Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного застосунку для моніторингу та аналізу активності користувача»

Виконав: студент 4 курсу
групи ЗПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кондратенко Ю.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф ОТ Городецька О.С.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ _____ Романюк О. Н.
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кондратенко Юрію Вадимовичу

1. Тема роботи – розробка програмного застосунку для моніторингу та аналізу активності користувача.

Керівник роботи: Ракитянська Ганна Борисівна, кандидат техн. наук, доцент кафедри програмного забезпечення, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки – Eclipse, Visual Studio Code, мови розробки – Java, HTML, JavaScript, фреймворк – Spring засоби організації даних програми – база даних PostgreSQL, операційна система – Windows.

4. Зміст текстової частини: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка структури та алгоритмів програмного застосунку; розробка модулів програмного застосунку; тестування програмного застосунку; висновки; перелік посилань; додатки; графічна частина.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми, мета, об'єкт та предмет дослідження, новизна отриманих результатів, порівняльний аналіз аналогів, розробка алгоритмів системи, розробка інтерфейсу, тестування системи, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ракитянська Г.Б., к.т.н., доц. кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та обґрунтування завдяння	14.03.2024 – 29.03.2024	вик.
2	Аналіз вхідних та вихідних даних системи	30.03.2024 – 07.04.2024	вик
3	Розробка моделі та алгоритмів системи	08.04.2024 – 26.04.2024	вик.
4	Розробка інтерфейсу	27.04.2024 – 04.05.2024	вик.
5	Розробка програмних модулів системи	05.05.2024 – 14.05.2024	вик.
6	Тестування системи	15.05.2024– 20.05.2024	вик.
7	Оформлення матеріалів до захисту БДР	21.05.2024 – 30.05.2024	вик.

Студент _____ **Ю.В.Кондратен**
(підпис) (ініціали та прізви

Керівник бакалаврської дипломної роботи _____ **Г.Б.Ракитянськ**
(підпис) (ініціали та прізви

АНОТАЦІЯ

Бакалаврська дипломна робота складається зі 120 сторінок формату А4, на яких є 30 рисунків, 3 таблиць, список використаних джерел містить 23 найменування.

У бакалаврській дипломній роботі проведено аналіз стану програмних застосунків для моніторингу та аналізу активності користувача та постановка задач дослідження. Було проведено порівняльний аналіз аналогів програного застосунку, визначено їх переваги та недоліки та доведено доцільність розробки програмного застосунку.

Під час виконання роботи було створено програмний застосунок, що дозволяє користувачам реєструвати свою діяльність за комп’ютером, переглядати детальну інформацію про проведений час, отримувати статистку та аналіз збережених даних.

Застосунок розроблено з використанням мов програмування Java, HTML та JavaScript та середовищ розробки програмного забезпечення Eclipse та Visual Studio Code.

Отримані в бакалаврській дипломній роботі результати можуть бути використані компаніями для аналізу та підвищення продуктивності офісних працівників.

Ключові слова: моніторинг активності, продуктивність, аналіз даних, програмний застосунок.

ABSTRACT

The bachelor's thesis consists of 120 A4 pages, with 30 figures, 3 tables, and a list of references containing 23 titles.

The bachelor's thesis analyses the state of the art of software applications for monitoring and analysing user activity and sets research objectives. A comparative analysis of analogues of the software application was carried out, their advantages and disadvantages were identified, and the feasibility of developing a software application was proved.

In the course of the work, a software application was created that allows users to record their computer activity, view detailed information about the time spent, obtain statistics and analyse the stored data.

The application was developed using the Java, HTML and JavaScript programming languages and the Eclipse and Visual Studio Code software development environments.

The results obtained in the bachelor's thesis can be used by companies to analyse and improve the productivity of office workers.

Keywords: activity monitoring, productivity, data analysis, software application.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі, де швидкість і ефективність стали необхідністю в нашому щоденному житті, аналіз продуктивності стає ключовим аспектом. Тому набуває особливої важливості здатність знаходити ефективні способи оптимізувати свій час і зусилля, щоб досягти максимальної продуктивності. Аналізуючи діяльність і використовуючи різноманітні інструменти та стратегії, можна забезпечити перевагу в цьому постійно змінюваному і конкурентному середовищі.

Аналіз продуктивності дозволяє краще розуміти, як витрачається час, які завдання займають більше часу, а також виявляти можливість оптимізації наших робочих процесів. За допомогою цього аналізу можна виявити шаблони та тенденції у продуктивності, виявити часові втрати та ефективно впоратися з перешкодами[1].

У цьому контексті розробка програмного застосунку для моніторингу та аналізу активності користувача має велике значення. Цей застосунок може надати користувачам інструмент для систематичного відстеження їхньої продуктивності, а також допомогти виявити потенційні області для поліпшення. Він дозволить аналізувати час, проведений за комп'ютером, ідентифікувати найбільш витратні за ресурсами завдання та програми, і надати інформацію, необхідну для прийняття обґрунтованих рішень щодо управління часом та ресурсами. Такий застосунок стане незамінним помічником у підвищенні ефективності та досягненні поставлених цілей[2].

Тому тема роботи «Розробка програмного застосунку для моніторингу та аналізу активності користувача» є актуальною.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Мета програмного застосунку для моніторингу та аналізу активності користувача полягає в покращенні

продуктивності та ефективності роботи користувача за комп'ютером шляхом моніторингу та аналізу його активності.

Відповідно до мети роботи необхідно виконати наступні завдання:

– визначити вимоги до програмного застосунку;

– спроєктувати базу даних, сумісну з вимогами програмного застосунку;

– розробити модуль для управління та комунікації з базою даних;

– розробити модуль для відстеження та збереження використання програм користувачем;

– розробити модуль графічного інтерфейсу, який може формувати представляти історію використаних програм та формувати графіки;

– провести тестування програмного застосунку.

Об’єкт дослідження – процес розробки програмного застосунку для моніторингу та аналізу активності користувача.

Предмет дослідження – методи і засоби розробки програмного застосунку для моніторингу та аналізу активності користувача.

Методи дослідження. Під час дослідження було використано методи розробки бази даних; методи створення клієнтського інтерфейсу; методи семантичної розмітки; методи побудови блок-схем алгоритмів, методи реалізації програмних застосунків мовою Java з використанням Spring Framework.

Новизна отриманих результатів.

1. Запропоновано модуль реєстрації дій користувача, який, на відміну від існуючих реалізацій, використовує інструменти Java та Spring Framework, що дозволяє автоматично вибирати його імплементацію в залежності від операційної системи. Це значно спрощує процес використання застосунку для кінцевого користувача, забезпечуючи високу сумісність та зручність у експлуатації.

2. Удосконалено алгоритм збереження зібраних даних до бази даних, у якому застосовано метод буферизації даних. Це дозволяє зменшити

навантаження на базу даних, тим самим мінімізуючи вплив застосунку на роботу інших програм. Такий підхід сприяє підвищенню загальної продуктивності системи та забезпечує стабільну роботу.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в тому, що розроблена програма може використовуватись компаніями для моніторингу, аналізу та підвищення продуктивності роботи працівників за комп’ютером.

Особистий внесок здобувача. Отримані результати дослідження та розробки представлені автором особисто. Зокрема, у тезах доповіді автор представляє аналіз відомих рішень.

Апробація результатів здійснена на міжнародній науково-практичній інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ. МН-2024», (Вінниця, 2023)

Публікації. Тези доповідей представлені в збірнику міжнародної науково-практичної інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ. МН-2024», (Вінниця, 2023)

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану програмних застосунків для моніторингу та аналізу активності користувача

Сьогодні комп'ютер став невід'ємною частиною нашого повсякденного життя, та основним інструментом в багатьох сферах діяльності. Він використовується для ведення документації, комунікації, розваг, роботи та навчання. Однак, через розмаїтість завдань, які можна виконати за допомогою комп'ютера, часто важко визначити, що саме займає багато часу та які аспекти робочих процесів можна оптимізувати для досягнення більшої продуктивності[3]. Співвідношення між різними видами діяльності та їх впливом на ефективність роботи може бути неочевидним, що ускладнює процес аналізу та виявлення оптимальних стратегій роботи.

Програмні застосунки для моніторингу та аналізу активності користувача можуть вирішити цю проблему, надаючи користувачам інструменти для систематичного відстеження їхньої робочої активності та аналізу цих даних[4]. Вони функціонують шляхом запису та категоризації часу, проведеного на різних видах діяльності, таких як використання різних програм чи веб-сайтів. Ці застосунки допомагають користувачам зрозуміти, які конкретні завдання займають більше часу та впливають на їхню продуктивність.

Зокрема, ці програмні засоби можуть автоматично відстежувати час, проведений на кожному веб-сайті або в кожній програмі, та надавати докладні звіти про ці дані. Наприклад, користувач може дізнатися, скільки часу він витрачає на соціальні мережі, електронну пошту, або інші незначні завдання, порівняно з часом, витраченим на виконання основних робочих обов'язків. Це дозволяє визначити, де саме витрачається час неефективно, і де є можливість для поліпшення.

Крім того, ці застосунки можуть пропонувати функціонал для встановлення цілей та обмежень. Наприклад, користувач може встановити

обмеження на час, проведений на певних сайтах, або поставити цілі щодо збільшення часу, проведеного на продуктивних завданнях. Деякі застосунки навіть включають функції для блокування відволікаючих сайтів після досягнення встановленого ліміту.

Завдяки інструментам для аналізу даних, користувачі можуть отримувати візуалізації своєї активності у вигляді графіків та діаграм, що допомагає більш наочно зрозуміти свої звички та тенденції у використанні часу. Це дозволяє здійснювати інформовані корективи та покращувати власну продуктивність.

Аналітичні звіти та графіки, що створюються застосунками, надають візуальне представлення робочих звичок, допомагаючи виявити тенденції та визначити області для покращення. Візуалізація даних у формі графіків, діаграм та звітів робить інформацію більш доступною та легкою для розуміння, що сприяє глибшому аналізу робочої активності[5].

Завдяки цим інструментам користувачі можуть детально бачити, як розподіляється їхній час між різними завданнями та проектами. Наприклад, вони можуть аналізувати, який відсоток робочого часу витрачається на ключові проекти, а який – на другорядні завдання або відволікання. Це дозволяє виявити неефективні робочі практики та знайти шляхи їхнього усунення.

Візуальні звіти допомагають у визначенні пікових періодів продуктивності протягом дня чи тижня. Користувачі можуть побачити, коли вони найбільш продуктивні, і відповідно планувати виконання найважливіших завдань у ці періоди. Це сприяє більш ефективному розподілу робочого навантаження та підвищенню загальної продуктивності.

Отже, розробка програмного застосунку для моніторингу та аналізу активності користувача є актуальною і важливою через постійне зростання потреби в оптимізації робочих процесів та підвищенні продуктивності. З його використанням, компанії та звичайні користувачі матимуть більше інструментів для аналізу своєї роботи. Такі застосунки стають необхідними помічниками в сучасному світі, де ефективне управління часом є ключовим фактором успіху.

1.2 Порівняльний аналіз аналогів

В умовах, коли все більше людей працює з віддалених місць і використовує комп'ютер для виконання робочих завдань, важливо мати інструменти, які дозволяють ефективно керувати часом та аналізувати продуктивність. Дистанційна робота може створювати унікальні виклики для управління часом і зосередження, і програмні застосунки для моніторингу та аналізу активності користувача стають незамінними інструментами у забезпеченні ефективності та успішності в цьому новому режимі роботи. Однією з основних проблем, з якою стикаються дистанційні працівники, є труднощі з підтримкою чіткої межі між роботою та особистим життям. Відсутність фізичного розмежування може призвести до зниження продуктивності, через постійні відволікання[6]. Програмні засоби для моніторингу дозволяють працівникам самостійно відстежувати свою продуктивність та вчасно коригувати свою роботу. До найбільш відомих рішень відносять:

- ManicTime;
- Toggl Track;
- RescueTime;
- Clockify.

Одним із прикладів програмного застосунку для моніторингу та аналізу активності користувача є ManicTime (рис. 1.1). Ця програма призначена для відстеження часу, дозволяючи користувачам контролювати свою продуктивність шляхом реєстрації часу, витраченого на різні завдання та діяльності на комп'ютері[7]. Переваги використання ManicTime:

- Автоматичне відстеження часу;
- Детальні звіти та аналітика;
- Інтеграція з іншими інструментами;
- Простота використання.

Недоліки використання ManicTime:

– Необхідність платної версії;

– Складність налаштування для новачків;

– Обмежена підтримка командної роботи;

– Відсутність веб-інтерфейсу.

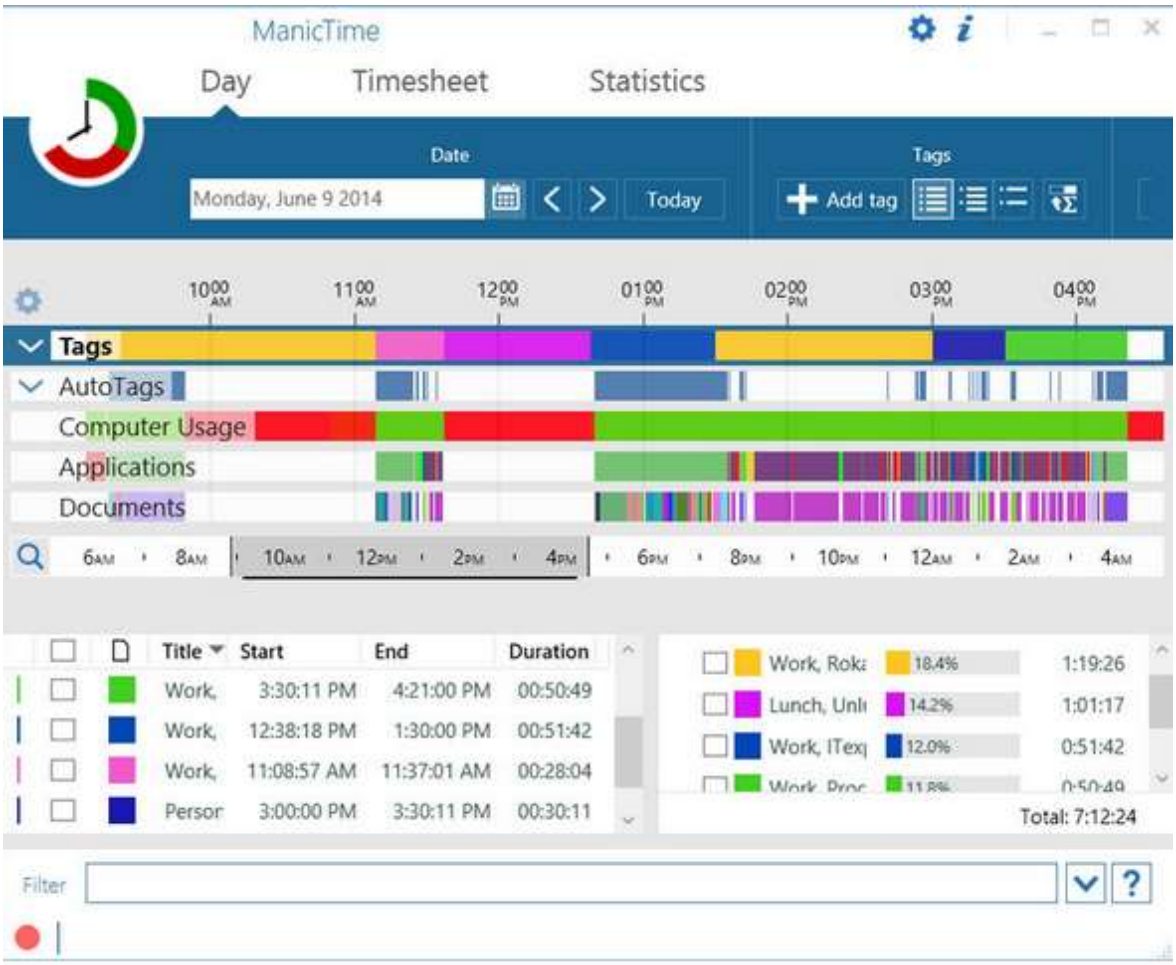


Рисунок 1.1 – Приклад головного вікна ManicTime

Інший приклад – застосунок Toggl Track (рис. 1.2), який призначений для відстеження та категоризації оплачуваних та неоплачуваних годин[8]. Цей тип застосунків зазвичай використовується на робочих місцях і відправляє інформацію менеджерам компанії для точної оцінки продуктивності працівників. Переваги використання Toggl Track:

Переваги використання Toggl Track:

– Простота використання;

– Кросплатформенність;
– Інтеграція з іншими інструментами;
– Безкоштовний план.

Недоліки використання Toggl Track:

– Вартість платних планів;
– Інтернет-залежність;
– Обмежені можливості офлайн-відстеження;
– Обмежені можливості для глибокої аналітики в безкоштовній версії.

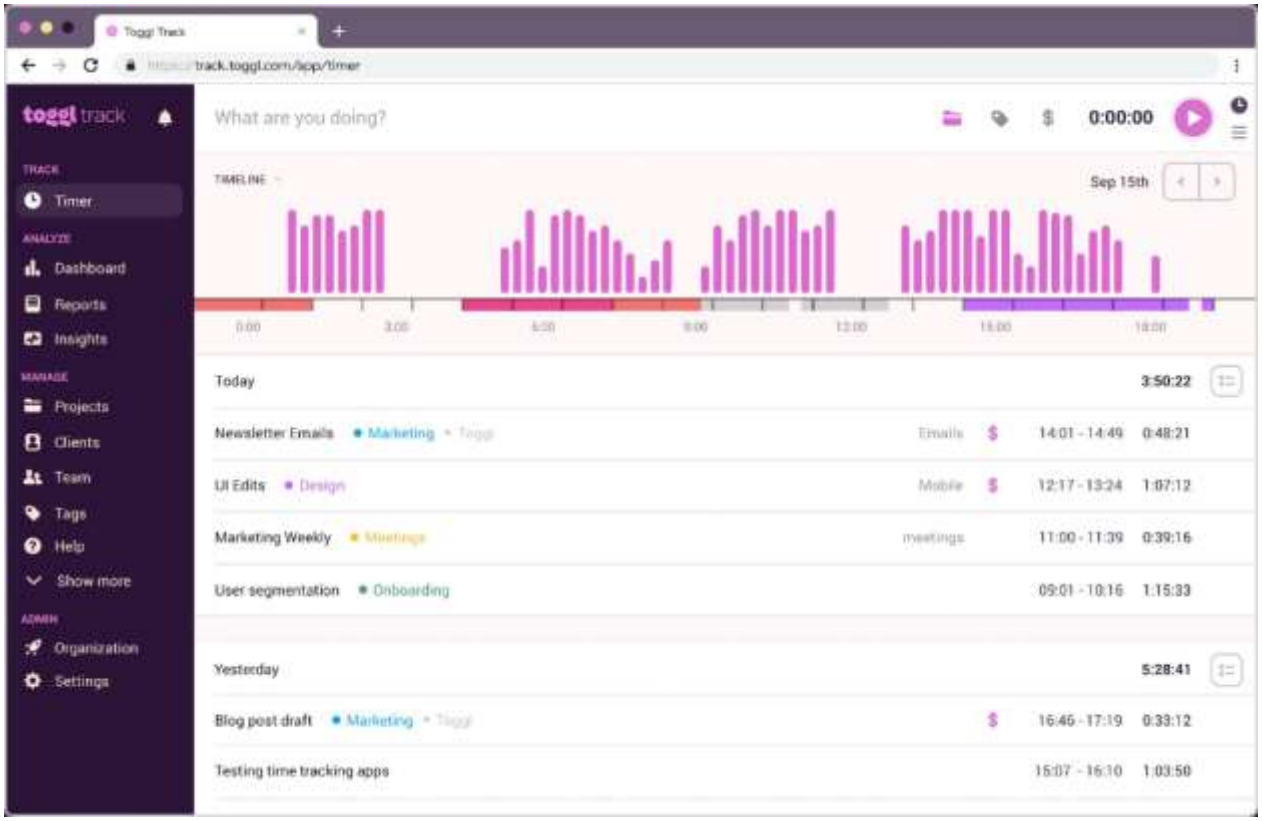


Рисунок 1.2 – Приклад інтерфейсу Toggle Track

Ще одним прикладом програмного забезпечення для моніторингу та аналізу активності користувача є RescueTime (рис. 1.3). Ця програма автоматично відстежує активність на комп'ютері або мобільному пристрої, після чого надає звіти та статистику про витрачений час[9]. Оскільки RescueTime має веб-інтерфейс, деякі функції можуть бути недоступні без Інтернет-з'єднання, що може бути не зручно в певних ситуаціях. Також для

повного функціоналу RescueTime потрібна підписка.

