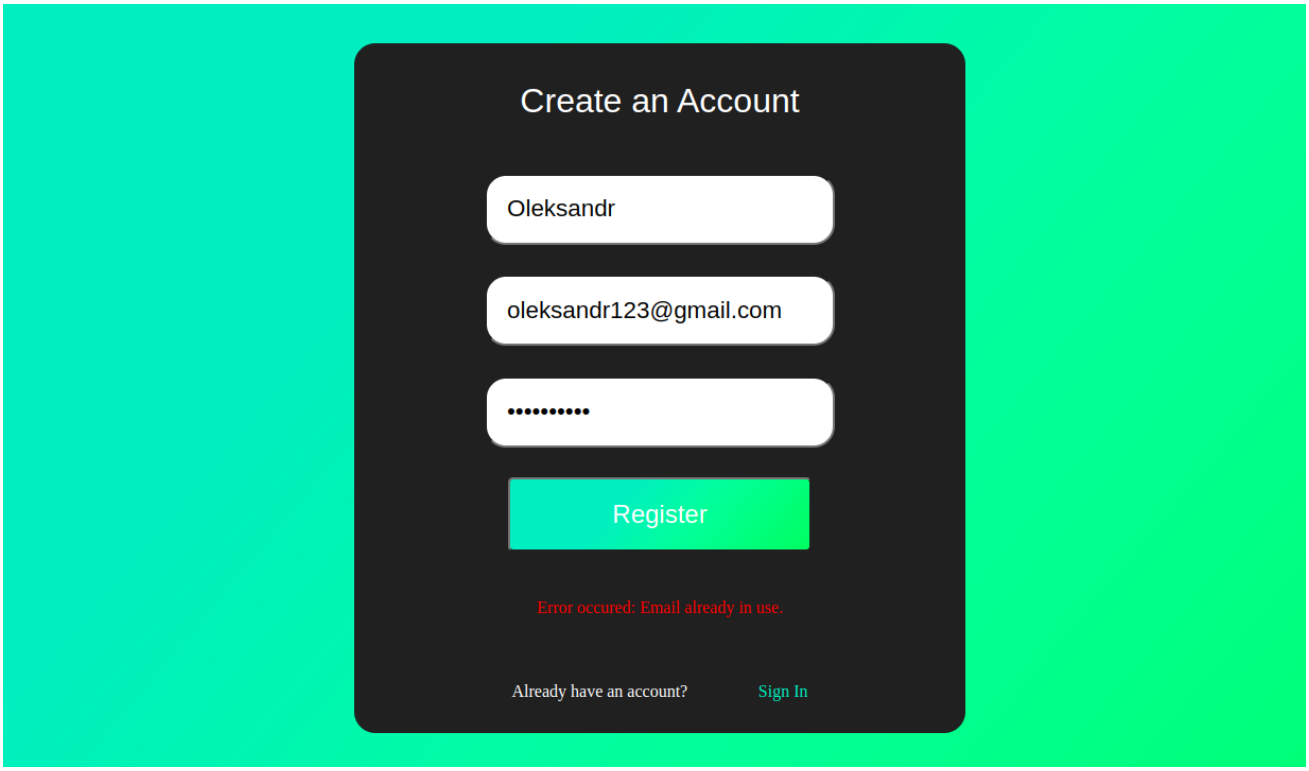


Рисунок 3.10 – Тестування форми реєстрації (помилка – електронна скринька вже використовується)

Тестування форми реєстрації на видання помилки, коли введена користувачем електронна скринька вже використовується (рисунок 3.10):

– відкрили сторінку реєстрації;

– заповнили поле “Name” валідним іменем;

– заповнили поле “Email” адресою електронної скриньки, яка вже зареєстрована в системі;

– заповнили поле “Password” валідним паролем;

– натиснули на кнопку “Register”;

– перевірили, що система відобразила повідомлення про помилку, яке вказує, що електронна скринька вже використовується.

В результаті система успішно виявила, що введена адреса електронної пошти вже зареєстрована в системі, і відобразила відповідне повідомлення про

помилку, що вказує на необхідність використання іншої адреси електронної пошти для реєстрації.

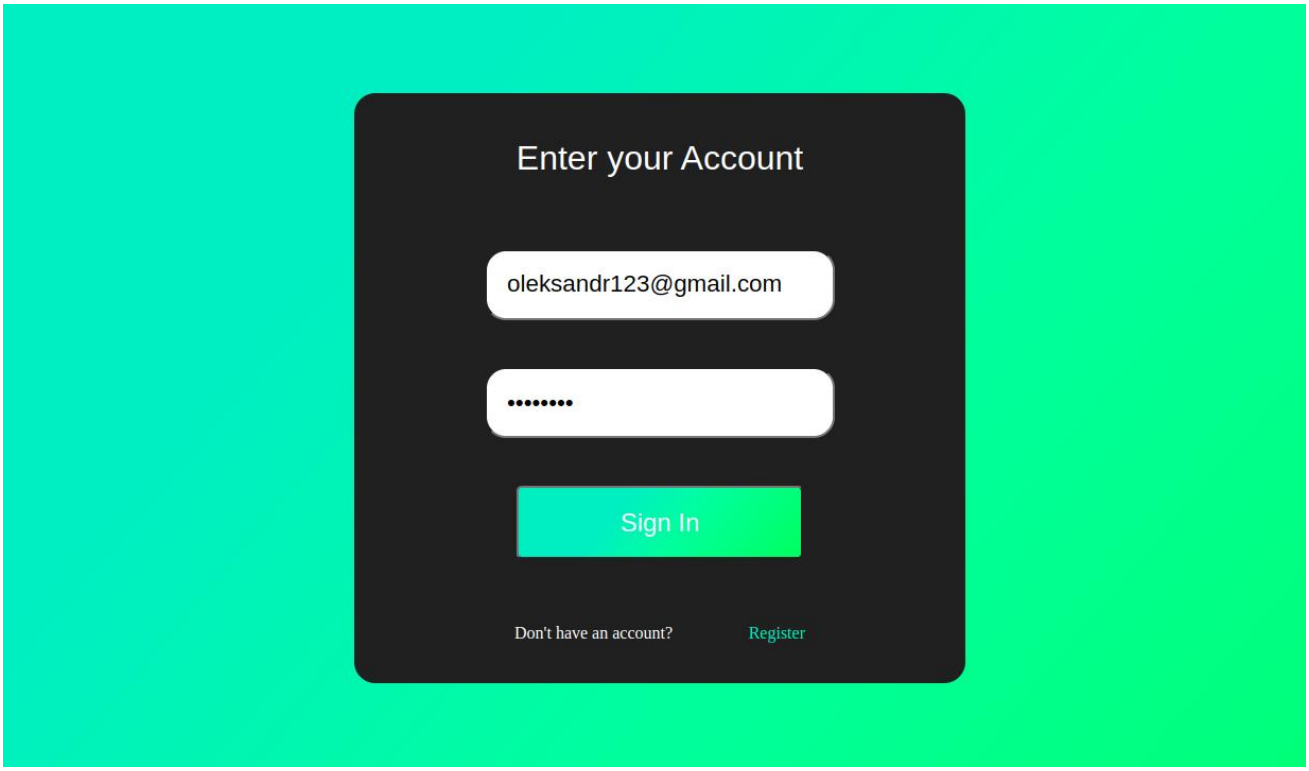


Рисунок 3.11 – Тестування форми входу в обліковий запис (дані зареєстрованого користувача)

Тестування форми входу на вхід в наявний обліковий запис за даними зареєстрованого користувача (рисунок 3.11):

– відкрили сторінку входу в обліковий запис;

– заповнили поле “Email” адресою електронної пошти, яка вже зареєстрована в системі (наприклад, “registered_user@example.com”);

– заповнили поле “Password” валідним паролем, який відповідає обліковому запису;

– натиснули на кнопку “Sign In”;

– перевірили, що система успішно автентифікувала користувача і перенаправила його на сторінку з розпорядком дня.

В результаті система успішно здійснила автентифікацію користувача з використанням введених даних і перенаправила його на сторінку з розпорядком дня, підтверджуючи правильність функціонування форми входу.

Далі важливо пояснити, що саме мається на увазі під поняттями "енергія" та "ефективність", оскільки ці параметри є ключовими показниками успішності роботи методу покращення планування розпорядку дня.

Енергія відображає фізичний та психічний стан користувача протягом дня. Це показник, який враховує кілька аспектів: фізичний та психічний стан користувача. «» Фізичний стан включає рівень втоми, кількість та якість сну, фізичну активність. Відповідно, хороший фізичний стан означає високу енергію. Психічний стан враховує рівень стресу, наявність перерв на відпочинок, заняття, які приносять задоволення та знижують психоемоційне навантаження. При розрахунку енергії враховується кількість годин, відведених на сон, кількість перерв у роботі, час, присвячений фізичній активності та відпочинку. Різні види активностей мають різний вплив на енергію. Наприклад, недостатній сон або його відсутність значно знижує цей показник, тоді як регулярна фізична активність, перерви та сон сприяють його підвищенню.

Ефективність визначає продуктивність користувача, тобто його здатність ефективно виконувати завдання протягом дня. Цей показник залежить від таких факторів, рівень енергії користувача під час роботи та розподіл часу. Розподіл часу передбачає оптимальне планування часу для різних видів діяльності, включаючи роботу, навчання, відпочинок та інші активності. Для розрахунку ефективності аналізується, як час розподіляється між різними активностями. Наприклад, якщо користувач присвячує занадто багато часу роботі без перерв, його ефективність знижується. З іншого боку, ефективний розподіл часу на важливі завдання, регулярні перерви для відновлення концентрації та запобігання втомі сприяють підвищенню ефективності.

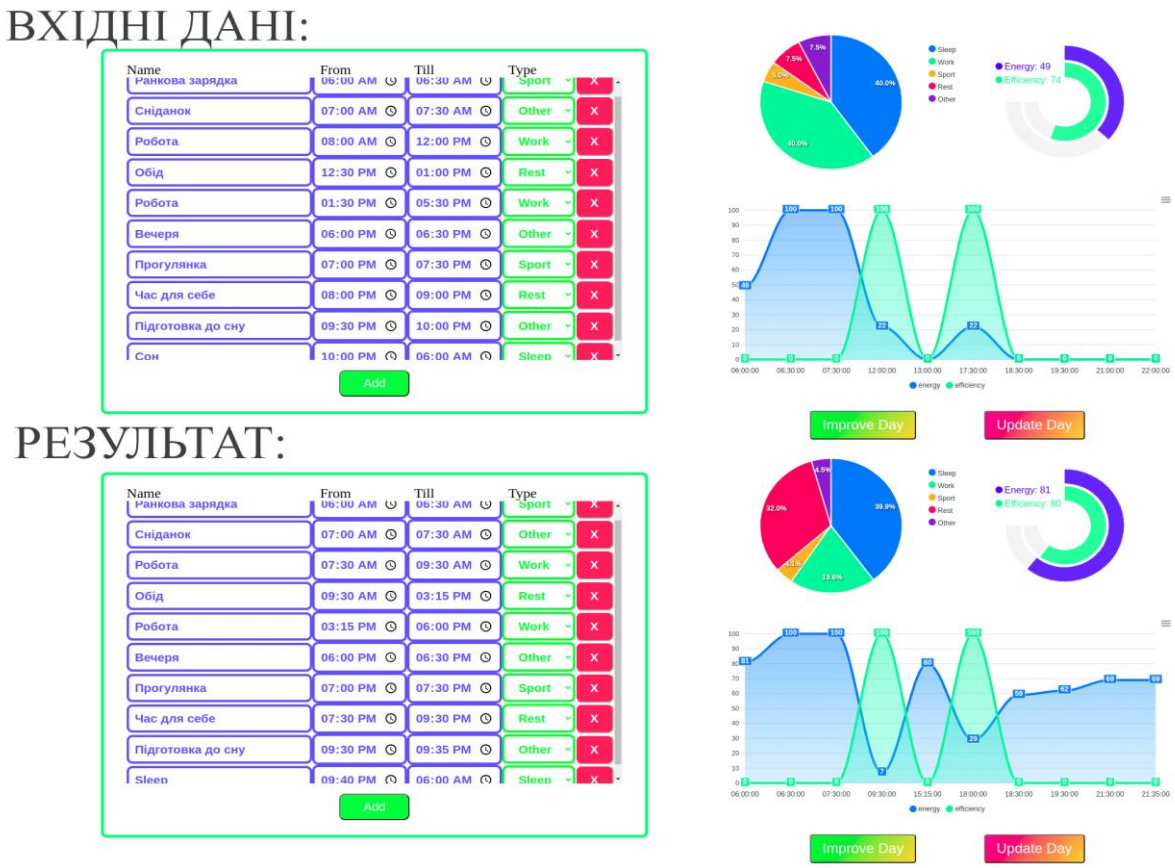


Рисунок 3.12 – Тестування методу покращення планування розпорядку дня. Пресет 1.

Тестування методу, використовуючи вхідні дані зображені на рисунку 3.12.

Початкові параметри є такими:

– загальна енергія – 49;

– ефективність – 74.

Після оптимізації генетичним алгоритмом отримали такі значення:

– зменшення годин роботи та збільшення часу відпочинку;

– загальна енергія підвищилася до 81;

– ефективність зросла до 80.

В результаті отримані показники енергії та ефективності на високому рівні, що вказує на успішне покращення розпорядку дня.