Рисунок 1.3 – Приклад інтерфейсу RescueTime

Переваги використання RescueTime

– Автоматичне відстеження часу;

– Категоризація активностей;

– Підтримка багатьох платформ;

– Безкоштовна версія.

Недоліки використання RescueTime

– Обмежені функції в безкоштовній версії;

– Приватність даних;

– Обмежені можливості інтеграції;

– Потреба в постійному підключенні до інтернету.

Clockify (рис. 1.4) - це повністю безкоштовний застосунок, який дозволяє відстежувати час, проведений за комп'ютером[10]. З його допомогою ви зможете реєструвати час, витрачений на різні проекти та завдання, створювати звіти і аналізувати продуктивність. Важливо зауважити, що деякі функції доступні лише у платних версіях програми, що може обмежити можливості

безкоштовних користувачів. Однак і безкоштовна версія надає значний функціонал для відстеження та управління вашим часом.

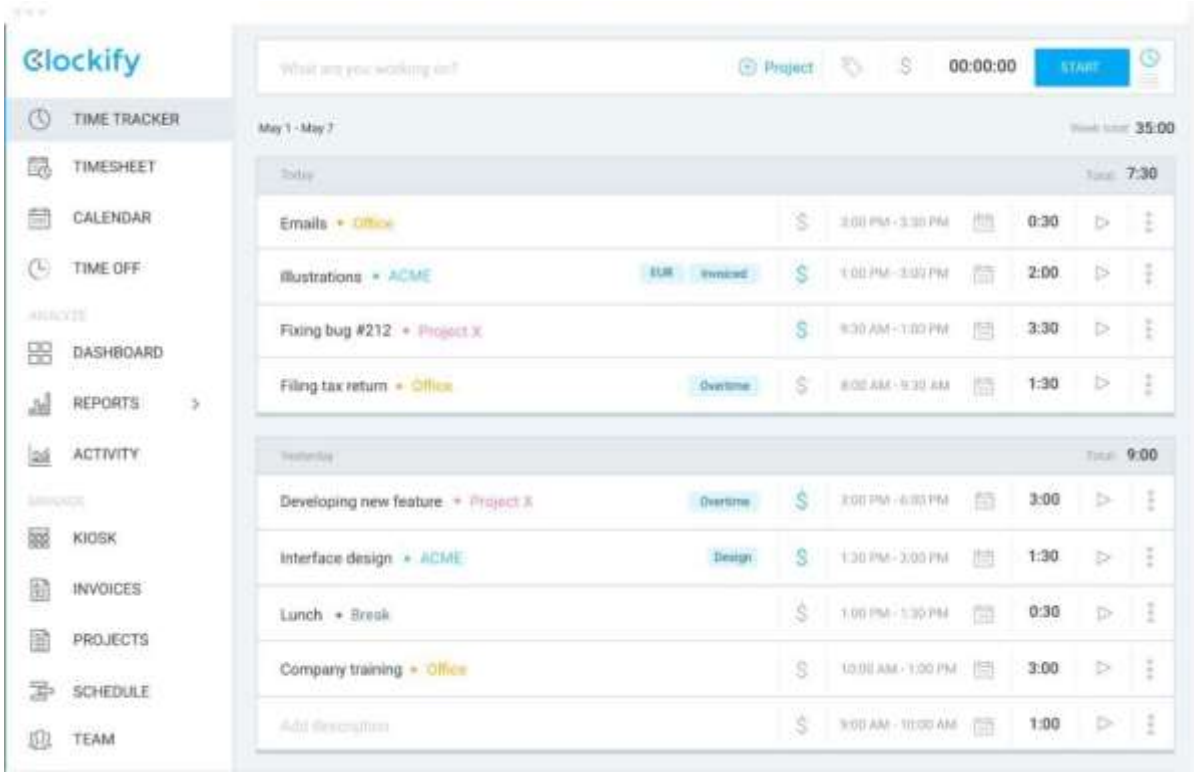


Рисунок 1.4 – Приклад інтерфейсу Clockify

Переваги використання Clockify

– Безкоштовний доступ;
– Інтуїтивний інтерфейс;
– Кросплатформенність;
– Детальні звіти та аналітика.

Недоліки використання Clockify

– Вартість платних планів;
– Складність налаштування для великих команд;
– Обмежені можливості офлайн-режиму.

Розробка програмного застосунку для моніторингу та аналізу активності користувача передбачає створення внутрішньої системи, яка може реєструвати активні вікна на комп'ютері та категоризувати їх за потребою. Інтерфейсна система передбачає створення користувацького інтерфейсу для зручного

подання інформації, зібраної системою. Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика

Характеристика	**ManicTime**	**Toggl Track**	**RescueTime**	**Clockify**	**UsageSense**
Український інтерфейс	+	-	-	-	+
Автоматичне відстеження	+	-	+	+	+
Детальні звіти і аналітика	+	+	+	+	+
Категоризація діяльності	+	+	-	+	+
Безкоштовний доступ	-	-	+	+	+
Відкритий код	-	-	-	-	+
Можливості спільної роботи	-	+	+	+	+
Загальна оцінка	66%	50%	66%	83%	100%

Відповідно до таблиці порівняльних характеристик, розробка програмного застосунку для моніторингу та аналізу активності користувача зможе покрити недоліки існуючих рішень та забезпечити новий досвід та можливості застосування технології доповненої реальності.

Отже, розробка програмного застосунку для моніторингу та аналізу активності користувача надасть більше доступних інструментів для управління власним часом, а також підвищить продуктивність роботи за комп'ютером. Цей застосунок зможе бути використаний як для особистого розвитку та самоконтролю, надаючи користувачам засоби для підвищення ефективності у будь-якій сфері їхньої діяльності, так і для професіоналів у сферах бізнесу та

управління, де продуктивність є критичним фактором для успіху.

1.3 Аналіз методів розв’язання задачі

Розробка програмного застосунку для моніторингу та аналізу активності користувача вимагає ретельного розгляду підходів до вирішення проблеми. Існують різні методи, кожен з яких має свої переваги та недоліки. У цьому розділі проаналізовано найбільш ефективні методи розробки таких програмних застосунків.

Одним з варіантів вирішення проблеми розробки програмного застосунку для моніторингу та аналізу активності користувача є створення стандартного десктопного застосунку. Такий підхід має свої переваги та недоліки.

Переваги:

– Робота без доступу до Інтернету: Десктопний застосунок може функціонувати автономно, без необхідності підключення до Інтернету, що забезпечує безперебійну роботу навіть при відсутності мережевого з'єднання.

– Глибока інтеграція з операційною системою: Десктопний застосунок має можливість глибокої інтеграції з операційною системою, що дозволяє точно і автоматично реєструвати активність користувача на комп’ютері.

Недоліки:

– Встановлення та оновлення: Необхідність встановлення та регулярного оновлення програмного забезпечення на кожному комп'ютері може бути складним завданням для системних адміністраторів, особливо в великих організаціях.

– Аналіз даних з декількох комп’ютерів: Складність збору та аналізу даних з декількох комп’ютерів одночасно може бути проблематичною, оскільки для цього потрібні додаткові зусилля для забезпечення синхронізації та обробки даних.

Інший підхід полягає в розробці веб-сервісу. У цьому варіанті програмний застосунок розробляється як веб-сервіс, який доступний через браузер.

Переваги:

– Доступ з будь-якого пристрою: Веб-сервіс дозволяє користувачам отримувати доступ до функціоналу застосунку з будь-якого пристрою, що підключений до Інтернету, що забезпечує високу мобільність і зручність.

– Легке масштабування: Веб-сервіс легко масштабується для великої кількості користувачів, оскільки обробка даних відбувається на сервері.

– Централізоване оновлення: Оновлення програмного забезпечення здійснюються централізовано на сервері, що значно зменшує зусилля, необхідні для підтримки і оновлення застосунку.

Недоліки:

– Залежність від Інтернету: Веб-сервіс залежить від доступу до Інтернету, тому він може бути недоступний у випадку відсутності мережевого з'єднання.

– Обмежений доступ до операційної системи: Застосунок, який працює через браузер, має обмежений доступ до функціоналу операційної системи, що може вимагати від користувача ручного введення деяких даних про активність.

Ще одним підходом до розробки програмного застосунку для моніторингу та аналізу активності користувача є комбінована архітектура, що поєднує в собі десктопний застосунок та веб-сервіс.

Переваги:

– Повний доступ до операційної системи: Десктопний застосунок має повний доступ до операційної системи, що дозволяє точно і автоматично реєструвати активність користувача на комп’ютері.

– Синхронізація та аналіз даних: Десктопний застосунок може працювати самостійно або бути з’єднаним з сервером для синхронізації та аналізу даних з декількох комп’ютерів, що забезпечує більш глибоку аналітику та зручність в управлінні.

– Гнучкість та мобільність: Користувачі можуть отримувати доступ до своїх даних через веб-сервіс з будь-якого пристрою, що підключений до Інтернету, що забезпечує високу гнучкість та мобільність.

Недоліки:

– Складність розробки: Комбінована архітектура вимагає розробки і підтримки більшої кількості коду, що може збільшити складність і вартість проекту.

– Підтримка та оновлення: Потрібно забезпечувати підтримку та оновлення як десктопного застосунку, так і веб-сервісу, що може потребувати додаткових ресурсів.

Розглянувши переваги та недоліки кожного методу, було вирішено використати метод розробки комбінованої архітектури, що поєднує в собі десктопний застосунок та веб-сервіс для реалізації програмного застосунку для моніторингу та аналізу активності користувача. Такий підхід дозволить отримати повноцінний програмний продукт що буде доступним для якнайбільшої кількості користувачів не жертвуючи важливим функціоналом.

1.4 Висновки

У першому розділі було розглянуто стан програмних застосунків для моніторингу та аналізу активності користувача. Було проведено порівняння відомих рішень для моніторингу та аналізу активності користувача, таких як: ManicTime, Toggl Track, RescueTime та Clockify.

У результаті порівняння було виявлено, що досліджувані рішення набули значної популярності завдяки своєму функціоналу та підтримці. Проаналізувавши переваги та недоліки відомих платформ, було виявлено, що всі рішення надають можливість автоматичного відстеження дій та спільної роботи. Також більшість щ них є пропрієтарними і часто платними. Таким чином було доведено доцільність розробки власного програмного застосунку. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власного програмного застосунку.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Аналіз вхідних даних є ключовим етапом у розробці програмного застосунку для моніторингу та аналізу активності користувача. Цей процес передбачає не лише огляд інформації, яку застосунок буде збирати, але й глибоке розуміння різних типів цих даних та їх значення для подальшого використання. Спершу необхідно визначити, які саме дані будуть збиратися, включаючи активність користувача на різних веб-сайтах, у додатках, час, проведений на виконанні певних завдань, та інші аспекти, що відображають робочі звички.

Ключовим моментом є розуміння значення кожного типу даних у контексті продуктивності та ефективності користувача. Наприклад, час, проведений на певних веб-сайтах, може свідчити як про корисну діяльність, пов'язану з роботою, так і про відволікання, що знижують продуктивність. Тому важливо не лише збирати ці дані, але й уміти їх правильно інтерпретувати. Це вимагає розробки алгоритмів та моделей, які зможуть автоматично категоризувати активність на основі її продуктивності.

Особиста інформація користувача є одним з перших типів вхідних даних, які варто ретельно розглянути. Персональні дані, такі як ім'я, контактні дані та адреса проживання, не лише допомагають ідентифікувати користувача, але й створюють основу для персоналізованих послуг та взаємодії з ним.

Основним типом вхідних даних для програмного застосунку є інформація про використання комп'ютера користувачем. Ці дані є важливим джерелом для розуміння активності користувача та можуть містити широкий спектр інформації, включаючи:

1. Активність у програмах: Інформація про те, які програми використовувалися користувачем і як довго вони були відкриті, дозволяє отримати уявлення про його робочі звички та пріоритети.

2. Час, проведений за робочим столом: Інформація про те, скільки часу користувач проводить перед комп'ютером, може бути корисною для управління часом та ефективності роботи.

Крім того, конфіденційність та захищеність цих даних є надзвичайно важливими, оскільки вони можуть містити особисту інформацію користувача, таку як логіни, паролі або особисті дані. Забезпечення високого рівня безпеки цих даних передбачає захист від несанкціонованого доступу та зловживання. Тому важливо вживати ефективні заходи забезпечення безпеки, такі як шифрування та автентифікація, для захисту інформації користувача.

Додатковою важливою складовою є інформація про застосунки, встановлені на комп’ютері користувача. Ці дані дозволяють зрозуміти, які програми користувач використовує найчастіше та як вони впливають на його робочий процес. Ретельний аналіз цих даних допомагає розробникам створювати більш ефективні та зручні застосунки для кінцевих користувачів.

Узагальнюючи, важливість аналізу вхідних даних полягає в зрозумінні потреб користувачів і покращенні функціоналу програмного застосунку. Цей процес допомагає оптимізувати використання ресурсів та підвищує якість вихідних графіків, що впливає на загальну ефективність роботи програми. Проаналізувавши ці дані, розробники можуть знайти шляхи для поліпшення функціоналу та вдосконалення користувацького досвіду.

2.2 Розробка структури інтерфейсу програмного застосунку

Етап розробки структури інтерфейсу програмного застосунка є критично важливим для успіху продукту, оскільки від нього залежить сприйняття користувачами та ефективність використання програми. Цей етап включає кілька ключових підпроцесів, які спрямовані на створення інтерфейсу, що відповідає потребам користувачів і допомагає досягти цілей застосунка. Перший крок у розробці структури інтерфейсу – це ретельний аналіз потреб та очікувань користувачів. Це включає визначення основних елементів інтерфейсу, їх розташування та способи навігації. На основі аналізу потреб

користувачів розробляється структура та організація інформації в межах програмного застосунку. На основі аналізу потреб користувачів розробляється структура та організація інформації в межах програмного застосунка. Це включає визначення основних елементів інтерфейсу, їх розташування та способи навігації[11].

Основні принципи дизайну інтерфейсу користувача допомагають створювати зручні, інтуїтивні та ефективні продукти, які покращують користувацький досвід. Ключові принципи дизайну інтерфейсу включають:

1. Простота. Інтерфейс повинен бути чистим і мінімалістичним, щоб користувачі могли швидко знайти необхідну інформацію та функції.

2. Консистентність забезпечує однаковість у використанні елементів інтерфейсу на всіх сторінках та екранах. Це стосується шрифтів, кольорів, стилів кнопок та інших візуальних елементів. Консистентність допомагає користувачам легше орієнтуватися в інтерфейсі, знижуючи необхідність навчання.

3. Запобігання помилкам. Інтерфейс повинен бути спроєктований таким чином, щоб зменшити ймовірність виникнення помилок.

4. Гнучкість та ефективність використання. Інтерфейс повинен бути гнучким і ефективним для використання як початківцями, так і досвідченими користувачами.

5. Естетика та мінімалізм. Естетично привабливий інтерфейс покращує загальне враження від використання програми. Мінімалістичний дизайн, що не перевантажений зайвими елементами, допомагає користувачам зосередитися на важливих завданнях.

З урахуванням вищенаведених принципів було створено структуру інтерфейсу застосунка, яка продемонстрована на рисунку 2.1.

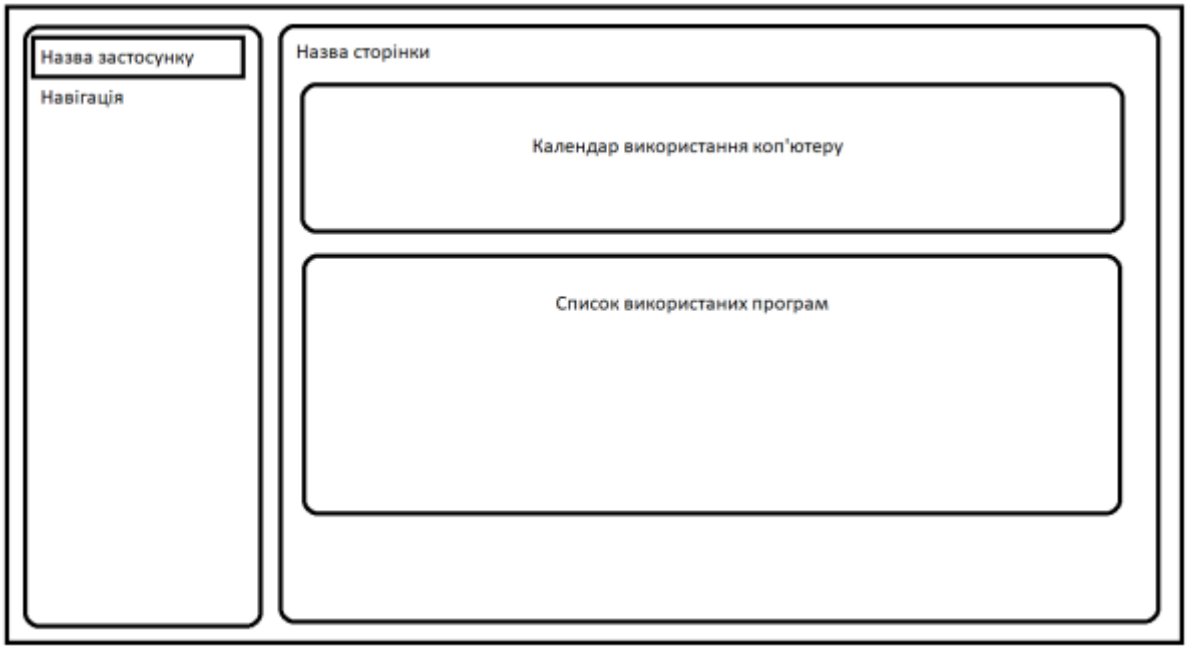


Рисунок 2.1 – Структура інтерфейсу головної сторінки

Така структура інтерфейсу, яка містить головний контент на головній сторінці без зайвого завантаження додаткового контенту, є дуже корисною та ефективною. Вона сприяє зручності користування і покращує загальний досвід користувача. Користувач може швидко отримати необхідну інформацію без необхідності перегортання багатьох сторінок або зайвого клікання. Зосередженняся на основній інформації дозволяє більш детально працювати над дизайном та взаємодією цих елементів, що може призвести до більш привабливого та ефективного інтерфейсу.

2.3 Розробка моделі системи

Для кращого розуміння роботи програмного застосунку для моніторингу та аналізу активності користувача було розроблено модель роботи системи на прикладі UML діаграми діяльності. UML-діаграма діяльності дозволяє візуалізувати послідовність дій або процесів у системі. На діаграмі можуть бути представлені кроки, прийоми рішень, умови та інші елементи, які допомагають зрозуміти логіку роботи програми. Ця модель допомагає краще зрозуміти, як взаємодіє система та яким чином вона виконує свої функції[12]. UML-діаграма

додатку для моніторингу та аналізу активності користувача продемонстрована на рисунку 2.2.

Рисунок 2.2 – Діаграма діяльності системи

Згідно з цією діаграмою, після запуску застосунку першим запускається алгоритм реєстрації активності користувача за комп’ютером. Наступним кроком користувач може відкрити інтерфейс застосунка. Відкривши інтерфейс застосунка, користувач має вибір дій. Користувач може переглянути загальний звіт за весь рік. Також є можливість переглянути детальну статистику за один

день. Ще однією опцією є перегляд детальної статистики щодо конкретного застосунка.

2.4 Розробка алгоритмів роботи системи

Для розробки програмного застосунку для моніторингу та аналізу активності користувача необхідно розробити багато алгоритмів, згідно з якими працюватиме цей застосунок. Для розробки цих алгоритмів були створені блок-схеми, які надають можливість візуального представлення послідовності операцій та умов, що виконуються в процесі виконання програми[13].

Одним з найголовніших алгоритмів програмного застосунку для моніторингу та аналізу активності користувача є алгоритм збереження даних про активне вікно. Блок-схема цього алгоритму представлена на рисунку 2.3. Алгоритм розпочинається на етапі 1 із запуском процесу збору даних. Далі в циклічному режимі перевіряється активність застосунку. Поки застосунок активний, алгоритм переходить до етапу 3, де зчитуються дані про активне вікно.

На етапі 4 відбувається зчитування даних про процес, асоційований з активним вікном. Після успішного зчитування даних алгоритм перевіряє, чи є активне вікно новим. У разі, якщо активне вікно не є новим, дані записуються до буфера. Якщо буфер заповнений, дані з нього зберігаються до бази даних, і процес продовжується зі зчитування нових даних. У разі виявлення нового вікна створюється новий запис та перевіряється існування програми у базі даних. Якщо програма вже існує в базі даних, алгоритм зберігає дані з буфера до бази даних та записує нові дані до буфера.

Якщо ж програма не існує в базі даних, відбувається реєстрація нової програми в базі даних, після чого зберігаються дані з буфера, і записуються нові дані до буфера. Процес завершується на етапі 15.

Цей алгоритм забезпечує ефективне збирання та збереження даних про активність користувача, що знижує навантаження на базу даних за рахунок буферизації даних. Використання буфера дозволяє зберігати дані пакетами, що

мінімізує кількість операцій запису до бази даних та зменшує вплив застосунку на роботу системи в цілому.

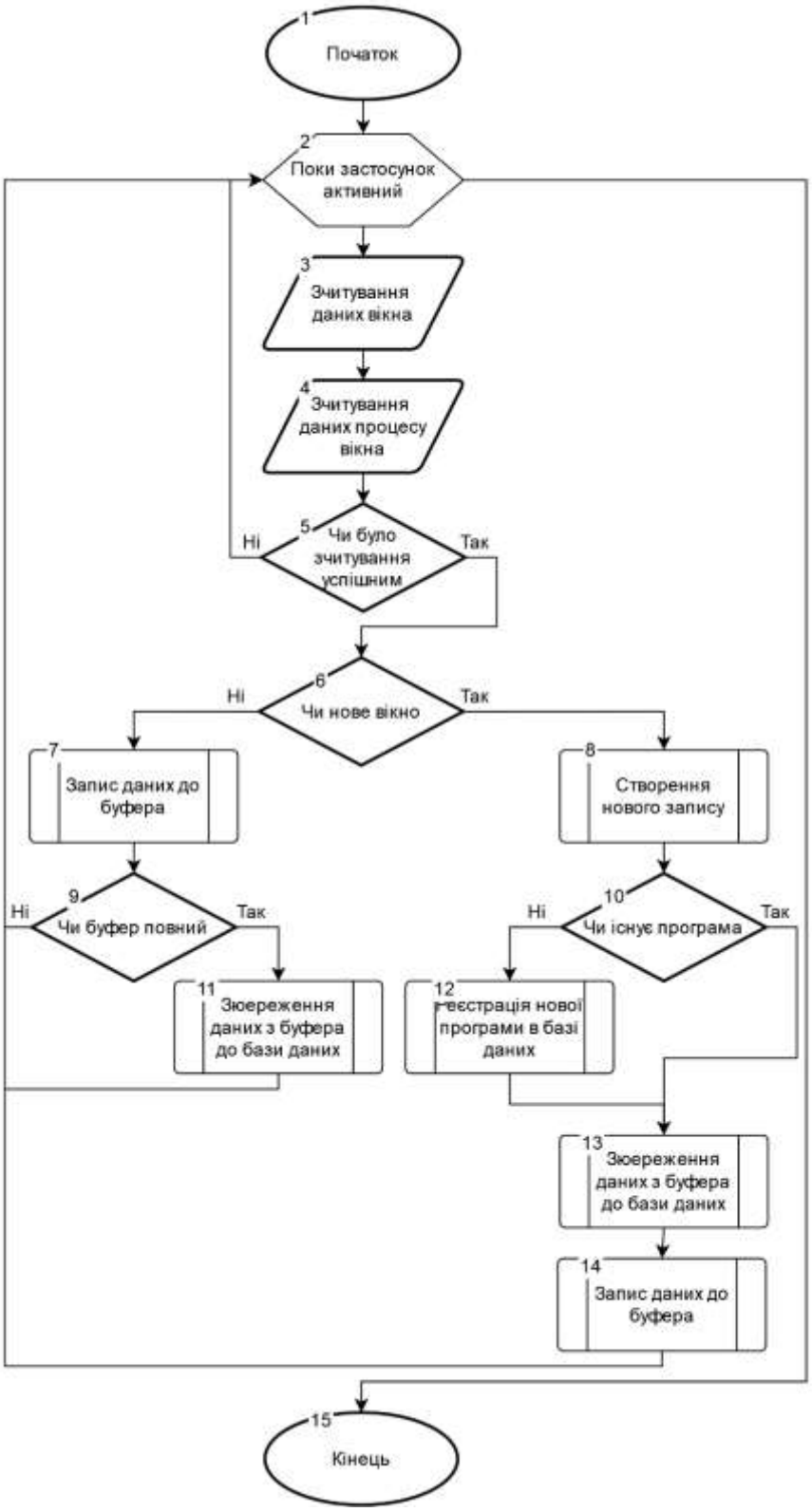


Рисунок 2.3 – Блок-схема роботи алгоритму збережнння даних про активне вікно

Ще одним алгоритмом програмного застосунку для моніторингу та аналізу активності користувача є алгоритм відображення таблиці з інформацією про кількість годин, проведених за комп’ютером кожного дня. Блок-схема цього алгоритму представлена на рисунку 2.4.

Рисунок 2.4 – Блок-схема алгоритму створення таблиці головної сторінки

Алгоритм починається, коли завантажується головна сторінка програмного застосунку. Він отримує дані про активність користувача з бази даних. Ці дані містять інформацію про те, скільки годин було проведено за комп’ютером кожного дня. Алгоритм створює в таблиці відступ у першому тижні року залежно від того, з якого дня починається рік. Алгоритм виконується в циклі, поки не обробить кожного дня року. Алгоритм створює клітинку таблиці для кожного дня. Перевіряє, чи містить клітинка таблиці дані, і якщо клітинка містить дані, створюються підказки при наведенні. Підказка містить інформацію про дату та кількість годин, проведених користувачем за комп'ютером. Посилання веде до детальної інформації про активність користувача за цей день. Якщо клітинка не містить даних, алгоритм переходить до наступного кроку. Алгоритм перевіряє, чи є поточна дата кінцем місяця. Якщо так, алгоритм створює в таблиці відступ для заголовку наступного місяця. Якщо ні, алгоритм переходить до наступного кроку. Потім алгоритм додає стилі до клітинок таблиці.

Перед завершенням алгоритм відображає таблицю з інформацією про кількість годин, проведених користувачем за комп'ютером кожного дня. Розглянувши вищенаведені алгоритми, можна зрозуміти, що розроблений програмний застосунок матиме надійні алгоритми, які дозволять надійно реєструвати дії користувача та зрозуміло подавати їх на графічному інтерфейсі.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмного застосунку. Також було розроблено основний алгоритм реєстрації дій користувача та алгоритм створення таблиці головної сторінки. Було створено модель роботи програмного продукту за допомогою UML діаграм.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ЗАСТОСУНКУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Для розробки цього програмного застосунку для моніторингу та аналізу активності користувача було необхідно вибрати сучасні та стабільні інструменти розробки. Тому було обрано Java, Spring, PostgreSQL та JNA з метою забезпечення надійності, ефективності та розширюваності. Java надає платформонезалежність, Spring дозволяє швидко розробляти та реалізовувати бекенд-функціонал з високою ефективністю, PostgreSQL відомий своєю стабільністю та можливостями розширення, а JNA дозволяє взаємодіяти з нативними бібліотеками для отримання доступу до системних ресурсів.

Java є мовою програмування загального призначення, яка відома своєю переносимістю та надійністю. Вона має велику екосистему бібліотек та інструментів розробки, що полегшують створення складних програмних застосунків. Крім того, Java підтримує об'єктно-орієнтований підхід до програмування, що дозволяє створювати модульний та легко змінюваний код.

Spring Framework є одним з найпопулярніших фреймворків для розробки веб-застосунків на Java[14]. Він надає потужні інструменти для реалізації бекенду застосунку, включаючи створення контролерів, сервісів та доступу до бази даних. Spring також сприяє впровадженню кращих практик у програмуванні, таких як інверсія контролю та впровадження залежностей.

PostgreSQL є потужною та надійною реляційною базою даних з відкритим вихідним кодом. Вона підтримує багато функцій SQL та має велику спільноту користувачів, що робить її привабливим вибором для проектів, які потребують ефективного та масштабованого зберігання даних[15].

JNA (Java Native Access) – це бібліотека для Java, яка дозволяє взаємодіяти з бібліотеками, написаними на мовах програмування рівня машинного коду, таких як C та C++. Це дозволяє програмному застосунку використовувати оптимізований нативний код для виконання певних завдань,

які можуть бути недосяжними або менш ефективними у Java[16].

Для розробки графічного інтерфейсу системи було обрано фреймворк Tailwind CSS. Tailwind CSS дозволяє розробникам використовувати класи безпосередньо в HTML для швидкого та ефективного стилізування елементів. Це значно прискорює процес розробки, оскільки не потрібно писати кастомні CSS правила. Крім того, Tailwind CSS забезпечує високу гнучкість та консистентність дизайну завдяки використанню конфігурованих тем, що дозволяє легко адаптувати стиль до потреб конкретного проекту. Бібліотека включає широкий набір корисних утиліт, які охоплюють майже всі аспекти веб-дизайну, від кольорів і типографіки до розміщення елементів та адаптивності, що робить її потужним інструментом для створення сучасних інтерфейсів.

Отже, вибір цих технологій є оптимальним для реалізації функціональності програмного застосунку та забезпечить його успішну розробку, впровадження та подальшу підтримку. Java надає переносимість та безпеку, Spring допомагає побудувати ефективний бекенд та забезпечує високий рівень модульності, PostgreSQL забезпечує надійне та масштабоване зберігання даних, а JNA дозволяє інтегрувати нативний код для взаємодії з системними ресурсами. У результаті, користувачі отримають зручний та потужний інструмент для відстеження та аналізу своєї активності на комп'ютері.

3.2 Розробка інтерфейсу програмного застосунку

Розробка інтерфейсу є дуже важливою частиною створення програмного застосунку для моніторингу та аналізу активності користувача, оскільки вона визначає спосіб, у який користувачі будуть взаємодіяти з програмним застосунком. Гарно спроектований та інтуїтивно зрозумілий інтерфейс допомагає забезпечити приємний та ефективний досвід користувача, підвищуючи ймовірність того, що користувачі будуть продовжувати використовувати застосунок[17].

Вибір палітри кольорів інтерфейсу застосунка є важливим аспектом

дизайну, який впливає на сприйняття користувачем, естетику і функціональність програмного забезпечення. Правильна палітра кольорів може значно покращити користувацький досвід, тоді як невдалий вибір може призвести до негативних вражень. Кольори відіграють ключову роль у створенні першого враження від застосунку. Палітра кольорів також допомагає підсилити ідентичність бренду, створюючи єдиний візуальний стиль, який відрізняє ваш застосунок від конкурентів.

Основними кольорами інтерфейсу обрано світло-сірий та білий, оскільки ці кольори є нейтральними і створюють чистий, сучасний та професійний вигляд. Світло-сірий та білий відтінки дозволяють фокусувати увагу користувачів на важливих елементах та інформації, не відволікаючи їхньої уваги.

Допоміжними кольорами обрано блакитний та бірюзовий, оскільки вони вдало підкреслюють необхідні деталі й додають інтерфейсу жвавості та свіжості. Бірюзовий колір, що поєднує властивості синього і зеленого, приносить відчуття енергії та свіжості, сприяє залученню уваги користувачів і полегшує орієнтацію в інтерфейсі. Вибрана кольорова палітра продемонстрована на рисунку 3.1.

Рисунок 3.1 – Кольорова палітра

При відкритті інтерфейсу застосунку, користувач бачить головну сторінку із загальним оглядом активності за весь рік. Головна сторінка є центральним елементом, що надає швидкий доступ до ключової інформації та аналітики. Це перше враження користувача від застосунку, тому вона повинна бути інтуїтивною та інформативною.

На головній сторінці продемонстрована таблиця з інформацією про кількість годин, проведених за комп’ютером кожного дня протягом року. Ця таблиця дозволяє користувачам швидко оцінити свою активність і виявити тенденції у використанні часу. Вона структурована так, щоб легко читалася і надавала чітке уявлення про робочий графік користувача. Ця сторінка продемонстрована на рисунку 3.2

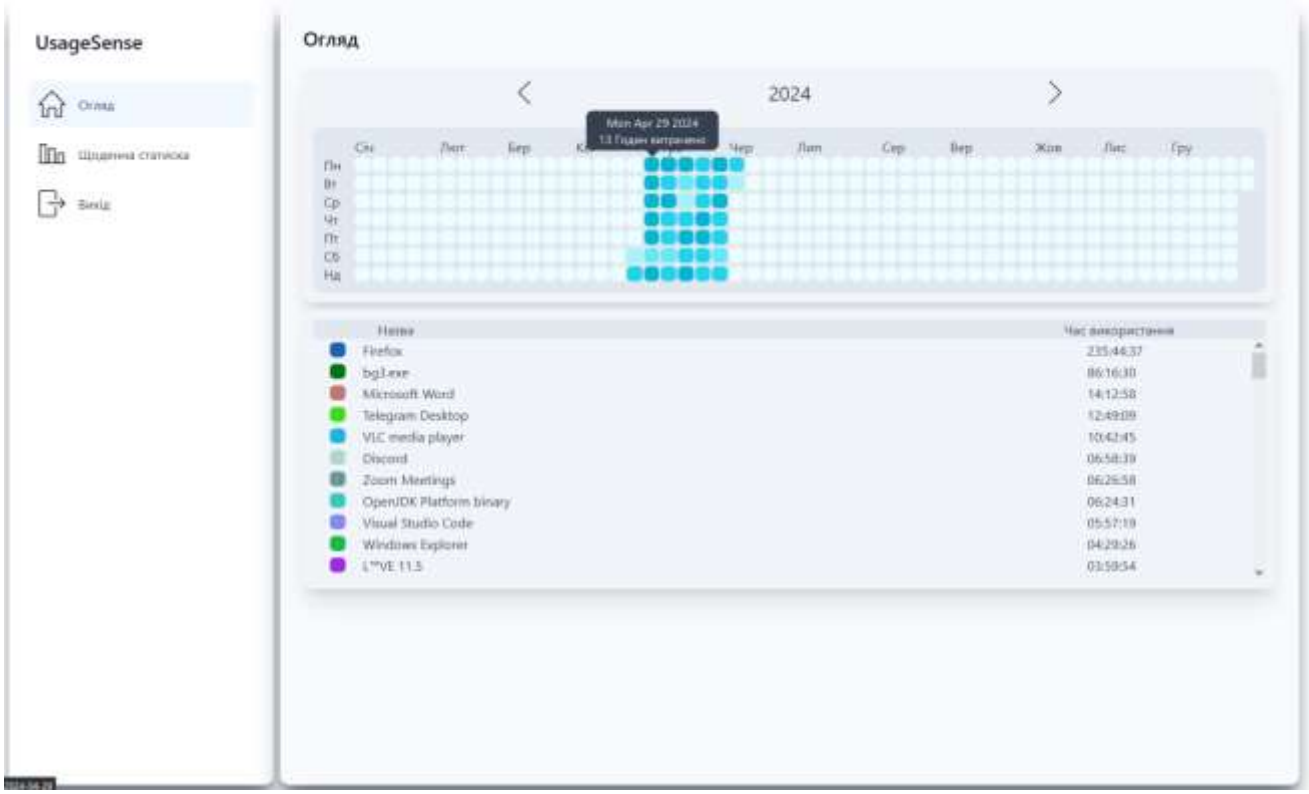


Рисунок 3.2 – Головна сторінка застосунку

При відкритті інтерфейсу застосунку, користувач бачить головну сторінку з загальним оглядом активності за весь рік. Головна сторінка є центральним елементом, що надає швидкий доступ до ключової інформації й

аналітики. Це перше враження користувача від застосунку, тому вона повинна бути інтуїтивною та інформативною.