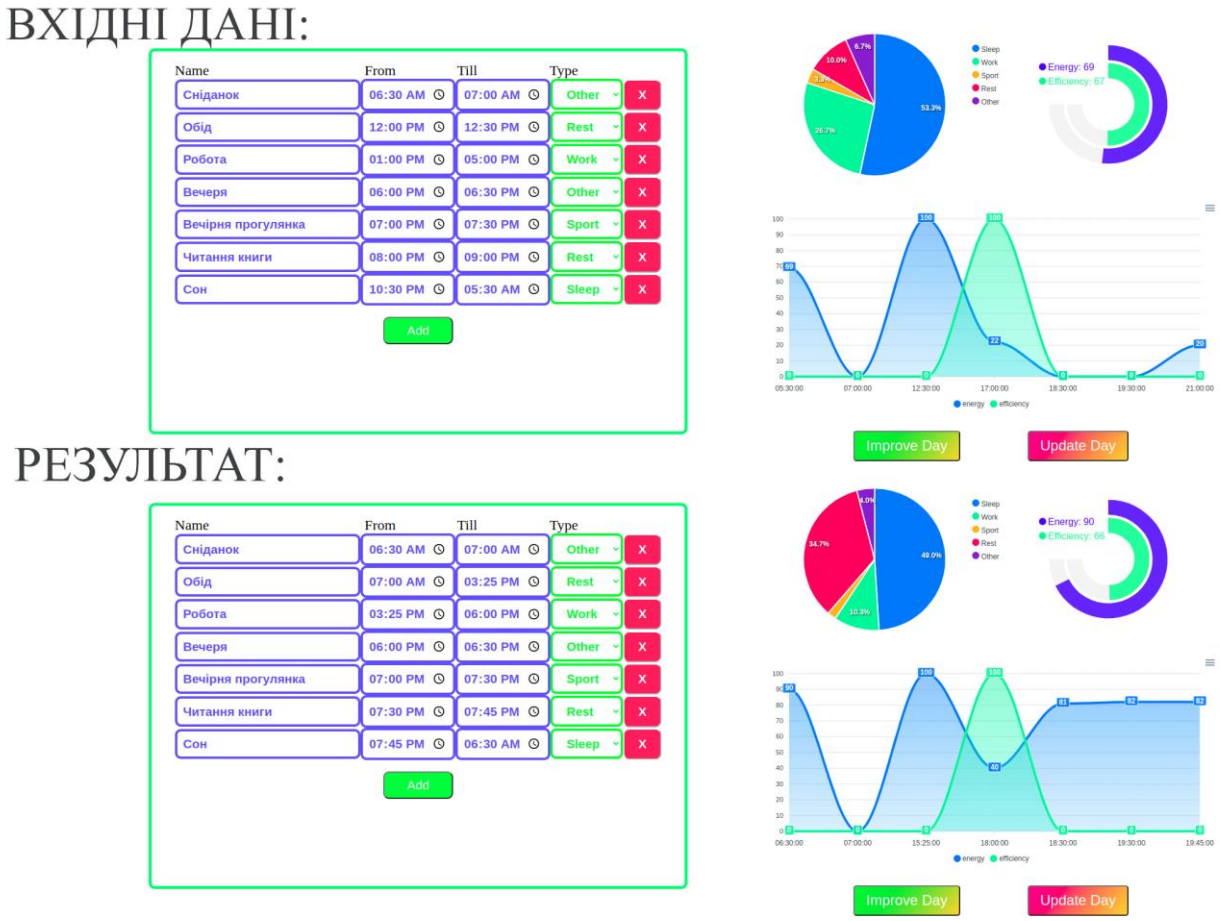


Рисунок 3.13 – Тестування методу покращення планування розпорядку дня. Пресет 2.

Тестування методу, використовуючи вхідні дані зображені на рисунку 3.13.

Початкові параметри встановили такими:

– загальна енергія – 69;

– ефективність – 67.

Після оптимізації генетичним алгоритмом отримано такі значення:

– зменшення годин роботи та збільшення часу відпочинку;

– загальна енергія підвищилася до 90;

– ефективність трішки спала до 66.

В результаті отримані показники енергії та ефективності на досить високому рівні, що вказує на успішне покращення розпорядку дня.

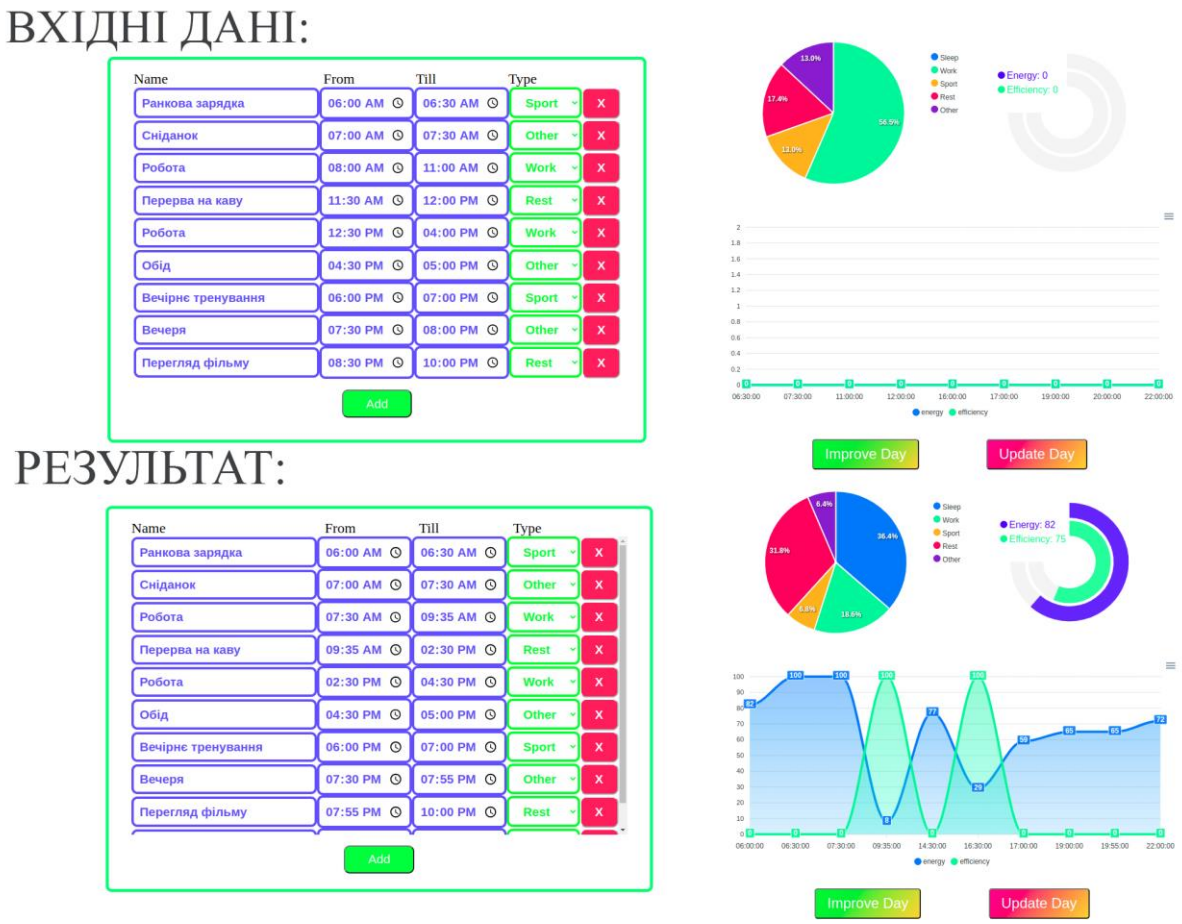


Рисунок 3.14 – Тестування методу покращення планування розпорядку дня. Пресет 3.

Тестування методу, використовуючи вхідні дані зображені на рисунку 3.14.

Початкові параметри поставили такими:

– відсутність сну у користувача;

– загальна енергія – 0;

– ефективність – 0.

Після оптимізації генетичним алгоритмом значення стали такими:

– додавання активності сну для користувача;

– зменшення годин роботи та збільшення часу відпочинку;

– загальна енергія підвищилася до 82;

– ефективність зросла до 75.

В результаті отримані показники енергії та ефективності на високому рівні, що вказує на успішне покращення розпорядку дня.

Рисунок 3.15 – Тестування методу покращення планування розпорядку дня. Пресет 4.

Тестування методу, використовуючи вхідні дані зображені на рисунку 3.15.

Початкові параметри:

– чотири активності роботи;

– загальна енергія – 56;

– ефективність – 77.

Після оптимізації генетичним алгоритмом:

– видалення однієї активності роботи;

– зменшення годин роботи та збільшення часу відпочинку;

– загальна енергія підвищилася до 81;

– ефективність зросла до 80.

В результаті отримані показники енергії та ефективності на високому рівні, що вказує на успішне покращення розпорядку дня.

Таблиця 3.1 – Зведена таблиця з результатами тестування різних пресетів на покращення з допомогою розробленого методу.

№ Пресету	**Параметри енергії та ефективності ДО покращення**	**Параметри енергії та ефективності ПІСЛЯ покращення**
1	49,74	81,80
2	69,67	90,66
3	0	82,75
4	56,77	81,80

Таблиця 3.1 демонструє суттєве підвищення показників ефективності та енергії користувачів після застосування методу.

Для першого пресету початкові показники енергії складали 49, а ефективності – 74. Після застосування методу ці показники збільшилися до 81 і 80 відповідно. Це означає, що метод зміг краще розподілити ресурси користувача протягом дня, зменшивши втрати енергії та підвищивши продуктивність. Генетичний алгоритм оптимізував розпорядок дня, забезпечивши більш ефективне використання часу, що дозволило значно підвищити загальні показники енергії та ефективності. Такий результат підтверджує, що застосування методу покращення планування розпорядку дня може суттєво вплинути на підвищення якості життя користувачів.

Для другого пресету початкові показники енергії та ефективності становили 69 та 67 відповідно. Після оптимізації енергія зросла до 90, тоді як ефективність залишилася майже незмінною – 66. Це свідчить про те, що метод зміг суттєво збільшити енергетичний резерв користувача, хоча ефективність виконання завдань залишилася на тому ж рівні. Можливо, це пов’язано з тим, що початковий розпорядок дня вже був досить добре організований, і основна оптимізація була спрямована на підвищення енергетичних ресурсів. Це

підтверджує здатність методу не тільки підвищувати продуктивність, але й ефективно розподіляти енергією користувачів.

Третій пресет показав найяскравіші результати, оскільки початкові показники енергії та ефективності складали 0, що означало повну відсутність організованого планування. Після застосування методу показники зросли до 82 для енергії та 75 для ефективності. Це свідчить про те, що алгоритм успішно покращив розпорядок дня, забезпечивши високий рівень енергії та значну ефективність. Такий результат демонструє, що навіть у випадках повної відсутності організованого розпорядку, метод може забезпечити кардинальні зміни, значно підвищивши продуктивність та енергію користувача.

Для четвертого пресету початкові показники енергії становили 56, а ефективності – 77. Після оптимізації ці показники збільшилися до 81 і 80 відповідно. Це свідчить про те, що метод зміг значно підвищити енергетичні ресурси користувача, а також трохи покращити ефективність виконання завдань. Як і в попередніх випадках, алгоритм забезпечив збалансоване використання часу, що сприяло підвищенню загальних показників. Цей результат підтверджує, що розроблений метод є універсальним і може ефективно покращувати управління часом для різних категорій користувачів.

Таблиця 3.1 демонструю, як розроблений метод, балансуючи часом між різними типами активностей, значно підвищує загальні показники енергії та ефективності. У всіх тестових наборах спостерігалася аналогічна тенденція: значне підвищення обох параметрів, що свідчить про високу ефективність розробленого методу. Ці результати підтверджують здатність методу оптимізувати розпорядок дня користувачів, покращуючи їхній енергетичний баланс і загальну продуктивність. Таблиця слугує важливим доказом ефективності розробленого методу і його практичної цінності для покращення управління часом. Загалом в цьому пункті було протестовано різні аспекти вебсервісу, включаючи форми реєстрації та входу в обліковий запис, а також функціонал оптимізації розпорядку дня за допомогою генетичного алгоритму.

Тестування виявило, що вебсервіс працює правильно та надійно, надаючи користувачам зручність у використанні та оптимальний розпорядок дня.

3.5 Інструкція користувача щодо використання вебсервісу

Цей підрозділ містить інструкції щодо роботи зі створеним вебсервісом, щоб забезпечити продуктивне використання його користувачем. Кожен наданий нижче скріншот показує різні сторінки вебсервісу та надає докладні вказівки щодо призначення їх компонентів та їх використання. Дотримуючись цих інструкцій, користувач зможе використовувати систему на максимум для досягнення своїх цілей та завдань. Також важливо відмітити що мінімальними вимогами для використання системи є наявність у користувача ПК (можна з телефону, але досвід використання на ПК буде максимально комфортним), браузера та підключення до мережі інтернет.

При відкритті вебсервісу, користувача зустрічає головна сторінка (рисунок 3.16).

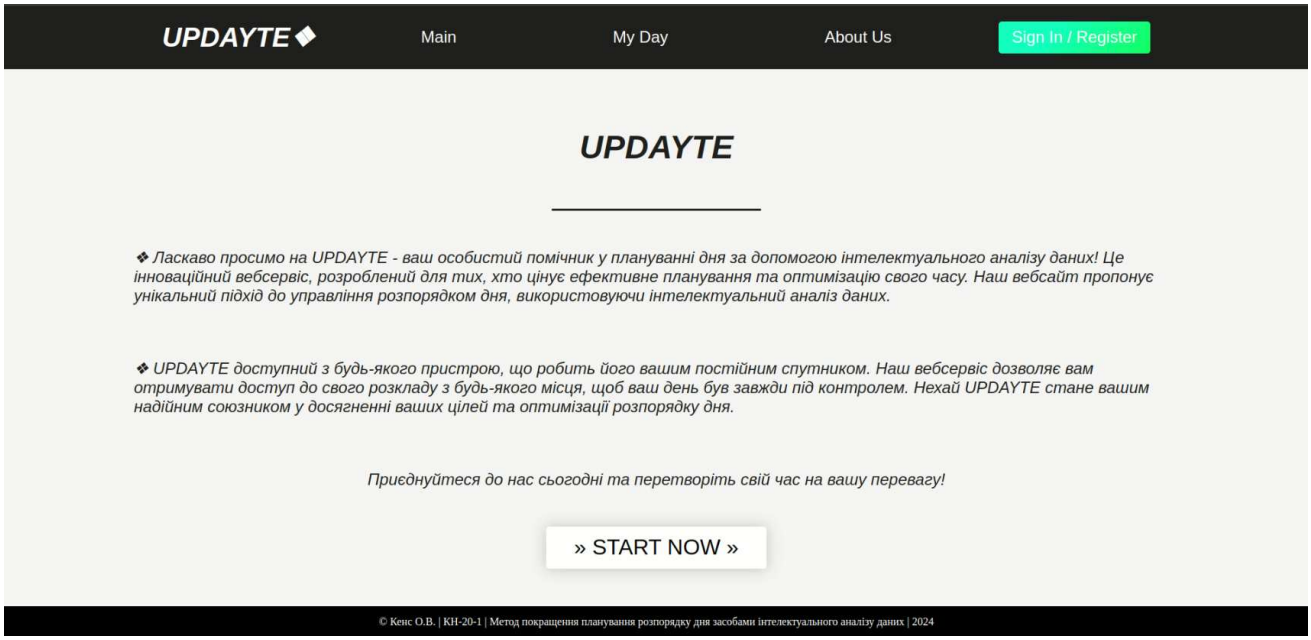


Рисунок 3.16 – Головна сторінка вебсервісу

Головна сторінка вебсервісу містить навігаційне меню з такими кнопками як “Main”, “My Day”, “About Us” та “Sign In/Register”. Крім того, по центру сторінки є кнопка “Start Now”. При натисканні на “Main” користувач буде перенаправлений на головну сторінку. “My Day” відкриє сторінку з розпорядком дня користувача, “About Us” – інформацію про розробника, а “Sign In/Register” – сторінку для входу або реєстрації. Кнопка “Start Now” дасть змогу неавторизованому користувачу зареєструватися, а авторизованому – перейти до його розпорядку дня.