На головній сторінці продемонстрована таблиця з інформацією про кількість годин, проведених за комп’ютером кожного дня протягом року. Ця таблиця дозволяє користувачам швидко оцінити свою активність і виявити тенденції у використанні часу. Вона структурована так, щоб легко читалася та надавала чітке уявлення про робочий графік користувача.

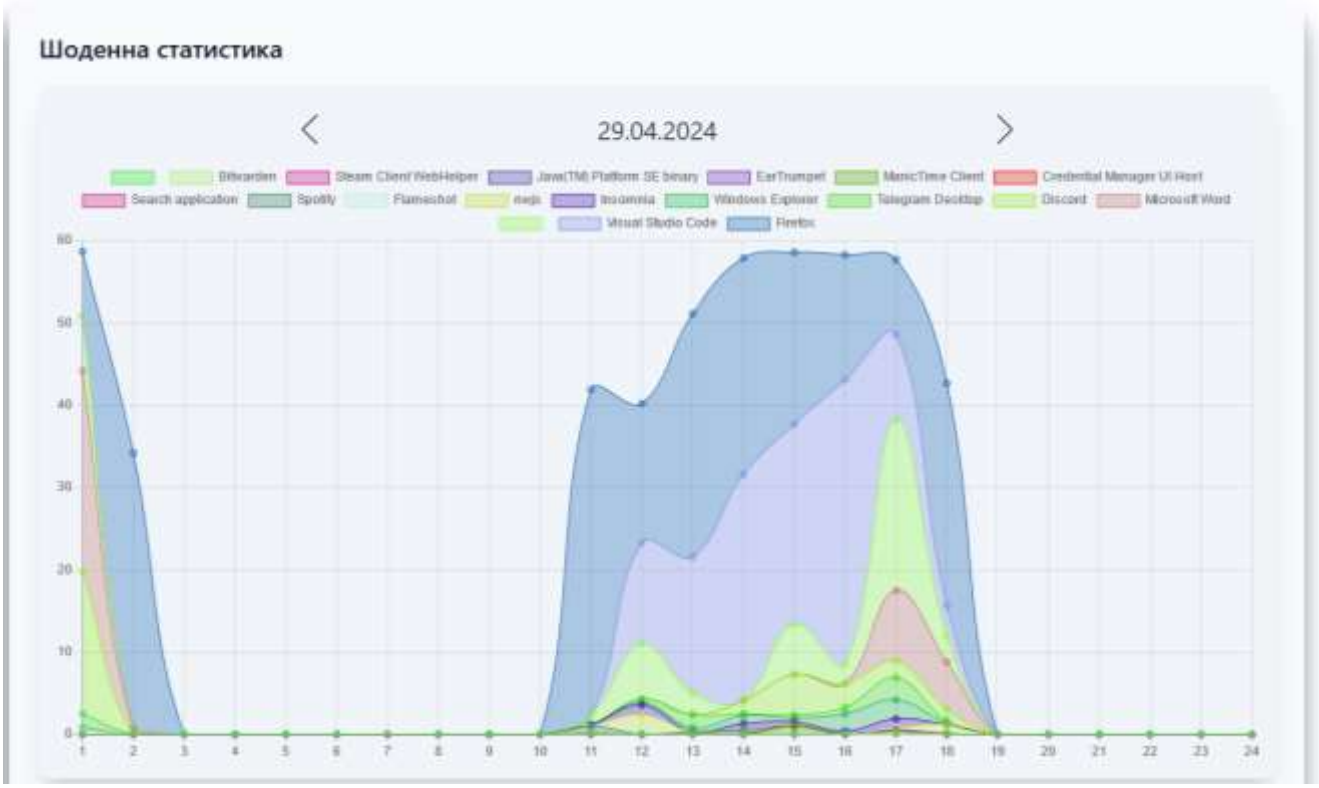


Рисунок 3.3 – Сторінка детальної інформації про конкретний день

Угорі сторінки можна побачити день, за який відображається статистика, а також кнопки для вибору іншої дати. На цій сторінці ви знайдете графічне зображення погодинного використання різних застосунків. Цей графік надає можливість оцінити, скільки часу було витрачено в кожному окремому застосунку протягом години, а також загальну кількість часу, проведений за комп'ютером. З його допомогою можна аналізувати не лише активність в окремих застосунках, а й узагальнену активність користувача за певний період.

Лінійний графік – найкращий вибір для візуалізації, коли ми хочемо подивитися, як використовується кожний застосунок з часом. Він допомагає порівняти застосунки між собою й виявити зміни у використанні протягом часу. Графік дозволяє легко спостерігати за трендами та зрозуміти дані без особливих знань.

Нижче на цій сторінці можна побачити список програм, що використовувалися протягом дня. Цей список є важливим елементом інтерфейсу, який забезпечує користувачів детальною інформацією про їх активність та використання різних застосунків. У цьому списку можна спостерігати за кольором відображення кожного застосунку, який є однаковим в межах всього додатку. Використання постійних кольорів для кожного застосунку надає можливість швидко й легко розрізняти їх, підвищуючи зручність та ефективність користування інтерфейсом. Список застосунків, використаних за день, продемонстровано на рисунку 3.4.

Рисунок 3.4 – Список використаних застосунків за конкретний день

Кольорові позначення не тільки роблять список більш візуально привабливим, але й допомагають користувачам швидше орієнтуватися серед великої кількості програм. Це знижує когнітивне навантаження, оскільки користувачі можуть миттєво впізнати програму за кольором, не читаючи її назву.

Також у списку можна побачити назву кожного застосунку та час,

проведений у ньому. Ці дані представлені у зручному форматі, що дозволяє користувачам швидко зрозуміти, які програми вони використовували найбільше, і як вони розподіляли свій час протягом дня. Інформація про час використання представлена як загальна сума годин та хвилин.

Натиснувши на одну з програм, користувач може відкрити детальнішу інформацію про цю програму. Ця функціональність дозволяє заглибитися у специфіку використання кожного окремого застосунку, надаючи більш детальний аналіз. Сторінка детального огляду програми представлена на рисунку 3.5 і містить різноманітну інформацію, що допомагає користувачу краще зрозуміти, як саме та коли використовувалася певна програма.

Рисунок 3.5 – Сторінка детальної інформації про програму

На цій сторінці наведена інформація про використання конкретного застосунку за весь час роботи застосунку UsageSense. У лівому блоці сторінки можна побачити стовпчасту діаграму, на якій відображено, скільки використовувалася програма в кожен з днів тижня. В заголовку діаграми можна побачити узагальнення, в якому вказано день тижня, в який програма

використовувалася найчастіше. У правому блоці сторінки знаходиться кругова діаграма, на якій відображено, скільки часу було проведено у вибраному застосунку за весь час у порівнянні з іншими програмами. Також на круговій діаграмі показано три найпопулярніші програми для порівняння та суму менш популярних застосунків. В заголовку діаграми знаходиться її узагальнення, де можна побачити, скільки відсотків часу було проведено у вибраному застосунку.

Розробка графічного інтерфейсу пройшла успішно завдяки використанню різноманітних інструментів та технологій. За допомогою мов програмування HTML, JS та TailwindCSS було створено ефективний та зручний інтерфейс, який забезпечує комфортну взаємодію користувача з програмою. Одним із ключових компонентів інтерфейсу стали графіки, які були реалізовані за допомогою бібліотеки Charts.js[18]. Це потужний інструмент, що надає можливості для створення різноманітних типів графіків з високою ступенем кастомізації. Графіки допомагають візуалізувати дані щодо використання програм та зробити їх зрозумілими та доступними для аналізу.

3.3 Розробка програмного модулю реєстрації дій користувача

Головним завданням програмного застосунку для моніторингу та аналізу активності користувача є збір даних, тому цей алгоритм є важливою складовою програмного застосунку. Збір даних дозволяє відслідковувати активність користувача, аналізувати його продуктивність та визначати, як користувач використовує свій час протягом дня. Ефективність і точність цього процесу залежать від правильного функціонування алгоритму збору даних, який повинен бути надійним та оперативним.

Спочатку був розроблений метод, який звертається до бібліотеки операційної системи і дістає інформацію про активне вікно. Цей метод продемонстрований на рисунку 3.6.

```
public ProcessInfo getActiveWindowInfo() {
    HWND fg = User32.INSTANCE.GetForegroundWindow();

    int pid = getPIDofWindow(fg);
    ProcessHandle.Info pInfo = getProcessInfo(pid);
    if (pInfo == null) {
        log.warn("Can't get wondow information of pid: " + pid);
        return null;
    }
    char[] buffer = new char[User32.INSTANCE.GetWindowTextLength(fg) + 1];
    User32.INSTANCE.GetWindowText(fg, buffer, buffer.length);
    String fxWindowName = Native.toString(buffer);
    return ProcessInfo.builder().pid(pid).windowName(fxWindowName).executablePath(pInfo.command().orElse(""))
        .build();
}
```

Рисунок 3.6 – Метод getActiveWindowInfo()

Після цього було розроблено клас Tracker, який є ключовим компонентом для реєстрації дій користувача. Цей клас відповідає за запуск і управління процесом збору даних про активність користувача в режимі реального часу. Tracker запускає потік реєстрації дій, який працює у фоновому режимі, забезпечуючи безперервний моніторинг активних вікон та додатків. Основні методи та функції цього класу продемонстровані на рисунку 3.7.

```
@PostConstruct
public void start() {
    if (runningThread != null || (runningThread != null && runningThread.isInterrupted())) {
        return;
    }
    Thread t = new Thread(() -> {
        while (true) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                log.error("Tracking thread has crushed: " + e.getMessage());
            }
            if (osHelper.isUserAway()) {
                continue;
            }
            registerActivity();
        }
    });
    t.setName("Tacking thread");
    runningThread = t;
    t.start();
    log.info("Started tracker thread");
}

public void stop() {
    if (!runningThread.isInterrupted()) {
        return;
    }
    runningThread.interrupt();
}

private void registerActivity() {
    ProcessInfo pInfo = osHelper.getActiveWindowInfo();
    if (pInfo == null) {
        return;
    }
    if (pInfo.getPid() == lastProcessInfo.getPid() && pInfo.getWindowName().equals(lastProcessInfo.getWindowName())) {
        entryService.updateLast();
        return;
    }
    entryService.registerNewEntry(pInfo);
    lastProcessInfo = pInfo;
}
```

Рисунок 3.7 – Методи класу Tracker

Однією з головних функцій класу Tracker є запуск потоку реєстрації дій. Цей потік виконується у фоновому режимі, не заважаючи основній діяльності користувача. Потік запускається методом startTracking(), який створює новий потік і запускає основний цикл реєстрації.

Клас EntryService відіграє важливу роль в алгоритмі моніторингу та аналізу активності користувача. Цей клас відповідає за обробку та зберігання даних про активність у базі даних. Він містить кілька ключових методів, серед яких особливе значення мають updateLast() та registerNewEntry(). Ці методи забезпечують точність і актуальність записів, а також зручність подальших розрахунків і аналізу. Метод updateLast() відповідає за оновлення тривалості останнього запису у базі даних. Він виконується щоразу, коли збирається нова інформація про активне вікно, що дозволяє зберігати поточну інформацію у реальному часі. Метод updateLast() продемонстровано на рисунку 3.8.

```
@Override
public void updateLast() {
    lastRegisteredEntry.setDuration(lastRegisteredEntry.getDuration() + 1);
    if (lastRegisteredEntry.getDuration() % 5 == 0) {
        entryRepository.save(lastRegisteredEntry);
        log.info("Updated entry: " +
                lastRegisteredEntry.getId() +
                "| In use " +
                formatter.format(new Date(lastRegisteredEntry.getDuration() * 1000L)));
    }
    LocalDateTime date = LocalDateTime.from(lastRegisteredEntry.getStartTime());
    date = date.plus(lastRegisteredEntry.getDuration(), ChronoUnit.SECONDS);
    if (lastRegisteredEntry.getStartTime().getDayOfMonth() != date.getDayOfMonth()) {
        registerNewEntry(
                ProcessInfo.builder()
                        .executablePath(lastRegisteredEntry.getProgramm().getExecutablePath())
                        .windowName(lastRegisteredEntry.getWindowName())
                        .build());
    }
}
```

Рисунок 3.8 – Метод updateLast()

Метод registerNewEntry() відповідає за створення нового запису у базі даних для фіксації нової сесії активності користувача. Це ключовий елемент забезпечення точного та структурованого збору даних, який дозволяє правильно відображати активність користувача. Метод виконує кілька

важливих кроків для створення нового запису та гарантує, що всі дані зберігаються належним чином. Метод registerNewEntry() продемонстровано на рисунку 3.9.

```
@Override
public void registerNewEntry(ProcessInfo pInfo) {
    if (lastRegisteredEntry != null) {
        entryRepository.save(lastRegisteredEntry);
        log.info("Updated entry: " +
        lastRegisteredEntry.getId() +
        "| In use " +
        formatter.format(new Date(lastRegisteredEntry.getDuration() * 1000L)));
    }
    Programm programm = programmService.findByExePath(pInfo.getExecutablePath());
    if (programm == null) {
        programm = programmService.registerNewProgramm(pInfo.getExecutablePath());
    }
    Entry entry = Entry.builder()
            .windowName(pInfo.getWindowName())
            .startTime(LocalDateTime.now())
            .duration(1)
            .programm(programm)
            .build();
    lastRegisteredEntry = entryRepository.save(entry);
    log.info("Registered new activity : " + lastRegisteredEntry.getId() + " " + lastRegisteredEntry.getWindowName());
}
```

Рисунок 3.9 – Метод registerNewEntry()

Метод registerNewEntry() тісно взаємодіє з іншими компонентами системи, такими як клас Tracker та метод updateLast(). Ця взаємодія забезпечує безперервний і точний збір даних, а також їх правильну обробку та зберігання. Використання цього методу дозволяє створити надійну і ефективну систему моніторингу активності користувача.

3.4 Розробка системи взаємодії з базою даних

Основна задача розробки системи взаємодії з базою даних полягає в створенні програмного забезпечення, яке забезпечить ефективний та безперебійний обмін інформацією між користувачами та базою даних. Ця система повинна забезпечити можливість зчитування, запису та оновлення даних в базі даних з врахуванням потреб користувачів та обмежень безпеки. Вона також має забезпечити оптимальний рівень продуктивності та надійності, щоб забезпечити ефективну роботу системи в будь-який час і в умовах великої кількості користувачів та обсягу даних.

Для забезпечення ефективного зберігання та обробки даних у програмному застосунку для моніторингу та аналізу активності користувача, були розроблені сутності, що відповідають таблицям у базі даних. Ці сутності дозволяють структурувати дані та забезпечити їх збереження у зручному форматі. Вони були створені відповідно до специфікації Jakarta Persistence API (JPA), яка надає можливість визначати, які об'єкти слід зберігати та як вони зберігаються у Java-застосунках[19]. Розроблені сутності продемонстровані на рисунку 3.10.

```java
@Data
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Programm {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    @Column(name = "ecexutable_path")
    private String executablePath;
    private String name;
    private String company;
    @Column(name = "display_color")
    private String displayColor;
    @Column(name = "logo_path")
    private String logoPath;

    @ManyToMany
    @JoinTable(
        name="tagged_programs",
        joinColumns = @JoinColumn(name = "programm_id"),
        inverseJoinColumns = @JoinColumn(name = "tag_id"))
    private Set<Tag> tags;
}
```

```java
@Data
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Entry {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "window_name")
    private String windowName;
    @Column(name = "start_time",columnDefinition = "TIMESTAMP")
    private LocalDateTime startTime;
    @Column(name = "duration")
    private int duration;

    @ManyToOne
    @JoinColumn(name = "programm_id", nullable = false)
    private Programm programm;

    @ManyToOne
    @JoinColumn(name = "document_id", nullable = true)
    private Document document;
}
```

Рисунок 3.10 – Сутності, розроблені для застосунку

Після розробки сутностей, що відповідають таблицям у базі даних, було створено інтерфейси репозиторіїв для взаємодії з цими сутностями. Репозиторії використовують Spring Data для автоматизації процесу створення реалізацій сховищ даних, що значно спрощує доступ до бази даних. Для кожної сутності була створена відповідна реалізація репозиторію, ці інтерфейси успадковуються від стандартного інтерфейсу JpaRepository, що надає базові методи для роботи з базою даних, такі як збереження, видалення та пошук. Інтерфейс ProgramRepository є прикладом репозиторію, який було створено для

роботи з сутністю Program. На рисунку 3.11 продемонстровано структуру цього інтерфейсу.

```
public interface ProgrammRepository extends JpaRepository<Programm, Long>{
    
    Optional<Programm> findOneByExecutablePath(String executablePath);
    
    @Query("""
            SELECT p
            FROM Programm p
            JOIN Entry e ON p.id = e.programm.id
            WHERE DATE(e.startTime) = :startTime
            """)
    List<Programm> findAllByEntriesStartTime(LocalDateTime startTime);
    
    @Query("""
            SELECT p
            FROM Programm p
            JOIN Entry e ON p.id = e.programm.id
            WHERE EXTRACT(YEAR FROM e.startTime) = :year
            """)
    List<Programm> findAllByYear(int year);
}
```

Рисунок 3.11 – Інтерфейс репозиторію ProgrammRepository

Для задоволення потреб у специфічних запитах, які не можуть бути повністю описані в назві методу, були розроблені власні SQL-запити. Цей підхід дозволяє поєднати простоту використання об'єктно-орієнтованого доступу до даних (DAO) і потужність мови SQL. Розширення інтерфейсу дозволяє отримати набір релевантних методів CRUD (створення, читання, оновлення, видалення) для стандартного доступу до даних, які доступні в стандартному DAO. Для написання власних SQL-запитів використовуються анотації @Query у Spring Data. Це дозволяє інтегрувати SQL-запити безпосередньо в інтерфейс репозиторію, забезпечуючи зручний спосіб доступу до них через об'єктно-орієнтований інтерфейс. Власні SQL-запити інтегруються у бізнес-логіку через сервіси, що взаємодіють з репозиторіями. Сервіси викликають методи репозиторіїв для виконання запитів і обробляють отримані дані відповідно до бізнес-правил застосунку. Показником ефективності цього