Наступний крок який варто зробити новому користувачу – це реєстрація. На сторінці реєстрації розташована форма з полями Name, Email, Password та кнопкою Register (рисунок 3.17).

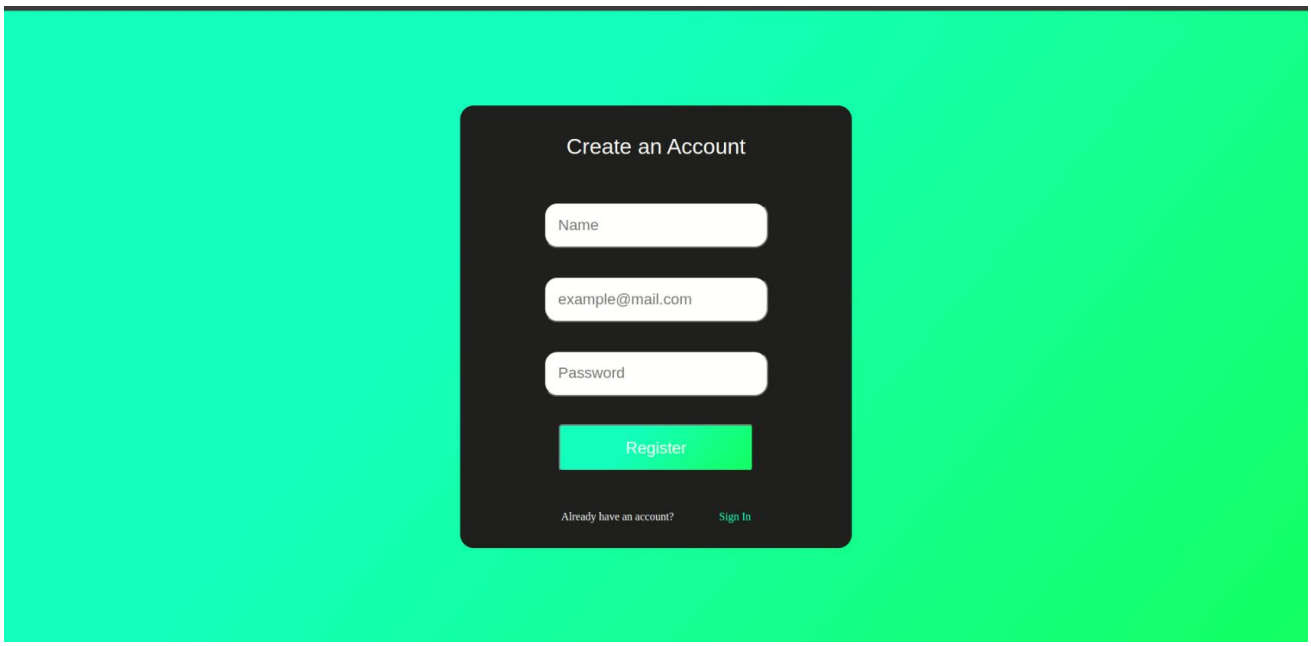


Рисунок 3.17 – Сторінка для реєстрації користувача вебсервісу

Після заповнення цих полів користувач може натиснути кнопку Register для реєстрації. У випадку введення неправильних даних або вже наявного електронного адресу система надсилатиме відповідні повідомлення про помилку. Нижче форми є посилання для переходу на сторінку входу в обліковий запис для вже зареєстрованих користувачів. Всі введені дані підуть на оброблення на бекенд і якщо транзакція пройшла успішно, з бекенду

повернеться ключ доступу (access token) який дасть змогу користувачеві повноцінно користуватися системою.

Якщо ж користувач має обліковий запис, він може ввійти в нього використовуючи сторінку для входу (рисунок 3.18).

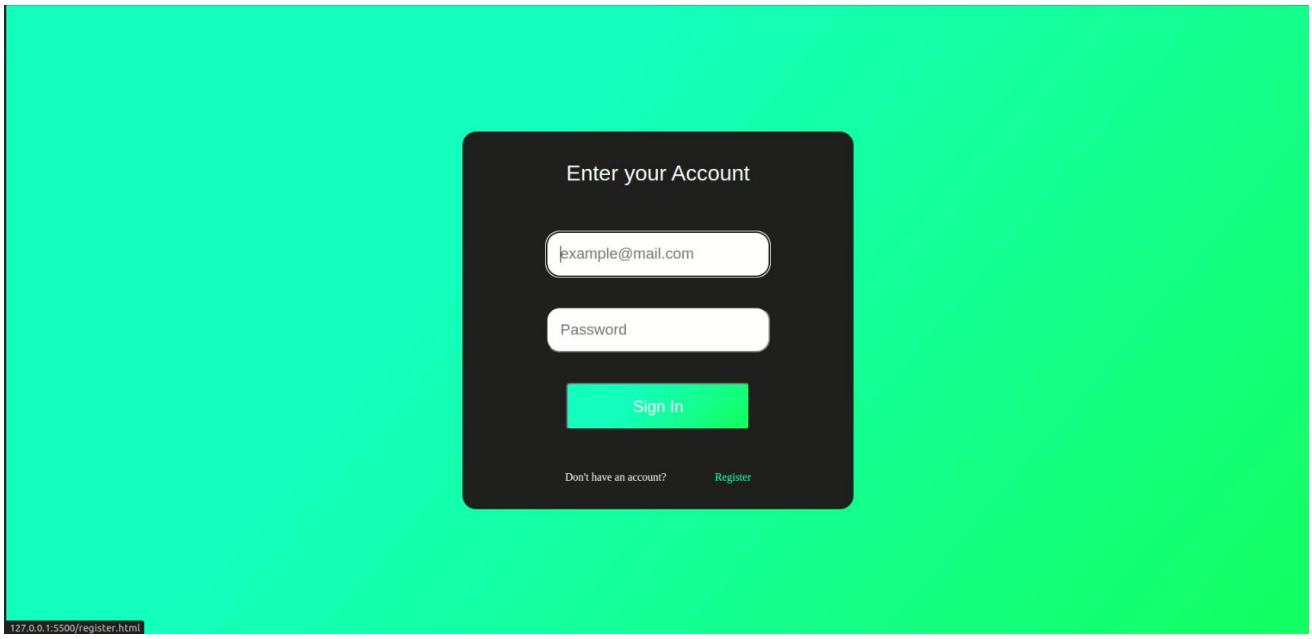


Рисунок 3.18 – Сторінка входу в наявний обліковий запис користувача вебсервісу

На сторінці входу до наявного облікового запису розташована форма з полями Email, Password та кнопкою Sign In. Після введення даних користувач може натиснути кнопку Sign In для входу. У разі неправильного введення даних система повідомлятиме про помилку. Нижче форми є посилання для переходу на сторінку реєстрації для нових користувачів. Всі введені дані, аналогічно до реєстрації, підуть на оброблення на бекенд для авторизації користувача.

Після успішної реєстрації користувача зустрічає сторінка з його розпорядком дня (рисунок 3.19), якщо ж це новий користувач, його зустріне автоматично створений пресет розпорядку дня, який він може змінити за його бажанням, або взагалі видалити.

Рисунок 3.19 – Сторінка зs списком активностей користувача, графіком та діаграмами

На сторінці з розпорядком дня користувач може переглядати свій розпорядок дня, графік енергії та ефективності в конкретні часові проміжки дня, а також діаграму загальної енергії та ефективності, разом із діаграмою відсоткового співвідношення всіх типів активностей за день. Користувач має можливість видаляти свої активності, натискаючи на червоний хрестик біля відповідної активності. Також він може додавати нові активності, натиснувши на кнопку “Add” внизу списку активностей, що призведе до появи рядка внизу списку з полями Name, From, Till, Type, які користувач повинен заповнити.

Щоб зберегти список активностей та оновити дані на графіку та діаграмах відповідно до оновленого розпорядку дня, користувач може натиснути на кнопку “Update Day”. У випадку, якщо користувача не задовільнять параметри на графіках та діаграмах, він може натиснути на кнопку “Improve Day” для максимального покращення розпорядку дня. Це викличе виконання генетичного алгоритму на бекенді. Користувачеві залишиться лише зачекати декілька секунд, доки генетичний алгоритм не завершить роботу та не поверне результат.

Якщо користувачеві стане цікаво дізнатись про розробника сайту, він може перейти на відповідну сторінку, натиснувши на кнопку “About Us” з

навігаційного меню головної сторінки вебсервісу. На цій сторінці доступні посилання на різні платформи, де можна знайти додаткову інформацію про розробника та його роботу (рисунок 3.20).

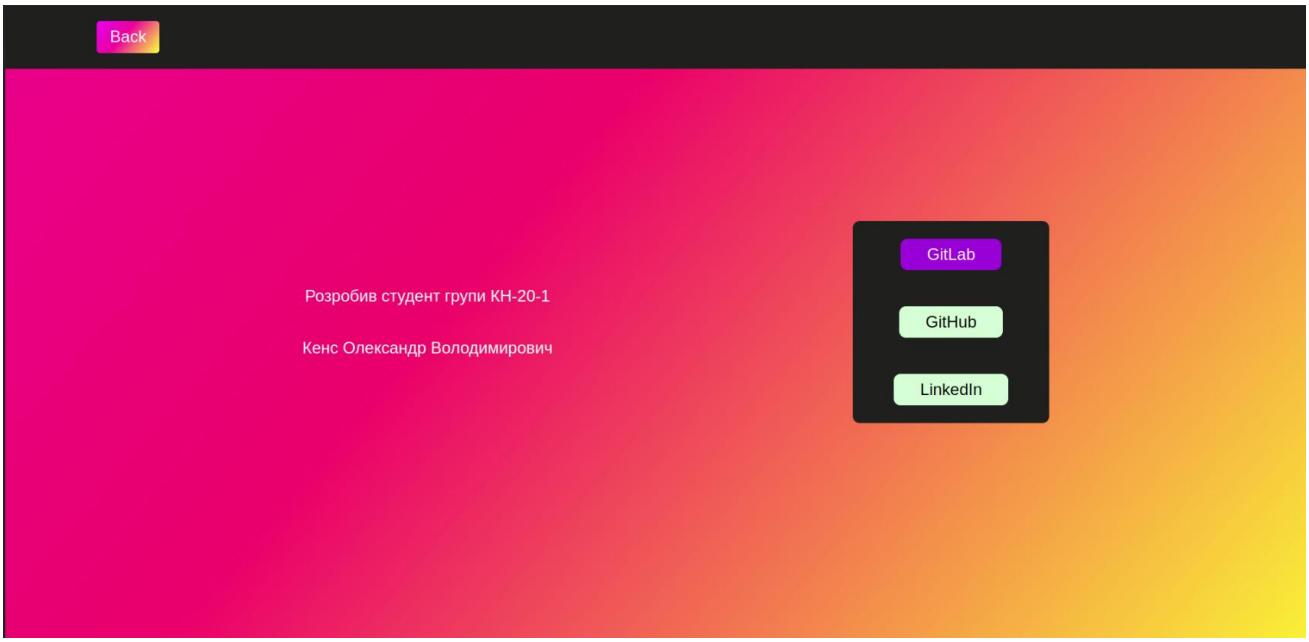


Рисунок 3.20 – Сторінка з посиланнями на розробника

Ці посилання включають GitHub, GitLab та LinkedIn. Користувач може натиснути на будь-яке з посилань, щоб перейти на відповідні профілі розробника на цих платформах.

Отже, в цьому пункті було розглянуто основні сторінки та функціонал вебсервісу, а також надано інформацію щодо його використання. Користувач може легко переміщатися вебсервісом за допомогою навігаційного меню на головній сторінці, використовувати форми для реєстрації та входу, переглядати свій розпорядок дня, керувати активностями.

3.6 Висновки до розділу 3

У розділі 3 описано в деталях різні аспекти розробленого методу покращення розпорядку дня. Проаналізовано шляхи дослідження та вибрано

необхідні засоби для розроблення програмного забезпечення на основі методу, враховуючи потреби користувачів та технічні можливості.

Структура та функціональне призначення програмних складових системи були ретельно розглянуті, з урахуванням необхідних функцій для оптимального використання користувачем. Під час тестування було перевірено різні аспекти функціонування системи, включаючи реєстрацію, вхід в обліковий запис, покращення планування розпорядку дня та відповідність вимогам щодо коректності та ефективності роботи. Насамкінець, у розділі подано інструкцію користувача, що призначена для надання інформації потенційним користувачам для зручного та ефективного використання вебсервісу.

Загалом, ретельне планування, розроблення та тестування дали змогу створити функціональний та надійний метод, який забезпечує користувачам необхідні можливості для покращення їхнього розпорядку дня, та готовий до використання в реальних умовах.

Загальні висновки

У результаті виконання кваліфікаційної роботи бакалавра було успішно досягнуто мету роботи, а саме покращено планування розпорядку дня через впровадження методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

У першому розділі проведено детальний аналіз наявних методів та програмних рішень, які застосовуються для вирішення цієї задачі, зокрема, проаналізовано їхні переваги та недоліки. У другому розділі подано опис розробленого методу покращення планування розпорядку дня засобами ІАД. Вхідними даними методу є різні аспекти розпорядку дня користувача, такі як робота, відпочинок, фізична активність та сон. Спроєктовано архітектуру вебсервісу на основі методу, що забезпечує взаємозв’язок між компонентами системи. В архітектурі були враховані питання масштабованості та надійності, щоб забезпечити її стабільну роботу навіть під час високих навантажень.

У третьому розділі описано процес програмної реалізації методу покращення розпорядку дня, включно з вибором засобів розроблення, структурою та функціональним призначенням програмних складових вебсервісу. Вибрано такі технології, як Java SpringBoot для бекенду, JavaScript для фронтенду, а також БД PostgreSQL та MongoDB для зберігання даних.

Проведено успішне тестування розробленого методу покращення розпорядку дня на створеного на його основі вебсервісу. Визначено вимоги до розгортання системи, що включають необхідні програмні та апаратні ресурси, а також надано рекомендації щодо безпеки даних користувачів.

Отже, ретельне проєктування та програмна реалізація методу покращення розпорядку дня дали змогу створити функціональний та надійний інструмент для управління часом. Створений вебсервіс може бути корисним користувачам, щоби підвищити їхню продуктивність, зменшити стрес та досягти кращого балансу між професійним та особистим життям.

Перелік посилань

1. Основні принципи ідеального розпорядку дня – публікації на Jobs.ua. *jobs.ua*. URL: https://jobs.ua/articles/osnovn-printsipi-dealnogo-rozporyadku-dnya-13768

2. Martin F., Stamper B., Flowers C. Examining student perception of readiness for online learning: Importance and confidence. *Online Learning*. 2020. Vol. 24, no. 2. P. 38–58. URL: https://doi.org/10.24059/olj.v24i2.2053

3. Ткачук В. Тайм-менеджмент як засіб підвищення ефективності навчання. *Grail of Science*. 2024. № 35. С. 414–417. URL: https://doi.org/10.36074/grail-of-science.19.01.2024.074

4. Як поліпшити баланс між роботою та особистим життям. Experience Dropbox. *Dropbox.com*. URL: https://experience.dropbox.com/uk-ua/resources/work-life-balance

5. Особливості впровадження інформаційних технологій в освітній процес / В. Осмятченко та ін. *Актуальні питання у сучасній науці*. 2023. № 7(13). С. 60–74. URL: https://doi.org/10.52058/2786-6300-2023-7(13)-60-74

6. Radiuk P., Pavlova O., Hrypynska N. An ensemble machine learning approach for Twitter sentiment analysis. *The 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2022). Volume I: Main Conference* : CEUR-Workshop Proceedings. Vol. 3171. (Gliwice, Poland, 12–13 May 2022). CEUR-WS.org, Aachen, 2022. P. 387–397. URL: https://ceur-ws.org/Vol-3171/paper32.pdf

7. Ismail N. M., Putri Z., Noviyanti A. Pomodoro technique analysis in zoom-based classrooms. *Journal of English Education and Linguistics Studies*. 2022. Vol. 9, no. 1. P. 75–96. URL: https://doi.org/10.30762/jeels.v9i1.4298

8. Rapti K. 12 Time Management Techniques to Mix & Match. *LinkedIn.com*. URL: https://www.linkedin.com/pulse/12-time-management-techniques-mix-match-katerina-rapti/

9. 8 причин, чому тайм-менеджмент не працює. *Anywhere Club*. URL: https://aw.club/global/uk/blog/why-time-management-does-not-work

10. Grabovska S., Musakovska O. Information overload: The psychological area. *Psychological Journal*. 2020. Vol. 6, no. 7. P. 18–29. URL: https://doi.org/10.31108/1.2020.6.7.2

11. Intelligent data analysis using artificial neural networks for decision making in the education domain / P. Radiuk et al. *Herald of Khmelnytskyi National University. Technical sciences*. 2021. Vol. 303, no. 6. P. 111–114. URL: https://doi.org/10.31891/2307-5732-2021-303-6-111-114

12. Analysis of deep learning methods in adaptation to the small data problem solving / I. Krak et al. In: Babichev, S., Lytvynenko, V. (eds) *Lecture Notes in Data Engineering, Computational Intelligence, and Decision Making*. ISDMCI–2022. Springer, Cham. 2023. Vol. 149. P. 333–352. URL: https://doi.org/10.1007/978–3–031–16203–9_20

13. Mixed-Integer Linear Programming (MILP) Algorithms - MATLAB & Simulink. *MathWorks.com*. URL: https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html

14. Optimal scheduling of copper concentrate operations under uncertainty / P. Cheng et al. *Computers & Chemical Engineering*. 2020. Vol. 140. P. 106919. URL: https://doi.org/10.1016/j.compchemeng.2020.106919

15. Radiuk P., Barmak O., Krak I. An approach to early diagnosis of pneumonia on individual radiographs based on the CNN information technology. *The Open Bioinformatics Journal*. 2021. Vol. 14, no. 1. P. 93–107. URL: https://doi.org/10.2174/1875036202114010093

16. Radiuk P. M. Application of a genetic algorithm to search for the optimal convolutional neural network architecture with weight distribution. *Herald of Khmelnytskyi National University. Technical sciences*. 2020. Vol. 281, No. 1. P. 7–11. URL: https://doi.org/10.31891/2307–5732–2020–281–1–7–11

17. Katoch S., Chauhan S. S., Kumar V. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*. 2021. Vol. 80. P. 8091–8126. URL: https://doi.org/10.1007/s11042-020-10139-6

18. GitHub - 2black0/GA-Python: Simple Genetic Algorithm on Python. *GitHub.com*. URL: https://github.com/2black0/GA-Python

19. Грінченко Р. Передумови оптимізації управління часом. *Економіка та суспільство*. 2021. № 29. URL: https://doi.org/10.32782/2524-0072/2021-29-34

20. Venkatraj V., Dixit M. K. Challenges in implementing data-driven approaches for building life cycle energy assessment: A review. *Renewable and Sustainable Energy Reviews*. 2022. Vol. 160. P. 112327. URL: https://doi.org/10.1016/j.rser.2022.112327

21. Trackabi – Free Time Tracking, Employee Monitoring, & Leave Management Software. *Trackabi.com*. URL: https://trackabi.com/

22. Trackabi Product Tour: Web, Desktop, & Mobile Apps to Boost Productivity. *Trackabi.com*. URL: https://trackabi.com/product-tour

23. Xwavesoft - Be Focused. *Xwavesoft.com*. URL: https://xwavesoft.com/be-focused-pro-for-iphone-ipad-mac-os-x.html?utm_source=zapier.com&utm_medium=referral&utm_campaign=zapier

24. Be Focused Pro - Focus Timer | App Price Drops. *App Sliced*. URL: https://appsliced.co/app?n=pomodoro-time-focus-timer

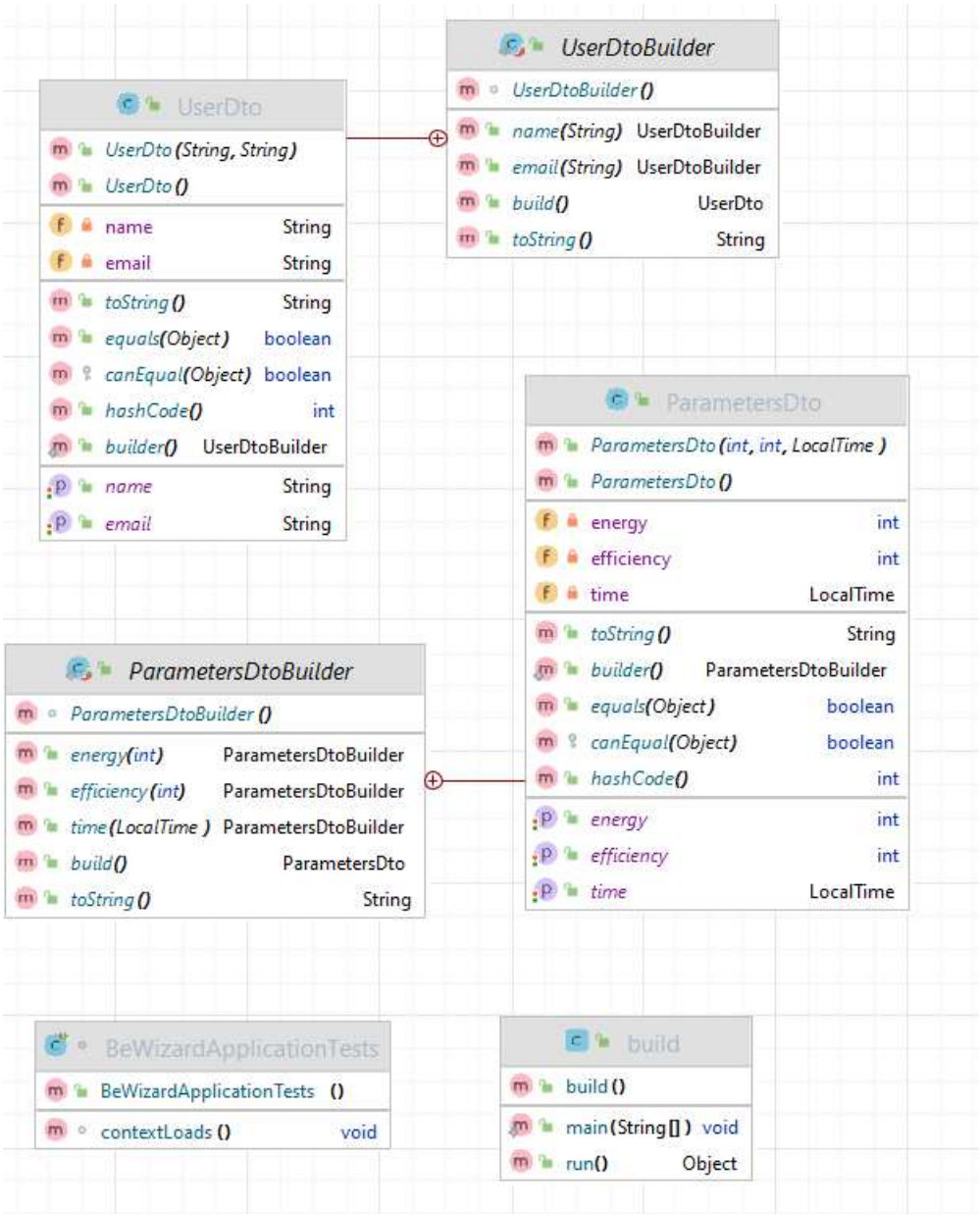
25. Wolters C. A., Brady A. C. College students' time management: A self-regulated learning perspective. *Educational Psychology Review*. 2020. Vol. 33. P. 1319–1351. URL: https://doi.org/10.1007/s10648-020-09519-z

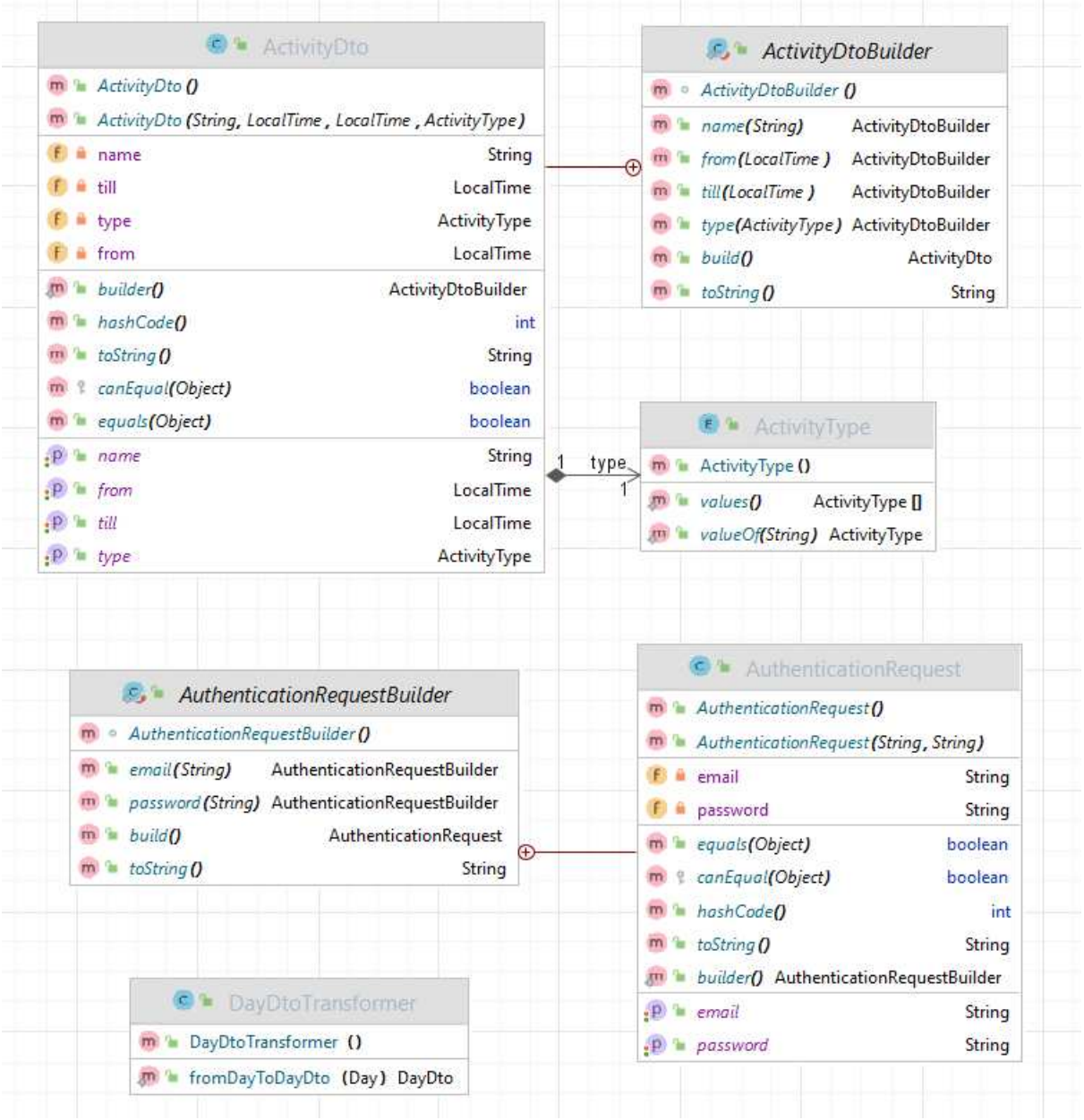
26. Cinar A. C., Kara T. B. The current state and future of mobile security in the light of the recent mobile security threat reports. *Multimedia Tools and Applications*. 2023. Vol. 82. P. 20269–20281URL: https://doi.org/10.1007/s11042-023-14400-6