підходу є наявність більш специфічних SQL-запитів, що демонструються на рисунку 3.12. Це дозволяє точніше визначати та обробляти потреби системи, забезпечуючи більш гнучкий та ефективний доступ до даних.

```
@Repository
public interface EntryRepository extends JpaRepository<Entry, Long> {

    @Query("""
            SELECT e
            FROM Entry e
            WHERE EXTRACT(YEAR FROM e.startTime) = :year
            ORDER BY DATE(e.startTime)
            """)
    List<Entry> findAllEnriesByYear(int year);

    @Query("""
            SELECT e
            FROM Entry e
            WHERE EXTRACT(YEAR FROM e.startTime) = EXTRACT(YEAR FROM CAST(:startTime AS timestamp))
            AND EXTRACT(MONTH FROM e.startTime) = EXTRACT(MONTH FROM CAST(:startTime AS timestamp))
            AND EXTRACT(DAY FROM e.startTime) = EXTRACT(DAY FROM CAST(:startTime AS timestamp))
            AND e.programm = :programm
            """)
    List<Entry> findAllByStartTimeAndProgramm(LocalDateTime startTime, Programm programm);

    @Query("""
            SELECT e
            FROM Entry e
            WHERE EXTRACT(YEAR FROM e.startTime) = :year
            AND e.programm = :programm
            """)
    List<Entry> findAllByYearAndProgramm(int year, Programm programm);
}
```

Рисунок 3.12 – Інтерфейс репозиторію EntryRepository

Наступним кроком розробки системи взаємодії з базою даних було створення сервісів. Сервіси використовуються для ізоляції бізнес-логіки та функціональності застосунка від інших шарів, таких як контролери та репозиторії. Це дозволяє підтримувати чистоту коду та розділяти обов'язки між різними компонентами програми. Приклад методу сервісу, який створює Map всіх програм, що використовувалися за день, представлено на рисунку 3.13. Цей метод виконує пошук відповідних записів в репозиторіях програм для кожної програми і додає їх до Map.

```
@Override¤¶
public Map<Programm, List<Entry>> getAllProgrammEntriesbyDay(LocalDateTime date) {¤¶
    List<Programm> programms = programmRepository.findAllByEntriesStartTime(date);¤¶
    Map<Programm, List<Entry>> map = new HashMap<>();¤¶
    programms.forEach(programm -> {¤¶
        List<Entry> entries = entryRepository.findAllByStartTimeAndProgramm(date, programm);¤¶
        map.put(programm, entries);¤¶
    });¤¶
    return map;¤¶
}¤¶
```

Рисунок 3.13 – Метод getAllProgrammEntriesbyDay

Також сервіси використовуються для додаткової абстракції взаємодії з базою даних шляхом створення рівня обробки даних перед збереженням в базу даних або після їх отримання. Метод створення нової програми, як показано на рисунку 3.14, є прикладом цього підходу.

```
@Override¤¶
public Programm registerNewProgramm(String executablePath) {¤¶
    File exe = new File(executablePath);¤¶
    BufferedImage logo = JIconExtract.getIconForFile(128, 128, executablePath);¤¶
    File logoFile = new File(¤¶
            System.getProperty("user.home") +¤¶
            "\\" +¤¶
            userDataFolder +¤¶
            "\\icons\\" +¤¶
            exe.getName() +¤¶
            ".png");¤¶
    try {¤¶
        ImageIO.write(logo, "png", logoFile);¤¶
    } catch (IOException e) {¤¶
        log.warn("Can't save logo : " + e.getMessage());¤¶
    }¤¶
    Programm programm = Programm.builder()¤¶
            .executablePath(executablePath)¤¶
            .name(osHelper.getExeName(executablePath))¤¶
            .company(osHelper.getExeCompany(executablePath)).logoPath(logoFile.getAbsolutePath())¤¶
            .displayColor(osHelper.generateHexColor(executablePath)).build();¤¶
    ¤¶
    Programm savedProgramm = programmRepository.save(programm);¤¶
    log.info("Registered new program : " + savedProgramm.getId() + " " + savedProgramm.getName());¤¶
    return savedProgramm;¤¶
}¤¶
```

Рисунок 3.14 – Метод registerNewProogramm

Отже, для забезпечення ефективної роботи застосунку було розроблено систему взаємодії з базою даних. Ця система дозволяє легко та зручно відправляти дані до бази даних і здійснювати запити для отримання необхідної інформації для подальшого відображення на інтерфейсі користувача.

3.5 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки системи. Проаналізувавши потреби програмного застосунку було обрано мову програмування Java та Spring Framework за їх стабільність та універсальність. Для зручності розробки графічного інтерфейсу було обрано HTML, JS та Tailwind за простоту використання та доступність. В якості бази даних було обрано PostgreSQL.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Суть тестування цього застосунку полягає в перевірці його функціональності та надійності через проведення різноманітних видів тестів[20]. Це включає модульне тестування окремих компонентів застосунку, функціональне тестування його як єдиного цілого, перевірку коректності та ефективності роботи графічного інтерфейсу на HTML, JS та Tailwind, а також тестування сумісності та продуктивності на різних пристроях та в різних середовищах. Метою тестування є забезпечення високої якості програмного продукту та відповідності його функціональності вимогам користувачів.

Модульне тестування є критично важливим процесом у розробці програмного забезпечення, оскільки дозволяє перевірити окремі компоненти або модулі на коректність їх роботи в ізоляції від інших частин системи[21]. Це допомагає виявити помилки на ранніх етапах розробки, знижуючи ризики та забезпечуючи високу якість кінцевого продукту. У контексті цього застосунку на Java Spring Framework, модульне тестування включає перевірку різних частин коду, таких як сервіси, контролери та репозиторії. Приклад модульного тесту продемонстровано на рисунку 4.1.

```
@Test
void testFindByExePath_ExistingProgramm() throws Exception {
    // Arrange
    String expectedExePath = "C:\\Windows\\explorer.exe";
    Programm expectedProgramm = new Programm();
    expectedProgramm.setExecutablePath(expectedExePath);

    when(programmRepository.findOneByExecutablePath(expectedExePath))
        .thenReturn(Optional.of(expectedProgramm));

    // Act
    Programm actualProgramm = programmService.findByExePath(expectedExePath);

    // Assert
    assertNotNull(actualProgramm);
    assertEquals(expectedProgramm, actualProgramm);
    verify(programmRepository).findOneByExecutablePath(expectedExePath);
}
```

Рисунок 4.1 – Модульний тест

У Spring Framework модульне тестування зазвичай виконується з використанням JUnit і Mockito [22]. JUnit є фреймворком для написання і запуску тестів, а Mockito — бібліотекою для створення мок-об'єктів, які імітують поведінку реальних об'єктів. Модульне тестування здійснюється шляхом створення тестів, які перевіряють функціональність окремих методів або класів без взаємодії з іншими компонентами системи. Це забезпечує точне визначення місця виникнення помилки та спрощує процес її усунення.

Також були розроблені модульні тести на відмову. Тестування на відмову допомагає виявити крайні випадки і граничні умови, які можуть бути не відразу очевидні під час нормальної роботи. Це сценарії, де вхідні дані або умови знаходяться на крайніх кінцях спектру або де може виникнути неочікувана поведінка. Тестуючи випадки відмов, розробники можуть переконатися, що їхні методи справляються з цими сценаріями без проблем.

Модульні тести на відмову включають перевірку реакції системи на різні типи некоректних або неочікуваних даних. Наприклад, це можуть бути порожні значення, значення, що перевищують допустимі межі, або дані неправильного типу. Модульне тестування на відмову є важливим аспектом забезпечення якості програмного забезпечення. Воно дозволяє перевірити, як система реагує на крайні випадки і граничні умови, забезпечуючи її надійність і стійкість до помилок. Завдяки тестуванню на відмову розробники можуть виявити і виправити потенційні проблеми на ранніх етапах, що сприяє підвищенню загальної якості і стабільності програмного забезпечення.

Також тестування сценаріїв відмов дозволяє переконатися, що механізми обробки помилок працюють належним чином. Воно гарантує, що відповідні повідомлення про помилки відображаються або реєструються, і що додаток виходить з ладу м'яко, а не поширює помилки або несподівано завершує роботу. Приклад модульного тесту сценарію відмови продемонстровано на рисунку 4.2.

```
@Test()¤¶
void testRegisterNewProgramm_LogoSaveFailure() throws IOException {¤¶
»   //·Arrange¤¶
»   String expectedExePath = "C:\\path\\to\\program.exe";¤¶
»   String expectedName = "Program Name";¤¶
»   String expectedCompany = "Program Company";¤¶

»   when(osHelper.getExeName(expectedExePath)).thenReturn(expectedName);¤¶
»   when(osHelper.getExeCompany(expectedExePath)).thenReturn(expectedCompany);¤¶
»   when(osHelper.generateHexColor(expectedExePath)).thenReturn("#123456");¤¶

»   //·Simulate logo extraction (avoid actual file operations)¤¶
»   BufferedImage mockLogo = Mockito.mock(BufferedImage.class);¤¶
»   when(JIconExtract.getIconForFile(128, 128, expectedExePath)).thenReturn(mockLogo);¤¶

»   //·Mock IOException to simulate logo saving failure¤¶
»   Mockito.doThrow(new IOException("Simulated logo save error"))¤¶
»   »   »   .when(ImageIO.write(mockLogo, "png", Mockito.any(File.class)));¤¶

»   //·Act (expect exception)¤¶
»   programmService.registerNewProgramm(expectedExePath);¤¶
}¤¶
```

Рисунок 4.2 – Модульний тест сценарію відмови

Для модульного тестування зазвичай використовуються фреймворки JUnit або Mockito. Тестові класи створюються для кожного класу або методу, який потрібно протестувати. У тестових методах встановлюються вхідні дані та очікувані результати, і після цього викликається тестований метод для перевірки правильності її роботи. Кожен тестовий метод включає наступні кроки:

– Підготовка: Встановлення початкових умов, включаючи створення необхідних об'єктів і налаштування початкового стану.

– Виконання: Виклик методу, який потрібно протестувати, з передачею необхідних вхідних даних.

– Перевірка: Порівняння фактичного результату з очікуваним результатом для визначення, чи пройшов тест.

Результати виконання модульних тестів були продемонстровані на рисунку 4.3.

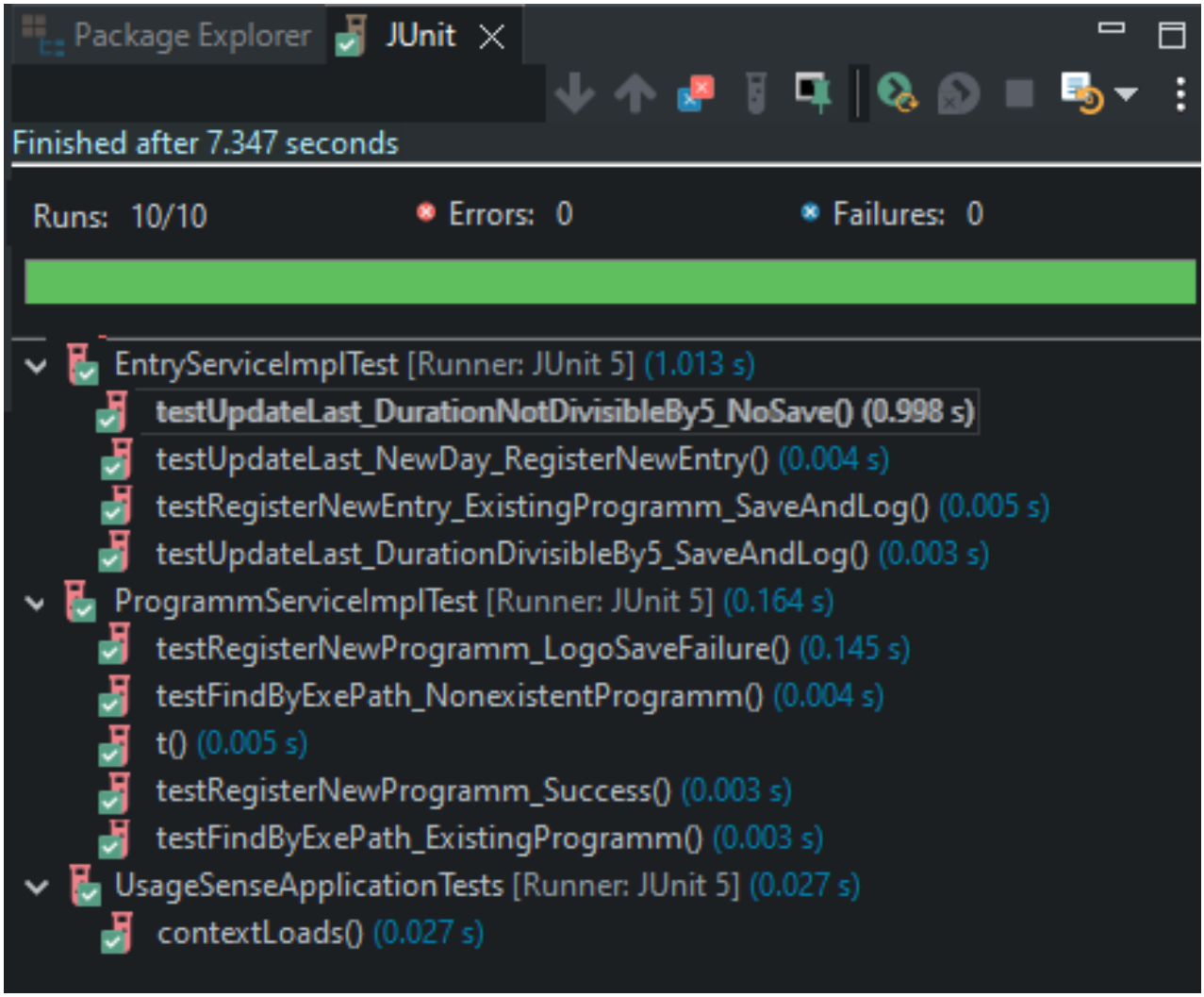


Рисунок 4.3 – Результати модульного тестування

Після цього була проведена оцінка продуктивності за допомогою Firefox Profiler. Оцінка продуктивності веб-сайту за допомогою Firefox Profiler є важливим етапом у виявленні та вирішенні проблем, пов'язаних з швидкістю та ефективністю завантаження сторінок. Firefox Profiler - це інструмент для аналізу продуктивності, який надає детальну інформацію про час виконання різних операцій у вашому веб-додатку. Використання Firefox Profiler дозволяє визначити, які частини коду або які елементи на сторінці впливають на час завантаження. Це дає можливість оптимізувати ці елементи для прискорення роботи додатку. Аналіз продуктивності допомагає виявити вузькі місця, де витрачається найбільше часу або ресурсів, що дозволяє розробникам сфокусувати свої зусилля на оптимізації саме цих частин. Швидка і ефективна

робота веб-сайту значно підвищує задоволеність користувачів, знижуючи час очікування та покращуючи загальну взаємодію з додатком. Результат роботи Firefox Profiler продемонстровано на рисунку 4.4.

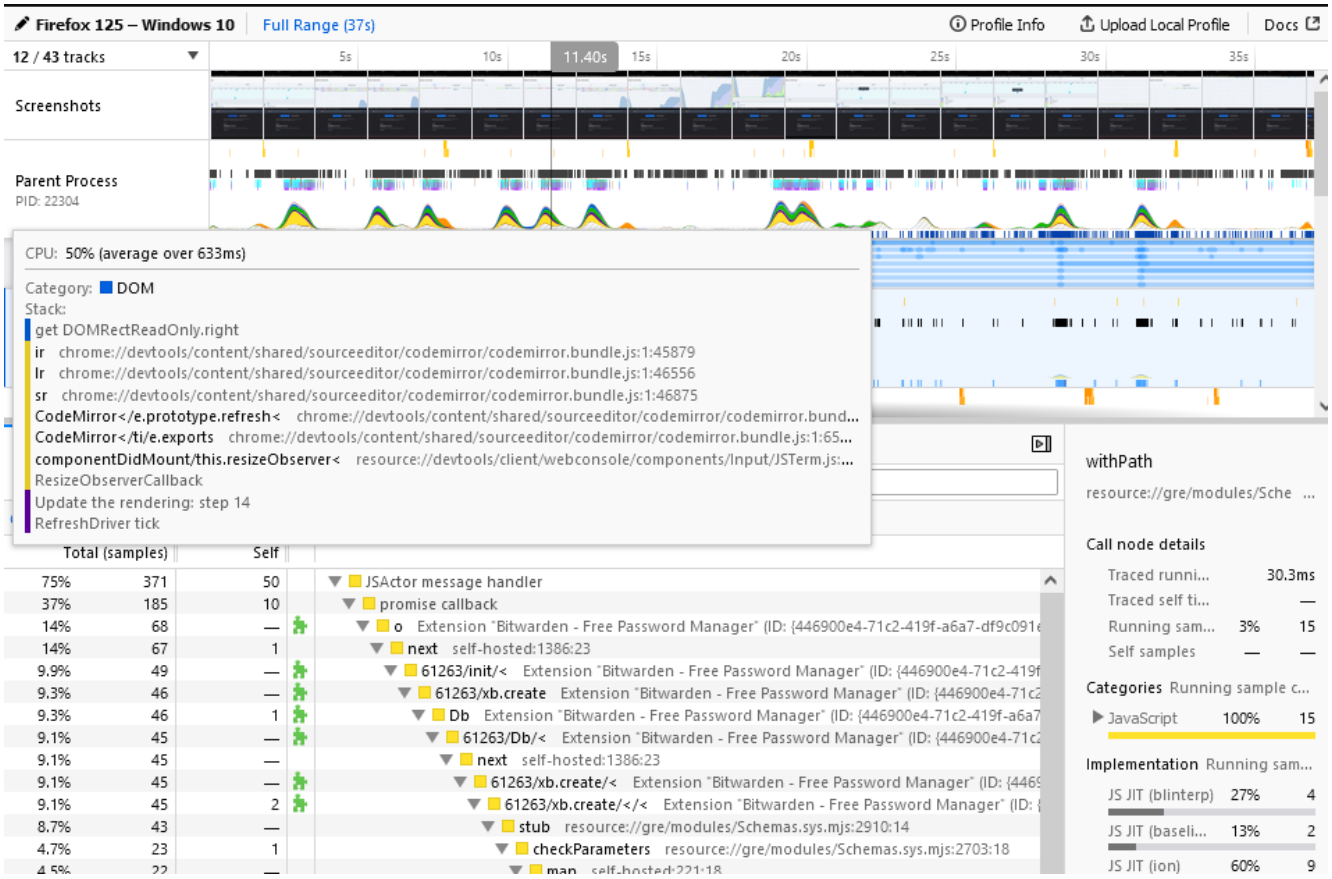


Рисунок 4.4 – Firefox Profiler

Профайлер дозволяє виміряти час, який потрібно для завантаження сторінки та її вмісту. Це допомагає виявити можливі затримки та оптимізувати процес завантаження. Також він надає можливість перевірити, які ресурси, такі як JavaScript, CSS, зображення тощо, використовуються вашим сайтом і як довго вони завантажуються. Це дозволяє виявити можливі проблеми з оптимізацією та ресурсоємність веб-сторінок. Профайлер може виявити функції або скрипти, які забирають багато часу для виконання. Це дозволяє вам знайти "точки гальма" у вашому коді та виправити їх для покращення продуктивності. Крім часу виконання, профайлер також надає інформацію про використання пам'яті вашим сайтом. Це допомагає виявити витоки пам'яті та оптимізувати

використання ресурсів.