ДОДАТКИ

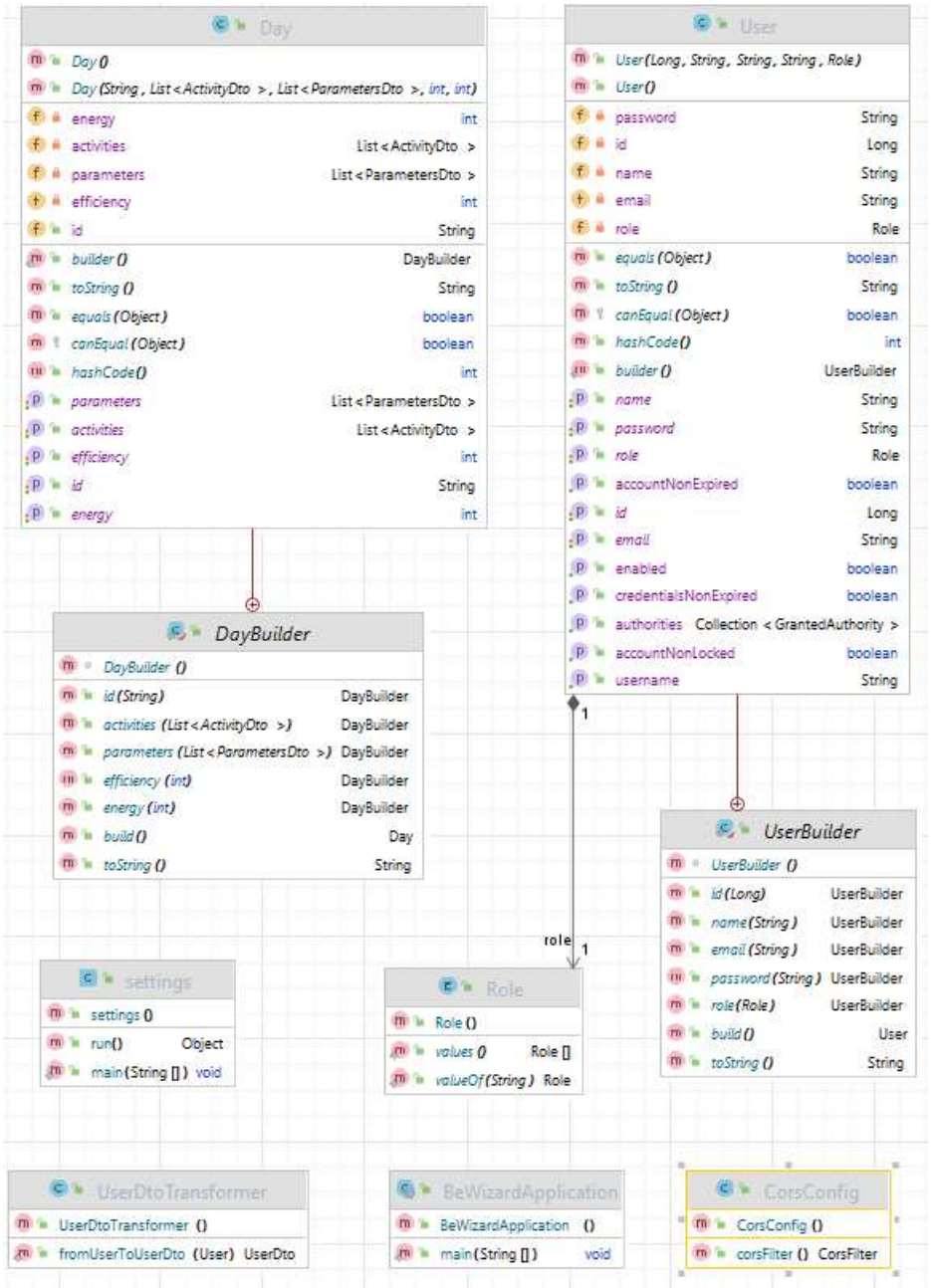
Додаток А

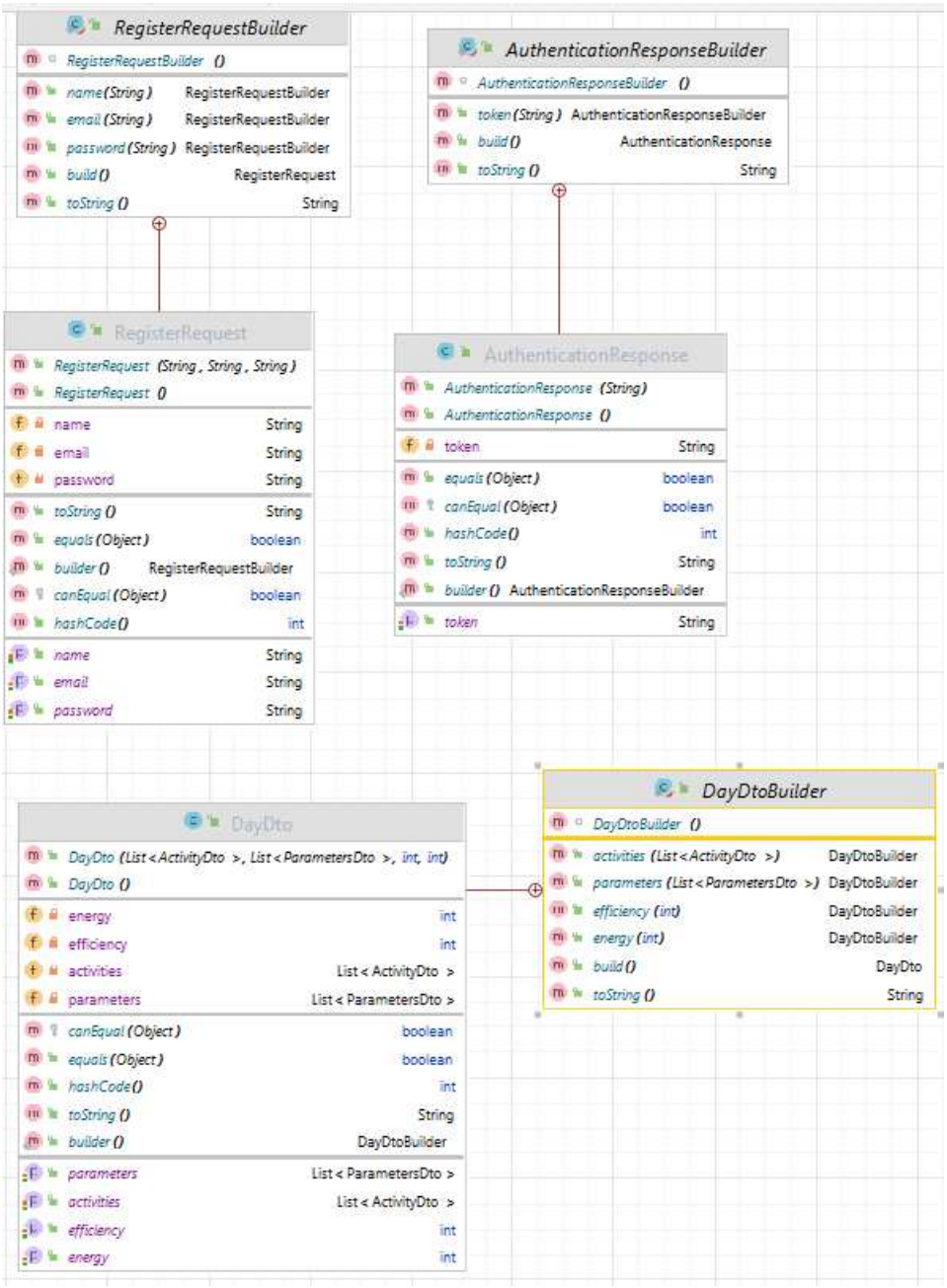
Діаграма класів мікросервісу be-wizard




ActivityDto
ActivityDto ()
ActivityDto (String, LocalTime , LocalTime , ActivityType)
name String
till LocalTime
type ActivityType
from LocalTime
builder() ActivityDtoBuilder
hashCode() int
toString() String
canEqual(Object) boolean
equals(Object) boolean
name String
from LocalTime
till LocalTime
type ActivityType
ActivityDtoBuilder
ActivityDtoBuilder ()
name(String) ActivityDtoBuilder
from(LocalTime) ActivityDtoBuilder
till(LocalTime) ActivityDtoBuilder
type(ActivityType) ActivityDtoBuilder
build() ActivityDto
toString() String
1
type
1
ActivityType
ActivityType ()
values() ActivityType []
valueOf(String) ActivityType
AuthenticationRequestBuilder
AuthenticationRequestBuilder ()
email(String) AuthenticationRequestBuilder
password(String) AuthenticationRequestBuilder
build() AuthenticationRequest
toString() String
AuthenticationRequest
AuthenticationRequest()
AuthenticationRequest(String, String)
email String
password String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
builder() AuthenticationRequestBuilder
email String
password String
DayDtoTransformer
DayDtoTransformer ()
fromDayToDayDto (Day) DayDto

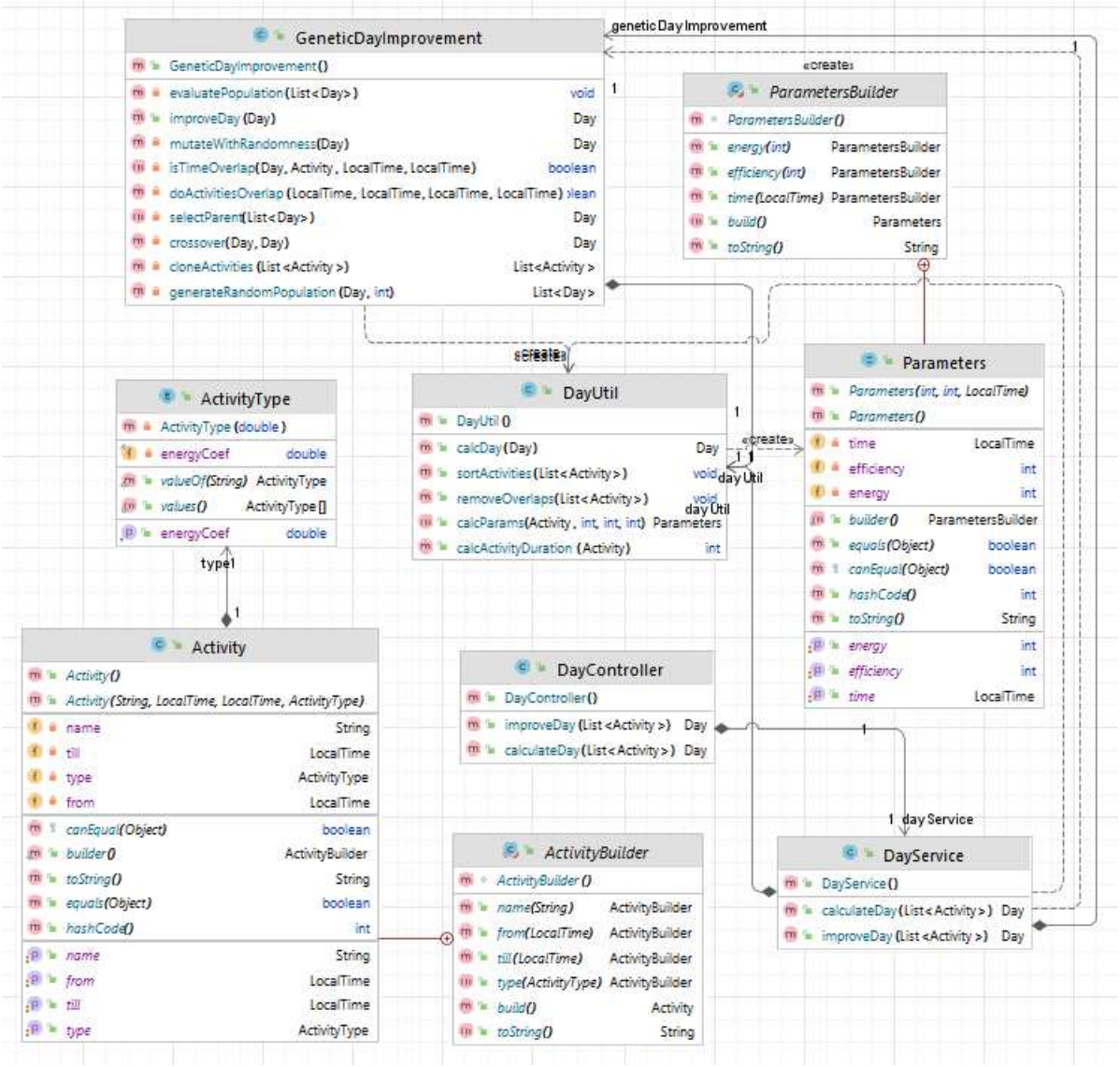

DayService
DayService ()
calculate (List<ActivityDto >, String) DayDto
improve (List<ActivityDto >, String) DayDto
create(Day) void
fetch(String) DayDto
findById (String) Day
update(DayDto, String) void
UserRepository
findByEmail (String) Optional <User>
AuthenticationController
AuthenticationController (AuthenticationService)
authenticate(AuthenticationRequest) ResponseEntity <?>
register(RegisterRequest) ResponseEntity <?>
authenticationService
UserAlreadyExistException
UserAlreadyExistException (String)
«create»
AuthenticationService
AuthenticationService (UserRepository, PasswordEncoder, DayServi
register(RegisterRequest) AuthenticationResponse
authenticate(AuthenticationRequest) AuthenticationResponse
DayRepository
findById (String) Optional <Day>
dayRepository
dayService
DayController
DayController ()
calculateDay (List<ActivityDto >) ResponseEntity <DayDto >
improveDay (List<ActivityDto >) ResponseEntity <DayDto >
day ResponseEntity <DayDto >
FeignService
improveDay (List<ActivityDto >) DayDto
calculateDay (List<ActivityDto >) DayDto
feignService
JwtSevice
JwtSevice()
isTokenExpired (String) boolean
extractExpiration (String) Date
extractClaim(String, Function<Claims, T>) T
generateToken(UserDetails) String
extractAllClaims (String) Claims
generateToken(Map<String, Object>, UserDetails) String
extractUsername(String) String
isTokenValid (String, UserDetails) boolean
signInKey Key
UserService
UserService (UserRepository)
getUserInfo (String) UserDto
userService
jwtService
jwtSevice
UserController
UserController ()
userInfo ResponseEntity <UserDto>
JwtAuthenticationFilter
JwtAuthenticationFilter (JwtSevice, UserDetailsService)
doFilterInternal (HttpServletRequest, HttpServletResponse, FilterC
jwtAuthFilter
ApplicationConfig
ApplicationConfig (UserRepository)
authenticationManager (AuthenticationConfiguration) AuthenticationManager
passwordEncoder () PasswordEncoder
userDetailsService () UserDetailsService
authenticationProvider () AuthenticationProvider
SecurityConfig
SecurityConfig (JwtAuthenticationFilter, AuthenticationProvider
securityFilterChain (HttpSecurity) SecurityFilterChain

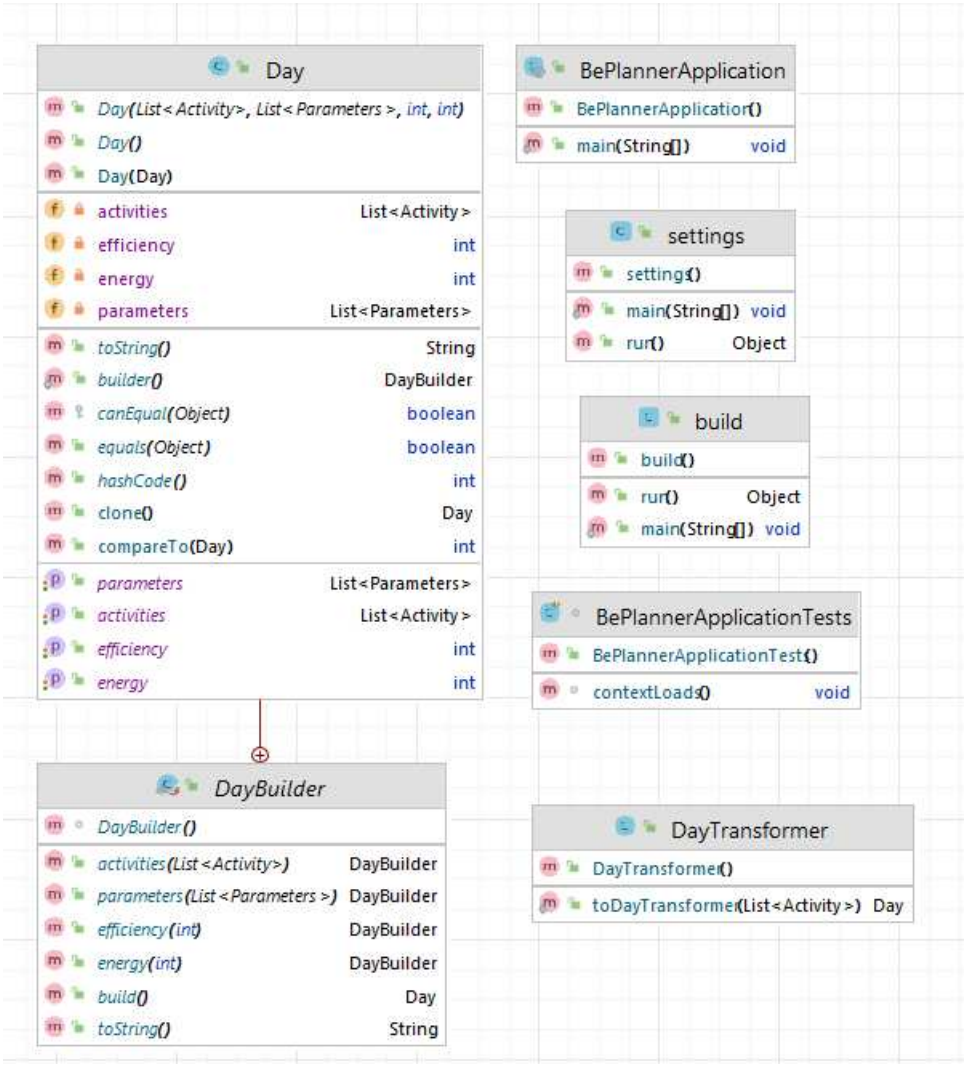

Day
Day()
Day(String, List<ActivityDto>, List<ParametersDto>, int, int)
energy int
activities List<ActivityDto>
parameters List<ParametersDto>
efficiency int
id String
builder() DayBuilder
toString() String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
parameters List<ParametersDto>
activities List<ActivityDto>
efficiency int
id String
energy int
User
User(Long, String, String, String, Role)
User()
password String
id Long
name String
email String
role Role
equals(Object) boolean
toString() String
canEqual(Object) boolean
hashCode() int
builder() UserBuilder
name String
password String
role Role
accountNonExpired boolean
id Long
email String
enabled boolean
credentialsNonExpired boolean
authorities Collection<GrantedAuthority>
accountNonLocked boolean
username String
DayBuilder
DayBuilder()
id(String) DayBuilder
activities(List<ActivityDto>) DayBuilder
parameters(List<ParametersDto>) DayBuilder
efficiency(int) DayBuilder
energy(int) DayBuilder
build() Day
toString() String
UserBuilder
UserBuilder()
id(Long) UserBuilder
name(String) UserBuilder
email(String) UserBuilder
password(String) UserBuilder
role(Role) UserBuilder
build() User
toString() String
1
role
1
settings
settings()
run() Object
main(String[]) void
Role
Role()
values() Role[]
valueOf(String) Role
UserDtoTransformer
UserDtoTransformer()
fromUserToUserDto(User) UserDto
BeWizardApplication
BeWizardApplication()
main(String[]) void
CorsConfig
CorsConfig()
corsFilter() CorsFilter


RegisterRequestBuilder
RegisterRequestBuilder ()
name(String) RegisterRequestBuilder
email(String) RegisterRequestBuilder
password(String) RegisterRequestBuilder
build() RegisterRequest
toString() String
AuthenticationResponseBuilder
AuthenticationResponseBuilder ()
token(String) AuthenticationResponseBuilder
build() AuthenticationResponse
toString() String
RegisterRequest
RegisterRequest(String, String, String)
RegisterRequest()
name String
email String
password String
toString() String
equals(Object) boolean
builder() RegisterRequestBuilder
canEqual(Object) boolean
hashCode() int
name String
email String
password String
AuthenticationResponse
AuthenticationResponse(String)
AuthenticationResponse()
token String
equals(Object) boolean
canEqual(Object) boolean
hashCode() int
toString() String
builder() AuthenticationResponseBuilder
token String
DayDto
DayDto(List<ActivityDto>, List<ParametersDto>, int, int)
DayDto()
energy int
efficiency int
activities List<ActivityDto>
parameters List<ParametersDto>
canEqual(Object) boolean
equals(Object) boolean
hashCode() int
toString() String
builder() DayDtoBuilder
parameters List<ParametersDto>
activities List<ActivityDto>
efficiency int
energy int
DayDtoBuilder
DayDtoBuilder()
activities(List<ActivityDto>) DayDtoBuilder
parameters(List<ParametersDto>) DayDtoBuilder
efficiency(int) DayDtoBuilder
energy(int) DayDtoBuilder
build() DayDto
toString() String

Додаток Б

Діаграма класів мікросервісу be-planner




Day
Day(List<Activity>, List<Parameters>, int, int)
Day()
Day(Day)
activities List<Activity>
efficiency int
energy int
parameters List<Parameters>
toString() String
builder() DayBuilder
canEqual(Object) boolean
equals(Object) boolean
hashCode() int
clone() Day
compareTo(Day) int
parameters List<Parameters>
activities List<Activity>
efficiency int
energy int
DayBuilder
DayBuilder()
activities(List<Activity>) DayBuilder
parameters(List<Parameters>) DayBuilder
efficiency(int) DayBuilder
energy(int) DayBuilder
build() Day
toString() String
BePlannerApplication
BePlannerApplication()
main(String[]) void
settings
settings()
main(String[]) void
run() Object
build
build()
run() Object
main(String[]) void
BePlannerApplicationTests
BePlannerApplicationTest()
contextLoads() void
DayTransformer
DayTransformer()
toDayTransformer(List<Activity>) Day

Додаток В

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

МЕТОД ПОКРАЩЕННЯ ПЛАНУВАННЯ РОЗПОРЯДКУ ДНЯ ЗАСОБАМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Виконав:
студент 4 курсу, групи КН-20-1
Кенс Олександр Володимирович

Керівник:
к.т.н., старший викладач кафедри КН
Радюк Павло Михайлович

2

Актуальність

На сьогодні, сучасний світ насичений різноманітними викликами та завданнями, тому ефективне управління часом стає важливою умовою успішності будь-якої людської діяльності.

Ріст потреб щодо покращення розпорядку дня та планування різноманітних діяльностей породжує необхідність у впровадженні до розв'язання цього завдання інноваційних підходів та інформаційних технологій.

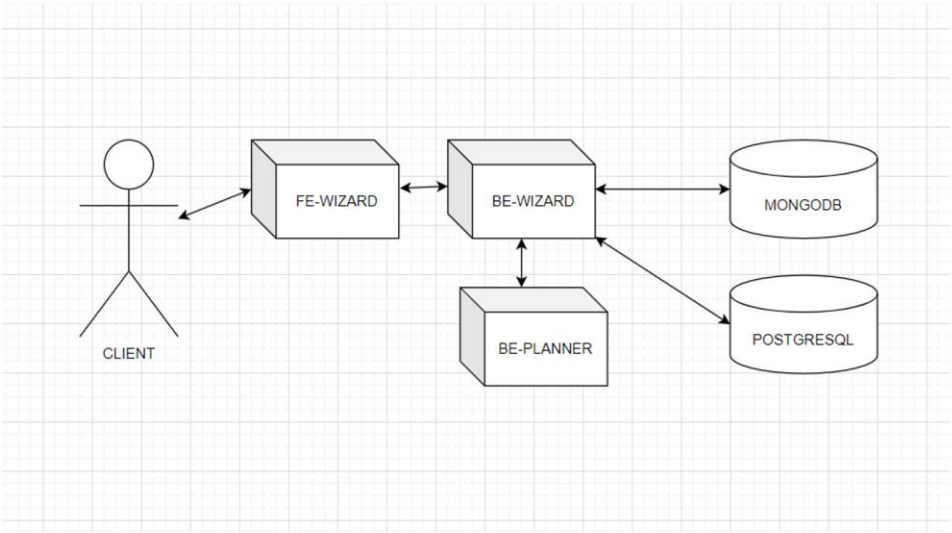
Тому актуальним є розроблення та впровадження **методу**, що дасть можливість покращити планування розпорядку дня за допомогою інтелектуального аналізу даних.

Мета і задачі роботи

Метою кваліфікаційної роботи бакалавра є покращення планування розпорядку дня засобами ІАД у вигляді вебсервісу. Для досягнення мети кваліфікаційної роботи необхідно виконати такі завдання:

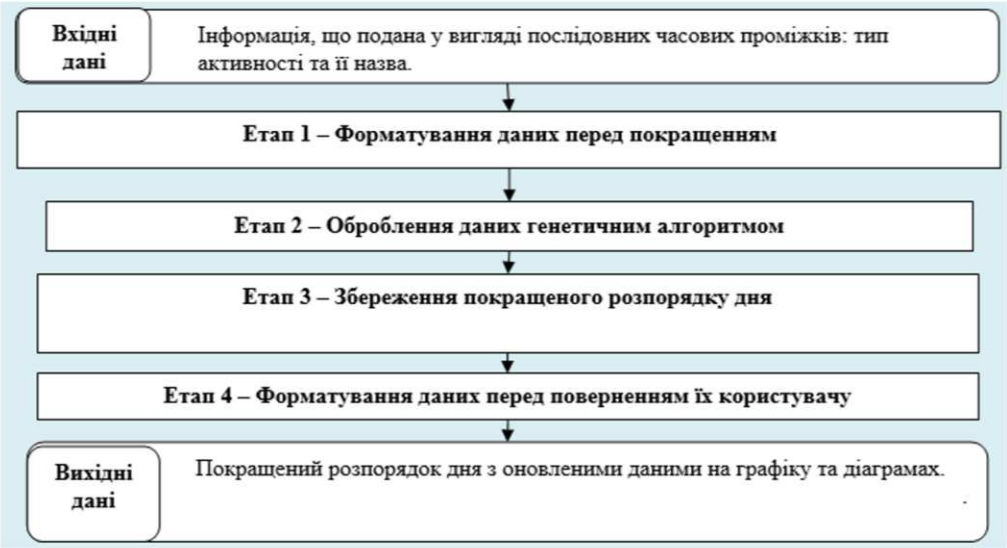
1. Провести ретельний аналіз наявних методів, засобів та підходів інтелектуального аналізу даних до управління часом та планування розпорядку дня засобами інтелектуального аналізу даних.
2. Розробити метод покращення планування розпорядку дня засобами ІАД.
3. Спроєктувати вебсервіс для покращення розпорядку дня на основі розробленого методу.
4. Виконати програмну реалізацію вебсервісу за розробленим методом покращення планування розпорядку дня.
5. Здійснити тестування розробленого методу покращення планування розпорядку дня та створеного на його основі вебсервісу.
6. Продемонструвати покращення планування розпорядку дня через впровадження розробленого методу.