Також було проведено тестування продуктивності в браузері на основі Chromium. Тестування продуктивності в браузері на основі Chromium має велику важливість для забезпечення високої якості веб-додатків та забезпечення задоволення користувачів. Результати тестування, продемонстровані на рисунку 4.5.

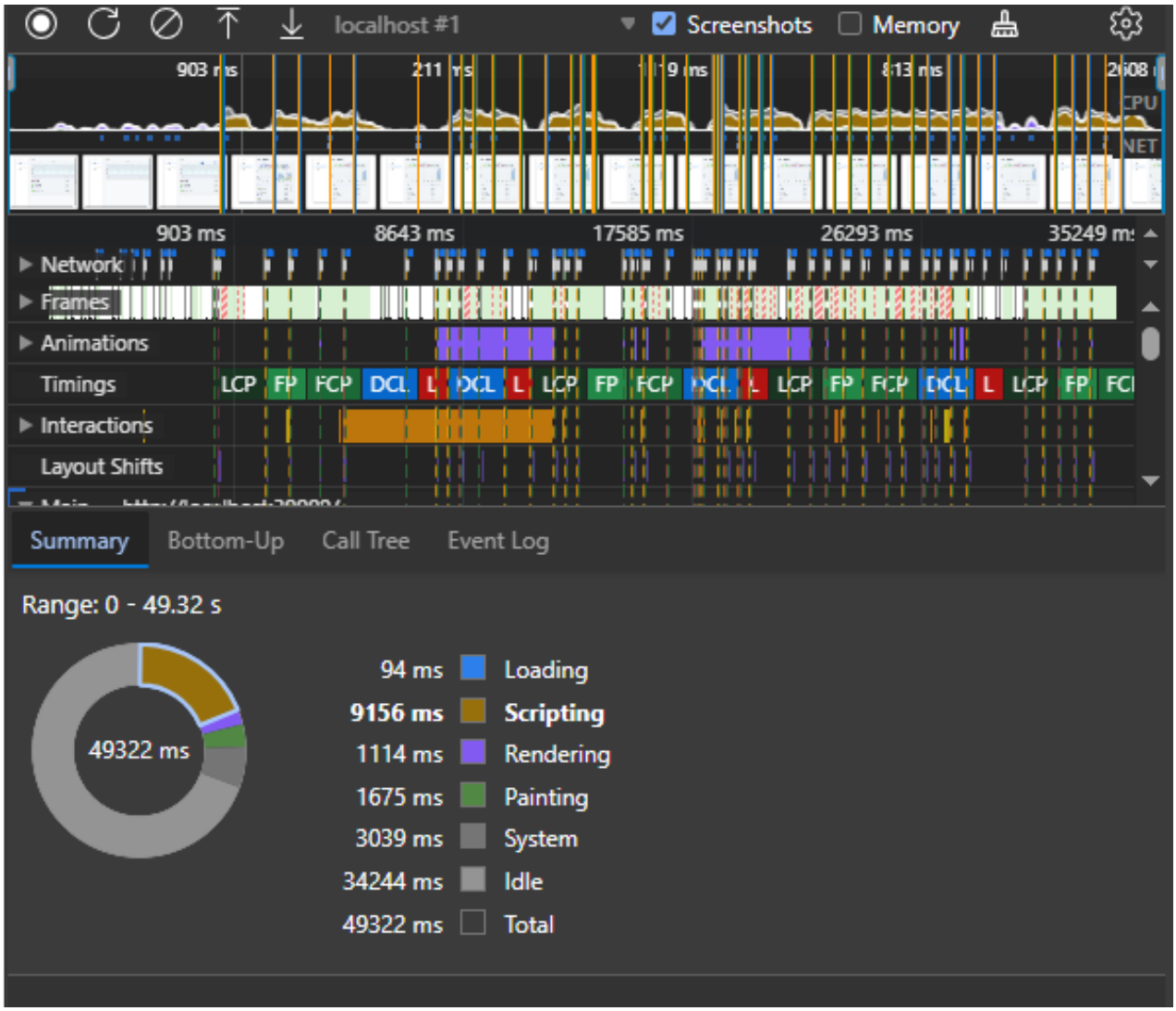


Рисунок 4.5 – Оцінка продуктивності в Edge

Отримавши результати оцінки продуктивності в браузерах на основі Chromium, можна зробити висновок, що застосунок має хорошу продуктивність у всіх найпопулярніших браузерах. Це означає, що швидкодія завантаження сторінок, використання ресурсів та загальна продуктивність додатка

відповідають або перевищують очікування користувачів. Такий результат свідчить про високу якість розробленого веб-додатка та успішну оптимізацію його роботи. Це, в свою чергу, сприяє покращенню користувацького досвіду та збільшенню його задоволення від використання додатка.

Після цього було проведено тестування інтерфейсу застосунку. Під час ручного тестування інтерфейсу перевіряється, як працює застосунок. Виконуються певні дії на сайті та перевіряються, чи працює все правильно. Якщо буде виявлено проблему, її записують, щоб розробники могли виправити. Такий процес допомагає зробити продукт кращим і забезпечити задоволення користувачів.

Ручне тестування дозволяє зрозуміти, як користувачі будуть взаємодіяти з застосунком, виявляючи проблеми, які можуть бути неочевидні під час автоматичного тестування[23]. Тестувальники можуть оцінити зовнішній вигляд і поведінку елементів інтерфейсу, перевіряючи коректність відображення і розташування всіх елементів. Під час ручного тестування часто виявляються дрібні, але важливі дефекти, які можуть вплинути на загальний користувацький досвід.

Під час ручного спочатку перевіряється головна сторінка застосунку. Визначаються основні сценарії використання, які потрібно перевірити. Це можуть бути реєстрація користувача, логін, навігація по сторінках, виконання певних дій на сайті тощо. Тестувальники виконують попередньо визначені сценарії, спостерігаючи за поведінкою застосунку. Вони перевіряють, чи всі функції працюють відповідно до вимог. Усі знайдені проблеми та помилки ретельно документуються. Записуються подробиці про проблему, умови її виникнення і можливі кроки для її відтворення. Після завершення тестування результати передаються команді розробників для виправлення виявлених дефектів.

Було перевірено, як працює перемикання року та правильність даних у таблицях. Також було перевірено правильність посилань на детальний огляд в комірках дат. Якщо обраний рік не має даних у базі даних, всі клітинки в

таблиці про кількість годин проведених за комп’ютером за кожен день мають бути порожніми, а в списку застосунків має з'явитися повідомлення «В цьому році не було зареєстровано ніякої активності». Результати тестування показано на рисунку 4.6. Цей процес допомагає перевірити, як працює застосунок у різних умовах та гарантує правильність відображення даних для користувачів.

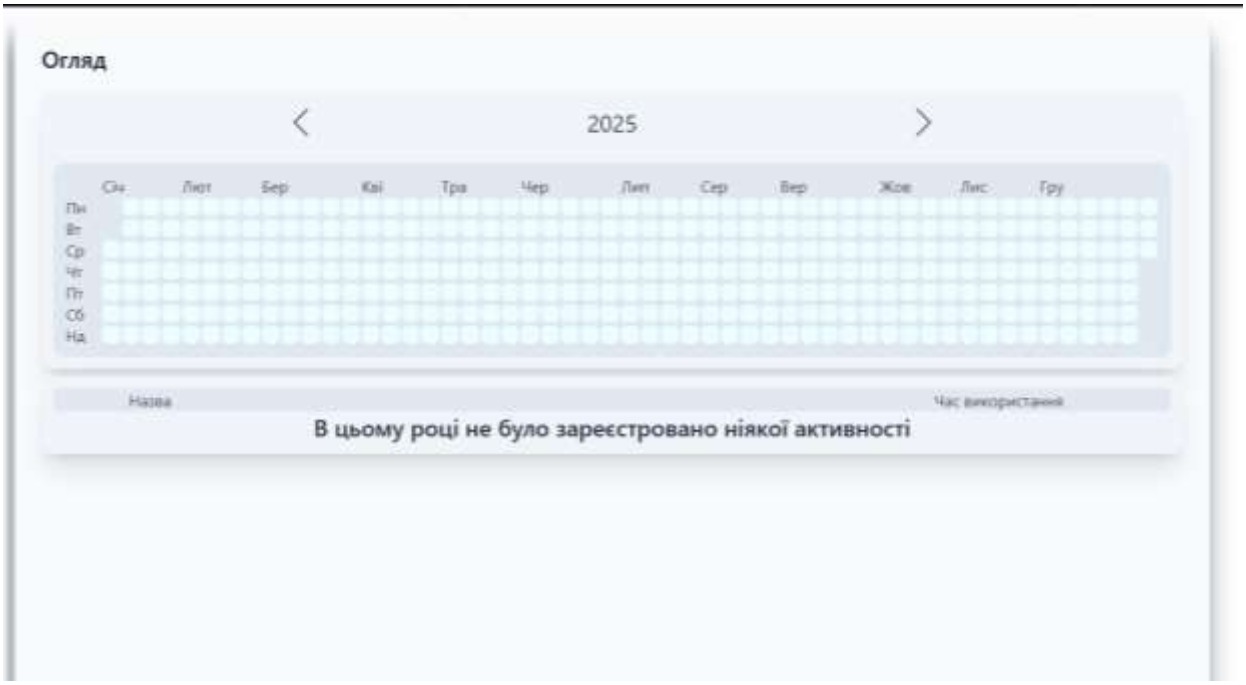


Рисунок 4.6 – Тестування спроби вибору пустого року

Після перевірки головної сторінки застосунку було проведено тестування сторінки детального огляду активності за день. Було перевірено можливість перемикання дати для огляду активності, щоб переконатися, що користувач може легко обрати потрібний день для перегляду детальних даних. Тестувальники обирали різні дати, щоб переконатися, що сторінка оновлюється відповідно до обраної дати.

У нормальних умовах на цій сторінці має відображатися графік погодинного використання застосунків за обраний день. Тестувальники перевірили, чи правильно відображаються всі дані на графіку, включаючи час і назви застосунків. Також було перевірено правильність відображення списку програм і загального часу їх використання. Список має відображати всі

програми, що використовувались у цей день, разом з відповідним часом використання для кожної програми. Було обрано дати, для яких не було даних у базі. У таких випадках, замість графіку, повинне з'являтися повідомлення: «В цей день не було зареєстровано ніякої активності», а список застосунків має залишитися пустим. Результати тестування цієї сторінки показано на рисунку 4.7. Такий процес перевірки допомагає забезпечити коректну роботу сторінок та правильне відображення інформації для користувачів.

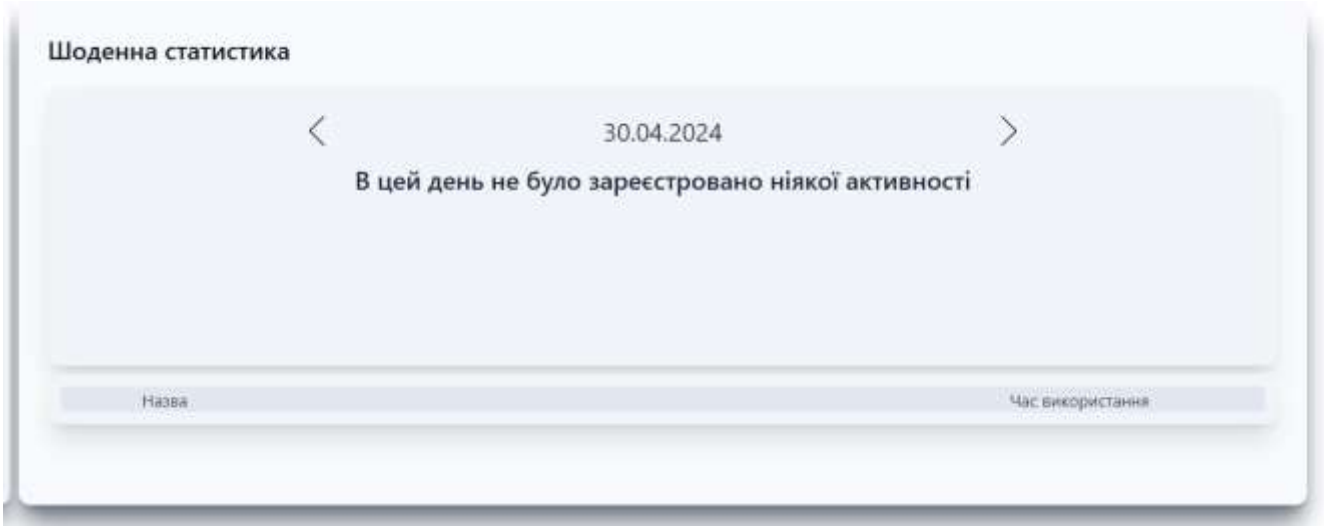


Рисунок 4.7 – Тестування спроби вибору дати без даних

Отже, тестування програмного застосунку включає оцінку функціональності, надійності та продуктивності програмного забезпечення. Комплексне тестування програмного застосунку, яке включало модульне, інтеграційне, продуктивне та ручне тестування, забезпечило високу якість кінцевого продукту. Такий підхід дозволив виявити та виправити помилки на різних етапах розробки, оптимізувати продуктивність і гарантувати зручність використання для кінцевих користувачів. Завдяки цьому застосунок став надійним, ефективним і відповідним очікуванням користувачів, що сприяє його успішному впровадженню та використанню.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп'ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	2-ядерний процесор з частотою 1.5 ГГц
Об'єм оперативної пам'яті	4 ГБ для 64-разрядної системи
Місце на жорсткому диску	2 ГБ
Операційна система	Необхідна підтримка Java 21 (Windows 10, 11, Oracle Linux 7+, Ubuntu 23.10+, IOS 11+)

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	4-ядерний процесор з частотою 2 ГГц
Об'єм оперативної пам'яті	8 ГБ для 64-разрядної системи
Місце на жорсткому диску	4 ГБ
Операційна система	Необхідна підтримка Java 21 (Windows 10, 11, Oracle Linux 7+, Ubuntu 23.10+, IOS 11+)

Після запуску застосунку користувач може вільно переходити до виконання своїх робочих завдань на комп'ютері. За допомогою кнопки у системному лотку він має можливість відкрити інтерфейс програми, де відображені дані про його робочі звички (рис. 4.8). Поступово, коли застосунок накопичить достатню кількість інформації про час та програми, які він використовує, користувач може переглядати ці дані і використовувати для аналізу своєї продуктивності.

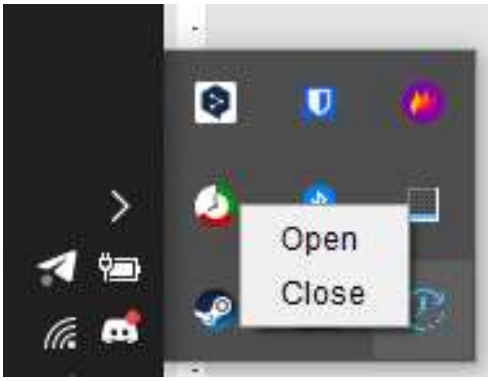


Рисунок 4.8 – Кнопка відкриття графічного інтерфейсу в системному лотку

За допомогою аналізу зібраних даних користувач може ідентифікувати патерни в своєму робочому процесі та зосередитися на їх оптимізації. Наприклад, він може виявити, що проводить багато часу на певних веб-сайтах або у певних програмах, і вирішити встановити обмеження на цей час або шукати способи підвищення ефективності роботи в цих середовищах. Такий підхід допомагає користувачеві краще управляти своїм часом та підвищити продуктивність.

Підсумовуючи, застосунок дозволяє користувачеві активно контролювати та аналізувати свої робочі звички, що сприяє підвищенню свідомості та ефективності у виконанні робочих завдань. Шляхом аналізу зібраних даних користувач може виявити області для поліпшення та прийняти кроки для оптимізації свого робочого процесу.

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів програмного застосунку для моніторингу та аналізу активності користувача. Було перевірено швидкодію та стабільність застосунку. Також було проведено ручне тестування інтерфейсу користувача. Було розроблено інструкцію користувача по встановленню та початковій експлуатації програмного застосунку.

ВИСНОВКИ

У цій бакалаврській дипломній роботі було розглянуто та досліджено ряд важливих аспектів розробки програмного застосунку для моніторингу та аналізу активності користувача. Основною метою дослідження було створення зручного та функціонального інструменту для покращення продуктивності та ефективності роботи користувача за комп'ютером шляхом моніторингу та аналізу його активності.

У результаті аналізу сучасних інструментів та засобів в розробці програмних застосунків було визначено, що для розробки будуть використовуватись такі засоби: Java, Spring Framework – для серверної частини і HTML, JavaScript та Tailwind – для клієнтської частини.

У бакалаврській дипломній роботі виконано такі задачі:

– визначено вимоги до програмного застосунку;

– спроєктовано базу даних, сумісну з вимогами програмного застосунку;

– розроблено модуль для управління та комунікації з базою даних;

– розроблено модуль для відстеження та збереження використання програм користувачем;

– розроблено модуль графічного інтерфейсу, який може формувати представляти історію використаних програм та формувати графіки;

– проведено тестування програмного застосунку.

Для розробки програмного застосунку розроблено UML-діаграму моделі роботи системи та алгоритми роботи системи: алгоритм реєстрації дій користувача та алгоритм взаємодії з базою даних.

У результаті тестування застосунку визначено, що застосунок працює стабільно. Ручне тестування інтерфейсу показало, що досвіду користувача відповідає задуму.

ЛІТЕРАТУРА

1. Продуктивність праці – що таке, формула, методи вимірювання [Електронний ресурс] – Режим доступу до ресурсу: https://workafford.com.ua/produktivnist-praczi/ (дата звернення: 15.03.2024).

2. Як програмне забезпечення для відстеження співробітників може покращити обслуговування клієнтів [Електронний ресурс] – Режим доступу до ресурсу: https://clevercontrol.com/uk/how-employee-tracking-software/ (дата звернення: 15.03.2024).

3. Тайм-менеджмент: ефективні методики управління часом [Електронний ресурс] – Режим доступу до ресурсу: https://interkassa.com/blog/tajm-menedzhment-efektivni-metodiki-upravlinnya-chasom (дата звернення: 16.03.2024).

4. Employee monitoring software: The Top four Monitor how your staff uses the computers [Електронний ресурс] – Режим доступу до ресурсу: https://clevercontrol.com/employee-monitoring-software-the-top-four-monitor-how-your-staff-uses-the-computers/ (дата звернення: 18.03.2024).

5. Unlocking Business Insights: The Power of Visual Reporting in Data Analysis [Електронний ресурс] – Режим доступу до ресурсу: https://datamyte.com/blog/visual-reporting/ (дата звернення: 18.03.2024).

6. Кондратенко Ю.В. АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ЗАСТОСУНКУ «USAGESENSE» ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ АКТИВНОСТІ КОРИСТУВАЧА / Ю. В. Кондратенко Г. Б. Ракитянська // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» , Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21149

7. Manage your time better [Електронний ресурс] – Режим доступу до ресурсу: https://www.manictime.com/ (дата звернення: 21.03.2024).

8. Toggl track [Електронний ресурс] – Режим доступу до ресурсу: https://toggl.com/ (дата звернення: 21.03.2024).

9. RescueTime: Fully Automated Time Tracking Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.rescuetime.com/ (дата звернення: 24.03.2024).

10. Clockify™ - FREE Time Tracking Software [Електронний ресурс] – Режим доступу до ресурсу: https://clockify.me/ (дата звернення: 24.03.2024).

11. UI/UX дизайн: принципи та методи створення зручного інтерфейсу користувача [Електронний ресурс] – Режим доступу до ресурсу: https://whileweb.com/uk/blog/uiux-dizajn-principi-ta-metodi-stvorennya-zruchnogo-interfejsu-koristuvacha/ (дата звернення: 08.04.2024).

12. The Unified Modeling Language [Електронний ресурс] – Режим доступу до ресурсу: https://www.uml-diagrams.org/ (дата звернення: 09.04.2024).

13. What is a flowchart? A complete guide [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/flowchart/what-is-a-flowchart/ (дата звернення: 10.04.2024).

14. Craig Walls. Spring in Action – Manning Publications – 2022 – 492 ст.

15. Luca Ferrari , Enrico Pirozzi. Learn PostgreSQL - Second Edition – Packt Publishing – 2023 – 744 ст.

16. Overview (JNA API) [Електронний ресурс] – Режим доступу до ресурсу: http://lynchjim.com/doc/libjna-java-doc/api/overview-summary.html (дата звернення: 29.04.2024).

17. How to choose the best color palette for your mobile app design [Електронний ресурс] – Режим доступу до ресурсу: https://decode.agency/article/choose-color-palette-for-apps/ (дата звернення: 03.05.2024).

18. Chart.js | Chart.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.chartjs.org/docs (дата звернення: 04.05.2024).

19. Introduction to Jakarta Persistence [Електронний ресурс] – Режим доступу до ресурсу: https://jakarta.ee/learn/docs/jakartaee-

tutorial/current/persist/persistence-intro/persistence-intro.html (дата звернення: 08.05.2024).

20. Why is Testing Necessary? [Електронний ресурс] – Режим доступу до ресурсу: https://www.toolsqa.com/software-testing/istqb/why-is-testing-necessary/ (дата звернення: 15.05.2024).

21. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship – Pearson – 2008 – 464 ст.

22. Unit Testing with JUnit 4 [Електронний ресурс] – Режим доступу до ресурсу: https://www.vogella.com/tutorials/JUnit4/article.html (дата звернення: 17.05.2024).

23. UI Testing Checklist [Електронний ресурс] – Режим доступу до ресурсу: https://www.browserstack.com/guide/ui-testing-checklist (дата звернення: 18.05.2024).

ДОДАТКИ

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу «Розробка програмного застосунку для моніторингу та аналізу активності користувача » за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. каф. ПЗ Г.Б. Ракитянська
« 12 » березня 2024р.

Виконав:
студент гр. ЗПІ-20Б Ю.В.Кондратенко
« 12 » березня 2024р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного застосунку для моніторингу та аналізу активності користувача».

Галузь застосування – сфера тайм-менеджменту в професійному та особистому використанні.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є розробка та реалізація програмного застосунку для моніторингу та аналізу активності користувача.

Основними задачами є:

– визначити вимоги до програмного застосунку;

– спроєктувати базу даних, сумісну з вимогами програмного застосунку;

– розробити модуль для управління та комунікації з базою даних;

– розробити модуль для відстеження та збереження використання програм користувачем;

– розробити модуль графічного інтерфейсу, який може формувати представляти історію використаних програм та формувати графіки;

– провести тестування програмного застосунку.

4. Вихідні дані для проведення БДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

5. Технічні вимоги

Середовища розробки – Eclipse, Visual Studio Code, мови розробки – Java, JavaScript, HTML, фреймворки – Spring Framework, Tailwind засоби організації даних програми – PostgreSQL, операційна система – Windows, Linux.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та обґрунтування завдання	14.03.2024 – 29.03.2024
2	Аналіз вхідних та вихідних даних системи	30.03.2024 – 07.04.2024
3	Розробка моделі та алгоритмів системи	08.04.2024 – 26.04.2024
4	Розробка інтерфейсу	27.04.2024 – 04.05.2024
5	Розробка програмних модулів системи	05.05.2024 – 14.05.2024
6	Тестування системи	15.05.2024– 20.05.2024
7	Оформлення матеріалів до захисту БДР	21.05.2024 – 30.05.2024

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ

ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного застосунку для моніторингу та аналізу активності користувача

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ракитянська Г. Б.

Оригінальність	95,7%
Схожість	4.3%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

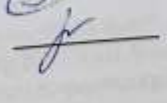
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку  Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи  Кондратенко Ю.В..

Керівник роботи  Ракитянська Г.Б.