Загальна архітектура методу покращення планування розпорядку дня засобами інтелектуального аналізу даних.



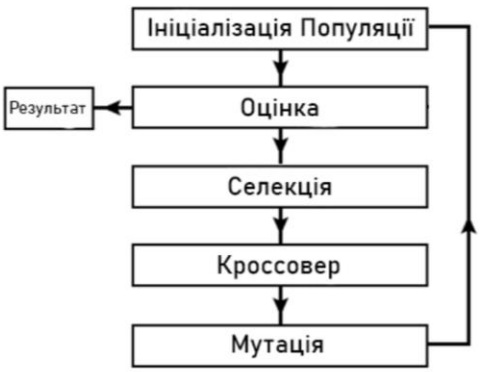
5

Основні етапи методу покращення планування розпорядку дня засобами інтелектуального аналізу даних.



6

Покращення розпорядку дня за допомогою генетичного алгоритму

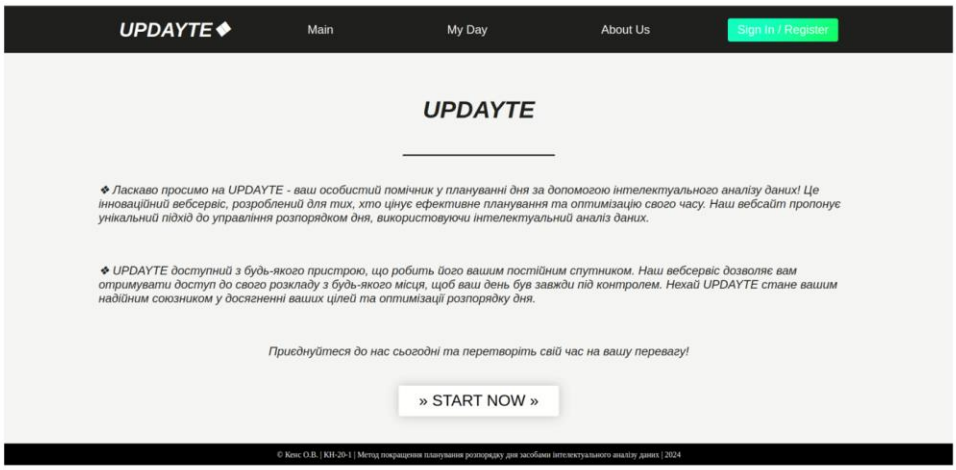


- **Ініціалізація популяції:** Створення початкового набору рішень на основі розпорядку дня користувача.
- **Оцінка:** Кожен варіант розпорядку дня оцінюється за допомогою функції пристосованості.
- **Селекція:** Обираються кращі рішення на основі їх пристосованості.
- **Кросовер:** Обмін активностями між двома найкращими вибраними індивідами для створення нового варіанту розпорядку дня.
- **Мутація:** Випадкові зміни в часових проміжках розпорядку дня для забезпечення різноманітності в популяції.
- **Повторна Ініціалізація популяції:** Створення набору рішень на основі мутованого розпорядку дня.
- **Повторна оцінка:** Кожен варіант розпорядку дня оцінюється за допомогою функції пристосованості. Якщо параметри задовільні, буде повернуто результат, в іншому випадку цикл продовжиться до тих пір доки не буде знайдено оптимальний розпорядок дня, або буде досягнута максимальна кількість ітерацій.

Оцінка розпорядку дня користувача

- Оцінка розпорядку дня користувача визначається двома основними параметрами – Енергією та Ефективністю.
- Енергія відображає рівень сил та бадьорості користувача. Цей параметр важливий для визначення загального фізичного та емоційного благополуччя користувача.
- Ефективність відображає, наскільки продуктивно та ефективно користувач витрачає свій час та енергію на виконання різних завдань. Цей параметр допомагає виявити, наскільки ефективно розподілені ресурси користувача, і чи можна оптимізувати його розпорядок дня для досягнення кращих результатів та балансу між роботою та відпочинком.

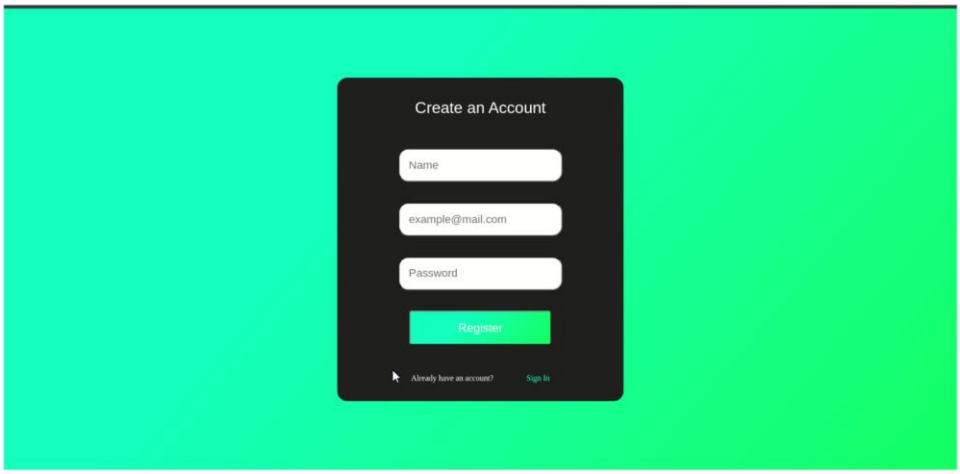
Програмна реалізація методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.



Головна сторінка

9

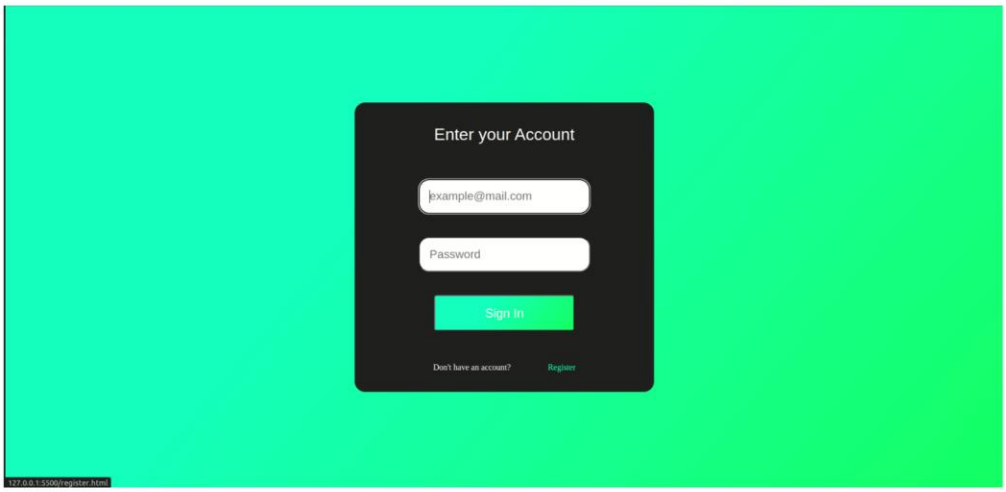
Програмна реалізація методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.



Сторінка реєстрації

10

Програмна реалізація методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.



Сторінка для входу в наявний аккаунт

11

Програмна реалізація методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

Кабінет користувача для аналізу та управління розпорядком дня

12

Результати роботи методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

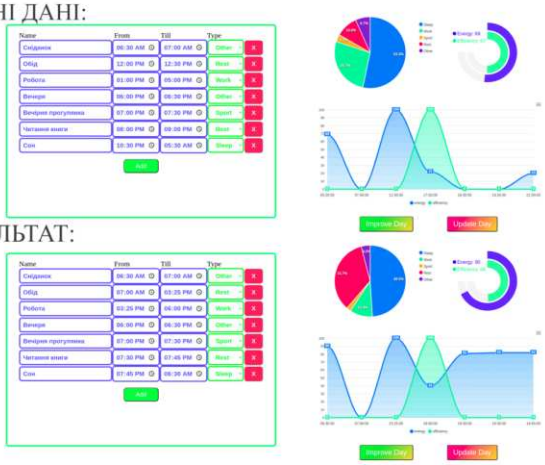
Тест 1

ВХІДНІ ДАНІ:

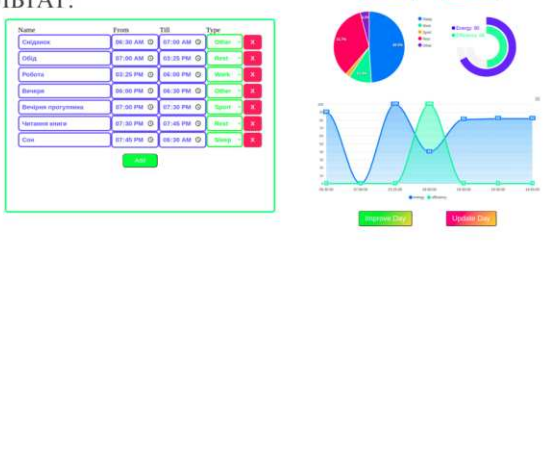
РЕЗУЛЬТАТ:

Тест 2

ВХІДНІ ДАНІ:



РЕЗУЛЬТАТ:



Результати роботи методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

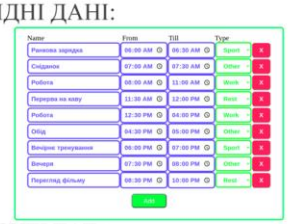
Тест 3

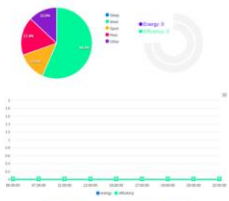
ВХІДНІ ДАНІ:

РЕЗУЛЬТАТ:

Тест 4

ВХІДНІ ДАНІ:





РЕЗУЛЬТАТ:

Висновки

1. У результаті виконання кваліфікаційної роботи бакалавра було успішно досягнуто мету роботи, а саме покращено планування розпорядку дня через впровадження методу покращення планування розпорядку дня засобами інтелектуального аналізу даних у вигляді вебсервісу.

2. Проведено успішне тестування розробленого методу покращення розпорядку дня на створеного на його основі вебсервісу.

3. Отже, ретельне проєктування та програмна реалізація методу покращення розпорядку дня дали змогу створити функціональний та надійний інструмент для управління часом.

4. Створений вебсервіс може бути корисним користувачам, щоби підвищити їхню продуктивність, зменшити стрес та досягти кращого балансу між професійним та особистим життям.

ДЯКУЮ ЗА УВАГУ!

Додаток Г

Програмні коди

```
package com.wizard.controller;


import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.wizard.dto.ActivityDto;
import com.wizard.dto.DayDto;
import com.wizard.service.DayService;


@RestController
@RequestMapping("/wizard/day")
public class DayController {

   @Autowired
   DayService dayService;

   @CrossOrigin
   @GetMapping
   @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
   public ResponseEntity<DayDto> getDay() {
      UserDetails userDetails = (UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
      return ResponseEntity.ok(dayService.fetch(userDetails.getUsername()));
   }

   @CrossOrigin
   @PostMapping("/improve")
   @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
   public ResponseEntity<DayDto> improveDay(@RequestBody List<ActivityDto> activities) {
      System.out.println("Received activities: " + activities);
        UserDetails userDetails = (UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
      return ResponseEntity.ok(dayService.improve(activities, userDetails.getUsername()));
   }

   @CrossOrigin
   @PostMapping("/calculate")
   @PreAuthorize("hasRole('USER') or hasRole('ADMIN')")
   public ResponseEntity<DayDto> calculateDay(@RequestBody List<ActivityDto> activities) {
      System.out.println("Received activities: " + activities);
      UserDetails userDetails = (UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
      return ResponseEntity.ok(dayService.calculate(activities, userDetails.getUsername()));
   }
}
package com.wizard.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.wizard.dto.UserDto;
import com.wizard.service.UserService;
```

```
@RestController
@RequestMapping(“/wizard/user”)
public class UserController {

   @Autowired
   UserService userService;

   @CrossOrigin
   @GetMapping
   @PreAuthorize(“hasRole(‘USER’) or hasRole(‘ADMIN’)”)
   public ResponseEntity<UserDto> getUserInfo() {
      UserDetails userDetails = (UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
      return ResponseEntity.ok(userService.getUserInfo(userDetails.getUsername()));
   }

}

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Day {
   @Id
   public String id;
    private List<ActivityDto> activities;
    private List<ParametersDto> parameters;
    private int efficiency;
    private int energy;
}


package com.planner.util;

import java.time.LocalTime;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

import com.planner.enums.ActivityType;
import com.planner.model.Activity;
import com.planner.model.Day;

public class GeneticDayImprovement {

    private final int POPULATION_SIZE = 10;
    private final int GENERATION_COUNT = 100;
    DayUtil dayUtil = new DayUtil();

    public Day improveDay(Day oldDay) {
       dayUtil.calcDay(oldDay);
        List<Day> population = generateRandomPopulation(oldDay, POPULATION_SIZE);

        for (int generation = 0; generation < GENERATION_COUNT; generation++) {
            evaluatePopulation(population);

            Collections.sort(population, Collections.reverseOrder());

            Day bestDay = population.get(0);

            List<Day> newPopulation = new ArrayList<>();
            newPopulation.add(bestDay);

            for (int i = 1; i < POPULATION_SIZE; i++) {
                Day parent1 = selectParent(population);
                Day parent2 = selectParent(population);

                Day child = crossover(parent1, parent2);
                mutateWithRandomness(child);

                newPopulation.add(child);
            }

            population = newPopulation;
        }
```

```
        Collections.sort(population, Collections.reverseOrder());
        return dayUtil.calcDay(population.get(0));
    }

    private List<Day> generateRandomPopulation(Day oldDay, int populationSize) {
        List<Day> population = new ArrayList<>();
        for (int i = 0; i < populationSize; i++) {
            population.add(mutateWithRandomness(oldDay.clone()));  // Introduce more randomness
        }
        return population;
    }

    private Day mutateWithRandomness(Day day) {

       // Check if there is a SLEEP activity
        boolean hasSleep = day.getActivities().stream()
                .anyMatch(activity -> activity.getType() == ActivityType.SLEEP);

        // IF NO SLEEP
        if (!hasSleep) {
            // Add SLEEP activity from 22:00 to 6:00
            day.getActivities().add(Activity.builder()
                  .name("Sleep")
                    .from(LocalTime.of(22, 0))
                    .till(LocalTime.of(6, 0))
                    .type(ActivityType.SLEEP)
                    .build());
        }

        dayUtil.calcDay(day);

        Random random = new Random();
        int maxMutations = 500;

        while((day.getEnergy() < 85 || day.getEfficiency() < 85) && maxMutations > 0) {
           int index = random.nextInt(day.getActivities().size());
            Activity randomActivity = day.getActivities().get(index);

            Activity mutatedActivity = Activity.builder()
                  .name(randomActivity.getName())
                    .from(randomActivity.getFrom())
                    .till(randomActivity.getTill())
                    .type(randomActivity.getType())
                    .build();

            LocalTime newFrom = randomActivity.getFrom();
            LocalTime newTill = randomActivity.getTill();


            if (randomActivity.getType() != ActivityType.OTHER && randomActivity.getType() !=
ActivityType.SPORT) {
               if(day.getEfficiency() < day.getEnergy() && day.getEnergy() > 60) {
                  if ((randomActivity.getType() == ActivityType.WORK)){
                      newTill = randomActivity.getTill().plusMinutes(random.nextInt(6) * 5);
                        newFrom = randomActivity.getFrom().minusMinutes(random.nextInt(6) * 5);
                   }
               }
               else {
                  if ((randomActivity.getType() == ActivityType.WORK &&
dayUtil.calcActivityDuration(randomActivity) > 120)){
                      newTill = randomActivity.getTill().minusMinutes(random.nextInt(6) * 5);
                        newFrom = randomActivity.getFrom().plusMinutes(random.nextInt(6) * 5);
                   }
                  else {
                     newTill = randomActivity.getTill().plusMinutes(random.nextInt(6) * 5);
                        newFrom = randomActivity.getFrom().minusMinutes(random.nextInt(6) * 5);
                  }
               }

            }

            if (!isTimeOverlap(day, mutatedActivity, newFrom, newTill)) {
                mutatedActivity.setFrom(newFrom);
                mutatedActivity.setTill(newTill);
                day.getActivities().set(index, mutatedActivity);
            }
```

```
            maxMutations--;
            dayUtil.calcDay(day);
        }

        return day;
    }

    private void evaluatePopulation(List<Day> population) {
        for (Day day : population) {
            Day improvedDay = dayUtil.calcDay(day);
            int improvedDayParams = (improvedDay.getEfficiency() + improvedDay.getEnergy()) / 2;
            day.setEnergy(Math.min(improvedDay.getEfficiency(), 100));
            day.setEfficiency(Math.min(improvedDayParams, 100));
        }
    }

    private Day selectParent(List<Day> population) {
        Random random = new Random();
        int index = random.nextInt(population.size());
        return population.get(index);
    }

    private Day crossover(Day parent1, Day parent2) {
        int size = Math.min(parent1.getActivities().size(), parent2.getActivities().size());

        // Ensure fromIndex is less than toIndex
        int fromIndex = new Random().nextInt(size);
        int toIndex = new Random().nextInt(size - fromIndex) + fromIndex + 1;

        List<Activity> childActivities = new ArrayList<>();
        childActivities.addAll(parent1.getActivities().subList(0, fromIndex));
        childActivities.addAll(parent2.getActivities().subList(fromIndex, toIndex));
        childActivities.addAll(parent1.getActivities().subList(toIndex, parent1.getActivities().size()));

        // Deep clone the Day object
        return Day.builder()
                .activities(cloneActivities(childActivities))
                .energy(parent1.getEnergy())
                .efficiency(parent1.getEfficiency())
                .build();
    }


    // Метод для глибокого клонування списку Activity
    private List<Activity> cloneActivities(List<Activity> activities) {
        List<Activity> clonedActivities = new ArrayList<>();
        for (Activity activity : activities) {
            clonedActivities.add(Activity.builder()
                  .name(activity.getName())
                    .from(activity.getFrom())
                    .till(activity.getTill())
                    .type(activity.getType())
                    .build());
        }
        return clonedActivities;
    }

    private boolean isTimeOverlap(Day day, Activity currentActivity, LocalTime newFrom, LocalTime newTill) {
        for (Activity otherActivity : day.getActivities()) {
            if (!currentActivity.equals(otherActivity) && doActivitiesOverlap(newFrom, newTill,
otherActivity.getFrom(), otherActivity.getTill())) {
                return true;
            }
        }
        return false;
    }

    private boolean doActivitiesOverlap(LocalTime from1, LocalTime till1, LocalTime from2, LocalTime till2) {
        return (from1.isBefore(till2) && till1.isAfter(from2)) || (from2.isBefore(till1) &&
till2.isAfter(from1));
    }
}
```

Wed Jun 19 10:56:48 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 131500 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних Додано в БД: 2024-06-19 Автора: Олександр КЕНС Керівники: Павло РАДЮК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	79767	1210	3660 (5%)	58 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КН

Дата перевірки:
19.06.2024 12:15:02 EEST

Дата звіту:
19.06.2024 12:16:35 EEST

ID перевірки:
1016374877

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005671

Назва документа: КН-20-1 Кенс_ЗАПИСКА

Кількість сторінок: 71 Кількість слів: 12375 Кількість символів: 99781 Розмір файлу: 2.75 MB ID файлу: 1016182724

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

9.66%
Схожість

Найбільша схожість: 3.85% з джерелом з Бібліотеки (ID файлу: 1016162161)

5.88% Джерела з Інтернету	827	Сторінка 73
7.37% Джерела з Бібліотеки	145	Сторінка 77

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	1
Підозріле форматування	11 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних*

Автор: *студент групи КН-20-1 Кенс Олександр*

Спеціальність: *122 Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *док. філ., ст. викл. Радюк П.М.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, що виявлені в роботі Кенса О.В., не є плагіатом, оскільки: запозичення знайдені в розділі огляду наявних підходів та не описують безпосередньо авторську роботу й не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять цитування джерел у переліку посилань; поміж запозичень є загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

– за системою Anti-Plagiarism: 3.0 %;

– за системою Unicheck: 9.66 %.

Отже, запозичення є допустимими та належать до описаних вище й адресуються до першоджерел, що, з урахуванням наведених обґрунтувань, свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ *Павло РАДЮК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *КН-20-1 Кенса Олександра Володимировича*

за темою *Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних*

1. Актуальність теми

На сьогодні, за бурливих та мінливих умов сучасного світу, вміння управляти часом відіграє все більшу роль у житті людини. Автоматизовані системи управління на основі засобів інтелектуального аналізу даних можуть суттєво покращити процес управління та планування розпорядку дня. Заразом впровадження методів комп'ютерних наук може значно покращити планування розпорядку дня користувача, з огляду на його активності та індивідуальні особливості. Тому розроблення методу покращення розпорядку дня є важливим та актуальним для покращення рівня продуктивності діяльності людини.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом спеціальності 122 Комп'ютерні науки, об'єктами вивчення та діяльності комп'ютерних наук є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів, а також методи й технології отримання, зберігання, оброблення, передачі та використання інформації. Метою даної кваліфікаційної роботи є покращення планування розпорядку дня засобами інтелектуального аналізу даних та створення вебсервісу для програмної реалізації цієї задачі. Під час виконання роботи студентом були використані методи та алгоритми розв'язання теоретичних і прикладних задач. Отже, результати виконання кваліфікаційної роботи бакалавра повністю відповідають стандарту бакалавра спеціальності 122 Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

У процесі виконання кваліфікаційної роботи бакалавра Кенс Олександр Володимирович проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені завдання. Під час розроблення прикладного програмного забезпечення та написання пояснювальної записки студент успішно засвоїв усі необхідні компетентності та професійні навички. Кенс О.В. продемонстрував результати навчання,

що повністю відповідають виконанню освітньо-професійної програми рівня вищої освіти «Бакалавр» за спеціальністю 122 Комп'ютерні науки.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Результати роботи та їхня обґрунтована практична значущість одержані та обумовлені студентом особисто внаслідок виконання ним усіх поставлених завдань.

5. Ступінь опанування методами дослідження

Під час розроблення методу покращення планування розпорядку дня засобами інтелектуального аналізу даних та створення на його основі вебсервісу студент Кенс О.В. продемонстрував достатній рівень компетентностей та володіння необхідними інструментами й обладнанням, методами, методиками та технологіями галузі 12 Інформаційні технології.

6. Повнота та якість розкриття теми роботи

Тема роботи повністю обґрунтована й розкрита; актуальність предметної галузі та відомі дослідження щодо обраної тематики проаналізовані достатньо. Усі поставлені завдання в роботі виконані повно, проте із незначною затримкою в часі. Розроблений метод покращення розпорядку дня та створений його основі вебсервіс відповідають технічним вимогам спеціальності 122 Комп'ютерні науки.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура кваліфікаційної роботи Кенса О.В. та послідовність викладення є логічними та повністю відповідають поставленій меті. Викладення матеріалу є послідовним, аргументованим та літературно грамотним, проте з кількома стилістичними помилками.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Спроєктований у роботі метод та його програмна реалізація можуть бути використані користувачами в їхньому повсякденному житті для покращення планування розпорядку дня.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*З огляду на достатній рівень виконання та забезпечення всіх необхідних вимог, вважаю, що кваліфікаційна робота Кенса Олександра Володимировича може бути допущена до захисту. Рекомендована оцінка – **«добре»**.*

Керівник _____ *док. філ., ст. викл. каф. КН Павло РАДЮК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *КН-20-1 Кенса Олександра Володимировича*
за темою: *Метод покращення планування розпорядку дня засобами інтелектуального аналізу даних*

1. Актуальність обраної теми

Актуальність обраної теми зумовлена ростом потреби щодо вдосконалення управління часом, що є важливим аспектом у сучасному суспільстві. Налаштування розпорядку дня та його покращення засобами інтелектуального аналізу даних може сприяти підвищенню якості життя користувачів. Проєктування методу покращення планування розпорядку дня за допомогою генетичного алгоритму та створення на його основі вебсервісу є важливим і актуальним завданням у галузі 12 Інформаційні технології.

2. Повнота розкриття мети та завдань роботи

У роботі достатньо розкрито мету та завдання дослідження. Студентом визначено методи та засоби для програмної реалізації покращення управління часом. Проаналізовано наявні методи та підходи до розв'язання задачі, а також наведено детальний огляд програмних засобів для покращення управління часом. Мету роботи успішно досягнуто через розроблення та впровадження вебсервісу для покращення планування розпорядку дня, що ґрунтується на інтелектуальному аналізі даних, зокрема на генетичному алгоритмові.

3. Зміст кожного розділу роботи

У першому розділі проведено аналіз проблеми управління часом, розглянуто методи та підходи до розв'язання задачі, а також здійснено огляд наявних програмних засобів. Студентом правильно формалізовано мету, задачі та вимоги до програмної реалізації методу. Другий розділ присвячено проєктуванню методу покращення розпорядку та проєктуванню вебсервісу на його основі. У розділі детально описано метод покращення планування розпорядку дня засобами інтелектуального аналізу даних, особливості отримання вхідної інформації та реалізації методу. Наведено проєктну архітектуру системи та взаємозв'язок компонентів. У третьому розділі описано програмну реалізацію методу у вигляді вебсервісу. Наведено опис тестування вебсервісу, розглянуто вимоги до розгортання та надано інструкцію користувача. Насамкінець оформлено висновки до виконаних у роботі завдань.

4. Оцінка розробленого програмного забезпечення, його практична цінність

Розроблений вебсервіс демонструє хороший рівень виконання та покращення планування розпорядку дня користувачів. Генетичний алгоритм дає змогу підібрати активності користувача, підвищуючи його загальну енергію ефективність упродовж дня. Вебсервіс має значну практичну цінність, оскільки тайм-менеджмент у сучасному світі є обов'язковою частиною успішного життя людини.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота бакалавра оформлена відповідно до вимог. Виклад матеріалу структурований та логічно послідовний. Текст роботи написаний чітко та зрозуміло, з використанням наукової термінології. Графічні матеріали, як от діаграми та схеми, допомагають краще зрозуміти зміст спроєктованого методу та створеного на його основі вебсервісу.

6. Недоліки кваліфікаційної роботи бакалавра

До недоліків роботи варто віднести обмежений функціонал розробленого вебсервісу. До системи варто було додати інтеграцію зі сторонніми календарями та іншими програмами для управління часом. Під час користування вебсервісом користувач може зустрічати нереалістичні сценарії розпорядку дня за малої кількості введених ним активностей. Проте попри ці недоліки, загальний рівень виконання роботи є достатнім.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка, на яку заслуговує кваліфікаційна робота.

*З огляду на хороший рівень виконання та забезпечення всіх необхідних вимог до кваліфікаційної роботи бакалавра, вважаю, що подана робота Кенса О.В. може бути допущена до захисту. Рекомендована оцінка – «**добре**».*

Рецензент к.т.н., доцент, доцент каф. КІС Гнатчук Є.Г. 19.06.24