ДОДАТОК В – ЛІСТИНГ ПРОГРАМИ

```
package org.fovvox.usagesense;

import java.awt.AWTException;
import java.io.File;
import java.io.IOException;

import org.fovvox.usagesense.ui.TrayController;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;

@SpringBootApplication
public class UsageSenseApplication {

	private TrayController trayController;

	@Value(value = "${user.data.folder}")
	private String userDataFolder;

	public static void main(String[] args) {
		SpringApplication.run(UsageSenseApplication.class, args);
	}

	@PostConstruct
	public void initDataFolder() {
		File folder = new File(System.getProperty("user.home") + userDataFolder);
		if (!folder.exists()) {
			folder.mkdir();
		}
		File icons = new File(folder.getAbsoluteFile() +  "\\icons");
		if (!icons.exists()) {
			icons.mkdir();
		}
	}

	@PostConstruct
	public void initTray() throws AWTException, IOException {
		trayController = new TrayController();
	}

	@PreDestroy
	public void removeTray() {
		trayController.delete();
	}

}

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
```

```
4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.2.5</version>
		<relativePath /> <!-- lookup parent from repository -->
	</parent>
	<groupId>org.fovvox</groupId>
	<artifactId>usagesense</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>UsageSense</name>
	<description>UsageSense - pc usage tracking software</description>
	<properties>
		<java.version>17</java.version>
		<start-class>org.fovvox.usagesense.UsageSenseApplication</start-class>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-thymeleaf</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-thymeleaf</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
		</dependency>

		<dependency>
			<groupId>org.apache.logging.log4j</groupId>
			<artifactId>log4j-api</artifactId>
		</dependency>
		<dependency>
			<groupId>org.apache.logging.log4j</groupId>
			<artifactId>log4j-core</artifactId>
		</dependency>
		<dependency>
			<groupId>org.apache.logging.log4j</groupId>
			<artifactId>log4j-slf4j-impl</artifactId>
		</dependency>
```

```
        <dependency>
                <groupId>org.postgresql</groupId>
                <artifactId>postgresql</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <optional>true</optional>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.restdocs</groupId>
                <artifactId>spring-restdocs-mockmvc</artifactId>
                <scope>test</scope>
        </dependency>

        <dependency>
                <groupId>net.java.dev.jna</groupId>
                <artifactId>jna</artifactId>
                <version>5.14.0</version>
        </dependency>

        <dependency>
                <groupId>net.java.dev.jna</groupId>
                <artifactId>jna-platform</artifactId>
                <version>5.14.0</version>
        </dependency>

</dependencies>

<build>
        <plugins>
                <plugin>
                        <groupId>org.asciidoctor</groupId>
                        <artifactId>asciidoctor-maven-plugin</artifactId>
                        <version>2.2.1</version>
                        <executions>
                                <execution>
                                        <id>generate-docs</id>
                                        <phase>prepare-package</phase>
                                        <goals>
                                                <goal>process-asciidoc</goal>
                                        </goals>
                                        <configuration>
                                                <backend>html</backend>
                                                <doctype>book</doctype>
                                        </configuration>
                                </execution>
                        </executions>
                        <dependencies>
                                <dependency>
                                        <groupId>org.springframework.restdocs</groupId>
```

```
                                        <artifactId>spring-restdocs-asciidoctor</artifactId>
                                        <version>${spring-restdocs.version}</version>
                                    </dependency>
                                </dependencies>
                            </plugin>
                            <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                                <configuration>
                                    <excludes>
                                        <exclude>
                                            <groupId>org.projectlombok</groupId>
                                            <artifactId>lombok</artifactId>
                                        </exclude>
                                    </excludes>
                                </configuration>
                            </plugin>
                            <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                                <configuration>

        <mainClass>org.fovvox.usagesense.UsageSenseApplication</mainClass>
                                    <layout>ZIP</layout>
                                </configuration>
                            </plugin>
                            <plugin>
                                <artifactId>maven-assembly-plugin</artifactId>
                                <configuration>
                                    <archive>
                                        <manifest>

        <mainClass>org.fovvox.usagesense.UsageSenseApplication</mainClass>
                                        </manifest>
                                    </archive>
                                    <descriptorRefs>
                                        <descriptorRef>jar-with-dependencies</descriptorRef>
                                    </descriptorRefs>
                                </configuration>
                            </plugin>
                        </plugins>
        </build>

</project>
```

```
package org.fovvox.usagesense.model.domain;

import java.time.LocalDateTime;
import java.util.Set;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
```

```
import jakarta.persistence.JoinColumn;
import jakarta.persistence.JoinTable;
import jakarta.persistence.ManyToMany;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Document {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	private String name;
	@Column(name = "display_color")
	private String displayColor;

	@ManyToMany
	@JoinTable(
			name="tagged_documents",
			joinColumns = @JoinColumn(name = "document_id"),
			inverseJoinColumns = @JoinColumn(name = "tag_id"))
	private Set<Tag> tags;

}



package org.fovvox.usagesense.model.domain;

import java.time.LocalDateTime;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Index;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Entry {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
	private Long id;

	@Column(name = "window_name")
	private String windowName;
	@Column(name = "start_time",columnDefinition = "TIMESTAMP")
	private LocalDateTime startTime;
	@Column(name = "duration")
	private int duration;

	@ManyToOne
	@JoinColumn(name = "programm_id", nullable = false)
	private Programm programm;

	@ManyToOne
	@JoinColumn(name = "document_id", nullable = true)
	private Document document;
}



package org.fovvox.usagesense.model.domain;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Set;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Index;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.JoinTable;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Programm {

	@Id
	@GeneratedValue(strategy = GenerationType.SEQUENCE)
	private Long id;
	@Column(name = "ecexutable_path")
	private String executablePath;
	private String name;
	private String company;
```

```
	@Column(name = "display_color")
	private String displayColor;
	@Column(name = "logo_path")
	private String logoPath;

	@OneToMany(mappedBy = "programm", fetch = FetchType.LAZY)
	private List<Entry> entryList;

	@ManyToMany
	@JoinTable(
		name="tagged_programs",
		joinColumns = @JoinColumn(name = "programm_id"),
		inverseJoinColumns = @JoinColumn(name = "tag_id"))
	private Set<Tag> tags;


}



package org.fovvox.usagesense.model.domain;

import java.time.LocalDateTime;
import java.util.Set;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Tag {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	private String name;
	@Column(name = "display_color")
	private String displayColor;

	@ManyToMany(mappedBy = "tags")
	private Set<Programm> programms;

	@ManyToMany(mappedBy = "tags")
	private Set<Document> documents;

}
```

```
package org.fovvox.usagesense.model.repository;

import org.fovvox.usagesense.model.domain.Document;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DocumentRepository extends JpaRepository<Document, Long>{

}
```

```
package org.fovvox.usagesense.model.repository;

import java.util.List;

import org.fovvox.usagesense.model.domain.Entry;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.time.LocalDateTime;
import org.fovvox.usagesense.model.domain.Programm;

@Repository
public interface EntryRepository extends JpaRepository<Entry, Long> {

	@Query("""
			SELECT e
			FROM Entry e
			WHERE EXTRACT(YEAR FROM e.startTime) = :year
			ORDER BY DATE(e.startTime)
			""")
	List<Entry> findAllEnriesByYear(int year);

	@Query("""
			SELECT e
			FROM Entry e
			WHERE EXTRACT(YEAR FROM e.startTime) = EXTRACT(YEAR FROM
CAST(:startTime AS timestamp))
			AND EXTRACT(MONTH FROM e.startTime) = EXTRACT(MONTH FROM
CAST(:startTime AS timestamp))
			AND EXTRACT(DAY FROM e.startTime) = EXTRACT(DAY FROM
CAST(:startTime AS timestamp))
			AND e.programm = :programm
			""")
	List<Entry> findAllByStartTimeAndProgramm(LocalDateTime startTime, Programm programm);

	@Query("""
			SELECT e
			FROM Entry e
			WHERE EXTRACT(YEAR FROM e.startTime) = :year
			AND e.programm = :programm
			""")
	List<Entry> findAllByYearAndProgramm(int year, Programm programm);


}
```

```
package org.fovvox.usagesense.model.repository;

import org.fovvox.usagesense.model.domain.Programm;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Optional;


public interface ProgrammRepository extends JpaRepository<Programm, Long>{

        Optional<Programm> findOneByExecutablePath(String executablePath);

        @Query("""
                        SELECT p
                        FROM Programm p
                        JOIN Entry e ON p.id = e.programm.id
                        WHERE DATE(e.startTime) = :startTime
                        """)
        List<Programm> findAllByEntriesStartTime(LocalDateTime startTime);

        @Query("""
                        SELECT p
                        FROM Programm p
                        JOIN Entry e ON p.id = e.programm.id
                        WHERE EXTRACT(YEAR FROM e.startTime) = :year
                        """)
        List<Programm> findAllByYear(int year);

        Optional<Programm> findById(Long id);

        Optional<Programm> findOneByName(String name);
}


package org.fovvox.usagesense.model.repository;

import org.fovvox.usagesense.model.domain.Tag;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TagRepository extends JpaRepository<Tag, Long>{

}


package org.fovvox.usagesense.model.service;

public class EntryNotFoundException extends RuntimeException{
```

```
	public EntryNotFoundException(String message) {
		super(message);
	}

}
```

```
package org.fovvox.usagesense.model.service;

import java.time.LocalDate;
import java.util.List;

import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.tracker.ProcessInfo;

public interface EntryService {

	void updateLast();

	void registerNewEntry(ProcessInfo pInfo);

	List<Entry> getAllEntriesByYear(int year);

	List<Entry> getAllEntriesByDateChartFormated(LocalDate lDate);

	List<Entry> getAllEntriesByDate(LocalDate lDate);

}
```

```
package org.fovvox.usagesense.model.service;

import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;
import java.time.temporal.TemporalUnit;
import java.util.Date;
import java.util.List;
import java.util.TimeZone;

import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.fovvox.usagesense.model.repository.EntryRepository;
import org.fovvox.usagesense.tracker.ProcessInfo;
import org.springframework.stereotype.Service;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Service
public class EntryServiceImpl implements EntryService {
	private final EntryRepository entryRepository;
	private final ProgrammService programmService;

	private final SimpleDateFormat formatter;
```

```
	private Entry lastRegisteredEntry;

	public EntryServiceImpl(EntryRepository entryRepository, ProgrammService programmService) {
		this.entryRepository = entryRepository;
		this.programmService = programmService;

		this.formatter = new SimpleDateFormat("HH:mm:ss");
		this.formatter.setTimeZone(TimeZone.getTimeZone("UTC"));
	}

	@Override
	public void updateLast() {
		lastRegisteredEntry.setDuration(lastRegisteredEntry.getDuration() + 1);
		if (lastRegisteredEntry.getDuration() % 5 == 0) {
			entryRepository.save(lastRegisteredEntry);
			log.info("Updated entry: " +
					lastRegisteredEntry.getId() +
					"| In use " +
					formatter.format(new Date(lastRegisteredEntry.getDuration() *
1000L)));
		}
		LocalDateTime date = LocalDateTime.from(lastRegisteredEntry.getStartTime());
		date = date.plus(lastRegisteredEntry.getDuration(), ChronoUnit.SECONDS);
		if (lastRegisteredEntry.getStartTime().getDayOfMonth() != date.getDayOfMonth()) {
			registerNewEntry(
					ProcessInfo.builder()

	.executablePath(lastRegisteredEntry.getProgramm().getExecutablePath())
						.windowName(lastRegisteredEntry.getWindowName())
						.build());
		}
	}

	@Override
	public void registerNewEntry(ProcessInfo pInfo) {
		if (lastRegisteredEntry != null) {
			entryRepository.save(lastRegisteredEntry);
			log.info("Updated entry: " +
			lastRegisteredEntry.getId() +
			"| In use " +
			formatter.format(new Date(lastRegisteredEntry.getDuration() * 1000L)));
		}
		Programm programm = programmService.findByExePath(pInfo.getExecutablePath());
		if (programm == null) {
			programm = programmService.registerNewProgramm(pInfo.getExecutablePath());
		}
		Entry entry = Entry.builder()
				.windowName(pInfo.getWindowName())
				.startTime(LocalDateTime.now())
				.duration(1)
				.programm(programm)
				.build();
		lastRegisteredEntry = entryRepository.save(entry);
		log.info("Registered new activity : " + lastRegisteredEntry.getId() + " " +
lastRegisteredEntry.getWindowName());
	}
```

```
	@Override
	public List<Entry> getAllEntriesByYear(int year) {
		return entryRepository.findAllEnriesByYear(year);
	}


	@Override
	public List<Entry> getAllEntriesByDate(LocalDate lDate) {

		return null;
	}


	@Override
	public List<Entry> getAllEntriesByDateChartFormated(LocalDate lDate) {
		// TODO Auto-generated method stub
		return null;
	}




}
```

```
package org.fovvox.usagesense.model.service;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Map;

import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;

public interface ProgrammService {

	Programm findByExePath(String executablePath);

	Programm registerNewProgramm(String executablePath);


	Map<Programm, List<Entry>> getAllProgrammEntriesbyDay(LocalDateTime date);

	Map<Programm, List<Entry>> getAllProgrammEntriesbyYear(int year);

	Programm getByID(long id);

	Map<Programm, Integer>  getPopularPrograms(long id);

	Programm findByName(String name);
}
```

```
package org.fovvox.usagesense.model.service;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.fovvox.usagesense.model.repository.EntryRepository;
import org.fovvox.usagesense.model.repository.ProgrammRepository;
import org.fovvox.usagesense.tracker.OSHelper;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.PropertySource;
import org.springframework.stereotype.Service;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Service
@PropertySource("classpath:application.properties")
public class ProgrammServiceImpl implements ProgrammService {
	private final ProgrammRepository programmRepository;
	private final EntryRepository entryRepository;
	private final OSHelper osHelper;

	@Value(value = "${user.data.folder}")
	private String userDataFolder;

	public ProgrammServiceImpl(ProgrammRepository programmRepository, OSHelper osHelper,
EntryRepository entryRepository) {
		this.programmRepository = programmRepository;
		this.entryRepository = entryRepository;
		this.osHelper = osHelper;
	}

	@Override
	public Programm findByExePath(String executablePath) {
		return programmRepository.findOneByExecutablePath(executablePath).orElse(null);
	}

	@Override
	public Programm findByName(String name) {
		return programmRepository.findOneByName(name).orElse(null);
	}

	@Override
	public Programm registerNewProgramm(String executablePath) {
//		File exe = new File(executablePath);
//		BufferedImage logo = JIconExtract.getIconForFile(128, 128, executablePath);
//		File logoFile = new File(
//				System.getProperty("user.home") +
//				"\\" +
//				userDataFolder +
//				"\\icons\\" +
//				exe.getName() +
```

```
//                                      ".png");
//                  try {
//                          ImageIO.write(logo, "png", logoFile);
//                  } catch (IOException e) {
//                          log.warn("Can't save logo : " + e.getMessage());
//                  }
                    Programm programm = Programm.builder()
                                    .executablePath(executablePath)
                                    .name(osHelper.getExeName(executablePath))
                                    .company(osHelper.getExeCompany(executablePath))
//                                  .logoPath(logoFile.getAbsolutePath())
                                    .displayColor(osHelper.generateHexColor(executablePath)).build();

                    Programm savedProgramm = programmRepository.save(programm);
                    log.info("Registered new program : " + savedProgramm.getId() + " " +
savedProgramm.getName());
                    return savedProgramm;
            }

            @Override
            public Map<Programm, List<Entry>> getAllProgrammEntriesbyDay(LocalDateTime date) {
                    List<Programm> programms = programmRepository.findAllByEntriesStartTime(date);
                    Map<Programm, List<Entry>> map = new HashMap<>();
                    programms.forEach(programm -> {
                            List<Entry> entries = entryRepository.findAllByStartTimeAndProgramm(date,
programm);
                            map.put(programm, entries);
                    });
                    return map;
            }

            @Override
            public Map<Programm, List<Entry>> getAllProgrammEntriesbyYear(int year) {
                    List<Programm> programms = programmRepository.findAllByYear(year);
                    Map<Programm, List<Entry>> map = new HashMap<>();
                    programms.forEach(programm -> {
                            List<Entry> entries = entryRepository.findAllByYearAndProgramm(year,
programm);
                            map.put(programm, entries);
                    });
                    return map;
            }

            @Override
            public Programm getByID(long id) {
                    return programmRepository.findById(id).orElseThrow(() -> new
EntryNotFoundException("Cant find program wiht id:" + id));
            }

            @Override
            public Map<Programm, Integer> getPopularPrograms(long id) {
                    List<Programm> allProgramms = programmRepository.findAll();
                    Map<Programm, Integer> popularPrograms = new HashMap<>();
                    Programm leastPopular = null;
                    for (Programm programm : allProgramms) {
                            int usage = programm.getEntryList().stream().mapToInt(entry ->
entry.getDuration()).sum();
```

```
                        if (programm.getId() == id) {
                                popularPrograms.put(programm, usage);
                                continue;
                        }

                        if (popularPrograms.size() < 4) {
                                popularPrograms.put(programm, usage);
                                if (leastPopular == null || popularPrograms.get(leastPopular) > usage) {
                                        leastPopular = programm;
                                }
                                continue;
                        }

                        if (usage > popularPrograms.get(leastPopular)) {
                                popularPrograms.remove(leastPopular);
                                popularPrograms.put(programm, usage);
                                leastPopular = programm;
                        }

                }
                int otherProgramsUsage = 0;

                for (Programm programm : allProgramms) {
                        if (popularPrograms.containsKey(programm)) {
                                continue;
                        }
                        otherProgramsUsage += programm.getEntryList().stream().mapToInt(entry ->
entry.getDuration()).sum();
                }

                Programm otherSum = Programm.builder().id(-
1L).displayColor("#374151").name("Others").build();
                popularPrograms.put(otherSum, otherProgramsUsage);

                return popularPrograms;
        }

}



package org.fovvox.usagesense.tracker;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Random;

import org.springframework.stereotype.Component;

import com.sun.jna.Native;
import com.sun.jna.platform.win32.User32;
import com.sun.jna.platform.win32.Kernel32;
```

```
import com.sun.jna.platform.win32.WinDef.HWND;
import com.sun.jna.platform.win32.WinUser.LASTINPUTINFO;
import com.sun.jna.ptr.IntByReference;

import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j
public class OSHelper {

	private final Random random = new Random();

	public ProcessInfo getActiveWindowInfo() {
		HWND fg = User32.INSTANCE.GetForegroundWindow();

		int pid = getPIDofWindow(fg);
		ProcessHandle.Info pInfo = getProcessInfo(pid);
		if (pInfo == null) {
			log.warn("Can't get wondow information of pid: " + pid);
			return null;
		}
		char[] buffer = new char[User32.INSTANCE.GetWindowTextLength(fg) + 1];
		User32.INSTANCE.GetWindowText(fg, buffer, buffer.length);
		String fxWindowName = Native.toString(buffer);
		return
ProcessInfo.builder().pid(pid).windowName(fxWindowName).executablePath(pInfo.command().orElse(""))
				.build();
	}

	public String getExeName(String path) {
		String exeName = readFileTooltip(path, "File description");
		if (exeName.isEmpty()) {
			exeName = new File(path).getName();
		}
		return exeName;
	}

	public String getExeCompany(String path) {
		return readFileTooltip(path, "Company");
	}

	public String generateHexColor(String input) {
		try {
			// Create a MessageDigest instance for MD5
			MessageDigest md = MessageDigest.getInstance("MD5");
			md.update(input.getBytes());

			// Get the hash bytes
			byte[] byteData = md.digest();

			// Convert bytes to a hexadecimal format
			StringBuilder hexString = new StringBuilder();
			for (byte b : byteData) {
				String hex = Integer.toHexString(0xff & b);
				if (hex.length() == 1) {
					hexString.append('0');
				}
```

```
                        hexString.append(hex);
                    }

                    // Return the first 6 characters of the hex string as the color code
                    return "#" + hexString.toString().substring(0, 6);

                } catch (NoSuchAlgorithmException e) {
                    log.info("Cant generate color for " + input + " : " + e.getMessage());
                    return "#" + generateRandomHex();
                }
            }

            public boolean isUserAway() {
                LASTINPUTINFO lastInputInfo = new User32.LASTINPUTINFO();
                if (!User32.INSTANCE.GetLastInputInfo(lastInputInfo)) {
                    return false;
                }
                long currentTime = Kernel32.INSTANCE.GetTickCount64();

                long idleTime = currentTime - lastInputInfo.dwTime;

                return idleTime > 600_000;
            }

            private String generateRandomHex() {
                int randomInt = random.nextInt(0xFFFFFF + 1);

                String hexColor = Integer.toHexString(randomInt).toUpperCase();
                while (hexColor.length() < 6) {
                    hexColor = "0" + hexColor;
                }
                return hexColor;
            }

            private ProcessHandle.Info getProcessInfo(int pid) {
                ProcessHandle processHandle = ProcessHandle.of(pid).orElse(null);
                if (processHandle == null) {
                    return null;
                }
                return processHandle.info();
            }

            private int getPIDofWindow(HWND window) {
                IntByReference procId = new IntByReference();
                User32.INSTANCE.GetWindowThreadProcessId(window, procId);
                return procId.getValue();
            }

            private String readFileTooltip(String path, String tooltip) {
                String data = "";
                try {
                    Process p =
Runtime.getRuntime().exec("C:\\Programing\\UsageSense\\UsageSense\\lib\\toolTipInfo.bat " + "\"" + path
+ "\"");
                    BufferedReader input = new BufferedReader(new
InputStreamReader(p.getInputStream()));
                    String line;
```

```
                        while ((line = input.readLine()) != null) {
                                if (line.contains(tooltip + ":"))
                                        data = line.split(":")[1].trim();
                        }
                        input.close();
                } catch (IOException e) {
                        log.warn("Can't read file details : " + e.getMessage());
                }
                return data;
        }
}
```

```
package org.fovvox.usagesense.tracker;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ProcessInfo {
        private int pid = -1;
        private String windowName = "";
        private String executablePath = "";
}
```

```
package org.fovvox.usagesense.tracker;

import org.fovvox.usagesense.model.service.EntryService;
import org.springframework.stereotype.Component;

import jakarta.annotation.PostConstruct;
import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j
public class Tracker {
        private final OSHelper osHelper;
        private final EntryService entryService;

        private Thread runningThread;
        private ProcessInfo lastProcessInfo = new ProcessInfo();

        public Tracker(OSHelper osHelper, EntryService entryService) {
                this.osHelper = osHelper;
                this.entryService = entryService;
        }


        @PostConstruct
        public void start() {
```

```
                if (runningThread != null || (runningThread != null && runningThread.isInterrupted())) {
                        return;
                }
                Thread t = new Thread(() -> {
                        while (true) {
                                try {
                                        Thread.sleep(1000);
                                } catch (InterruptedException e) {
                                        log.error("Tracking thread has crushed: " + e.getMessage());
                                }
                                if (osHelper.isUserAway()) {
                                        continue;
                                }
                                registerActivity();
                        }
                });
                t.setName("Tacking thread");
                runningThread = t;
                t.start();
                log.info("Started tracker thread");
        }

        public void stop() {
                if (!runningThread.isInterrupted()) {
                        return;
                }
                runningThread.interrupt();
        }

        private void registerActivity() {
                ProcessInfo pInfo = osHelper.getActiveWindowInfo();
                if (pInfo == null) {
                        return;
                }
                if (pInfo.getPid() == lastProcessInfo.getPid() &&
pInfo.getWindowName().equals(lastProcessInfo.getWindowName())) {
                        entryService.updateLast();
                        return;
                }
                entryService.registerNewEntry(pInfo);
                lastProcessInfo = pInfo;
        }


}


package org.fovvox.usagesense.ui;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@org.springframework.stereotype.Controller
public class IndexController {

        @GetMapping({"", "/", "index", "index.html"})
        public String index() {
```

```
		return "index";
	}


}
```

```
package org.fovvox.usagesense.ui;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class StatisticController {


	@GetMapping({"/stats"})
	public ModelAndView stats(ModelMap model) {
		String date = LocalDate.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
		return new ModelAndView("redirect:/stats/day/" + date, model);
	}


	@GetMapping({"/stats/day/{date}"})
	public String dailyStats(@PathVariable String date) {
		return "dailyStats";
	}

	@GetMapping({"/stats/programm/{programmID}"})
	public String programStats(@PathVariable String programmID) {
		return "programStats";
	}
}
```

```
package org.fovvox.usagesense.ui;

import java.awt.AWTException;
import java.awt.Desktop;
import java.awt.Image;
import java.awt.MenuItem;
import java.awt.PopupMenu;
import java.awt.SystemTray;
import java.awt.Toolkit;
import java.awt.TrayIcon;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
```

```
import javax.swing.JOptionPane;
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class TrayController {

	private TrayIcon trayIcon;

	public TrayController() throws AWTException {
		System.setProperty("java.awt.headless", "false");
		if (!SystemTray.isSupported()) {
			log.error("System tray is not supported!");
			return;
		}
		Image image =
Toolkit.getDefaultToolkit().getImage("C:\\Users\\fovvox\\UsageSense\\icon.png");

		PopupMenu trayPopupMenu = new PopupMenu();

		MenuItem openItem = new MenuItem("Open");
		openItem.addActionListener(e -> openApplication());

		MenuItem closeItem = new MenuItem("Close");
		closeItem.addActionListener(e -> closeApplication());

		TrayIcon trayIcon = new TrayIcon(image, "UsageSense", trayPopupMenu);
		// adjust to default size as per system recommendation
		trayIcon.setImageAutoSize(true);

		trayIcon.addActionListener(e -> openApplication());

		trayIcon.setPopupMenu(new java.awt.PopupMenu());
		trayIcon.getPopupMenu().add(openItem);
		trayIcon.getPopupMenu().add(closeItem);

		this.trayIcon = trayIcon;
		SystemTray.getSystemTray().add(trayIcon);

	}

	public void delete() {
		SystemTray.getSystemTray().remove(trayIcon);
	}

	private void openApplication() {
		try {
			// Open link in default browser
			Desktop.getDesktop().browse(new URI("http://localhost:39999"));
		} catch (IOException | URISyntaxException e) {
			e.printStackTrace();
			JOptionPane.showMessageDialog(null, "Failed to open application");
		}
	}

	private void closeApplication() {
		try {
```

```
                        URL url = new URL("http://localhost:39999/actuator/shutdown");
                        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                        connection.setRequestMethod("POST");
                        connection.setDoOutput(true);
                        connection.connect();

                        int responseCode = connection.getResponseCode();
                        if (responseCode == 200) {
                                JOptionPane.showMessageDialog(null,
                                                "Shutdown initiated. Please wait for confirmation in the
application window.");
                        } else {
                                JOptionPane.showMessageDialog(null, "Failed to trigger shutdown. Status
code: " + responseCode);
                        }
                        connection.disconnect();
                } catch (Exception e) {
                        e.printStackTrace();
                        JOptionPane.showMessageDialog(null, "Error triggering shutdown: " +
e.getMessage());
                }
        }

}
```

```
package org.fovvox.usagesense.api;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;

import org.fovvox.usagesense.api.dto.ProgramHourlyUsageDTO;
import org.fovvox.usagesense.api.dto.ProgrammUsageDTO;
import org.fovvox.usagesense.api.dto.mapper.ProgramHourlyUsageMapper;
import org.fovvox.usagesense.api.dto.mapper.ProgrammUsageMapper;
import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.fovvox.usagesense.model.service.EntryNotFoundException;
import org.fovvox.usagesense.model.service.ProgrammService;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.server.ResponseStatusException;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@RestController
@RequestMapping(ProgrammController.BASE_URL)
public class ProgrammController {
        public static final String BASE_URL = "/api/programm/";
```

```
	private final ProgrammService programmService;
	private final ProgramHourlyUsageMapper programHourlyUsageMapper;
	private final ProgrammUsageMapper programmUsageMapper;

	public ProgrammController(ProgramHourlyUsageMapper programHourlyUsageMapper,
ProgrammService programmService, ProgrammUsageMapper programmUsageMapper) {
		this.programmService = programmService;
		this.programHourlyUsageMapper = programHourlyUsageMapper;
		this.programmUsageMapper = programmUsageMapper;
		}

	@GetMapping("all/{date}")
	public List<ProgramHourlyUsageDTO> getProgramHourlyUsage(@PathVariable String date) {
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
		LocalDate lDate = LocalDate.parse(date, formatter);

		return
programHourlyUsageMapper.programmEntiesMapToProgramHourlyUsageDTOs(programmService.getAllP
rogrammEntriesbyDay(lDate.atStartOfDay()));
	}

	@GetMapping("all/year/{year}")
	public List<ProgrammUsageDTO> getProgramYearlyUsage(@PathVariable String year) {
		int y = Integer.parseInt(year);

		return
programmUsageMapper.programmEntriesMapToProgrammUsageDto(programmService.getAllProgrammEn
triesbyYear(y));
	}

	@GetMapping("{programmID}")
	public Programm getProgramm(@PathVariable String programmID) {
		try {
			Programm programm = programmService.getByID(Long.parseLong(programmID));
			programm.getEntryList().forEach(e -> e.setProgramm(null));
			return programm;
		} catch (EntryNotFoundException e) {
			log.warn(e.getMessage());
			throw new ResponseStatusException(HttpStatus.NOT_FOUND);
		}
	}

	@GetMapping("{programmID}/pie")
	public List<ProgrammUsageDTO> getProgramPieData(@PathVariable String programmID) {
		long id = Long.parseLong(programmID);

		return
programmUsageMapper.programmUsageMapToProgrammUsageDto(programmService.getPopularPrograms
(id));
	}

}


package org.fovvox.usagesense.api;

import java.time.LocalDate;
```

```
import java.time.format.DateTimeFormatter;
import java.util.List;

import org.fovvox.usagesense.api.dto.DailyUsageListDTO;
import org.fovvox.usagesense.api.dto.mapper.DailyUsageMapper;
import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.service.EntryService;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping(EntryController.BASE_URL)
public class EntryController {
	public static final String BASE_URL = "/api/entry/";

	private final EntryService entryService;
	private final DailyUsageMapper dailyUsageMapper;

	public EntryController(EntryService entryService, DailyUsageMapper dailyUsageMapper) {
		this.entryService = entryService;
		this.dailyUsageMapper = dailyUsageMapper;
	}

	@GetMapping("year/{year}")
	public DailyUsageListDTO getUsagesByYear(@PathVariable String year) {
		return
dailyUsageMapper.entriesListToDailyUsageListDTO(entryService.getAllEntriesByYear(Integer.parseInt(ye
ar)));
	}

	@GetMapping("day/all/{date}")
	public List<Entry> getAllEntriesForADay(@PathVariable String date) {
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
		LocalDate lDate = LocalDate.parse(date, formatter);

		return entryService.getAllEntriesByDateChartFormated(lDate);
	}

}




package org.fovvox.usagesense.api.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;

@Data
@AllArgsConstructor
@Builder
public class ProgrammUsageDTO {
	private long id;
```

```
	private String name;
	private String color;
	private int usage;
}
```

```
package org.fovvox.usagesense.api.dto;

import java.util.Map;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;

@Data
@AllArgsConstructor
@Builder
public class ProgramHourlyUsageDTO {
	private long id;
	private String name;
	private String iconPath;
	private String color;
	private Map<Integer, Double> hourlyUse;
}
```

```
package org.fovvox.usagesense.api.dto;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class DailyUsageListDTO {
	private List<DailyUsageDTO> dailyUsageDTOs;
}
```

```
package org.fovvox.usagesense.api.dto;

import java.time.LocalDate;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class DailyUsageDTO {
	private LocalDate date;
	private double hours;
}
```

```
package org.fovvox.usagesense.api.dto.mapper;

import java.util.List;
import java.util.stream.Collectors;

import org.fovvox.usagesense.api.dto.DailyUsageDTO;
import org.fovvox.usagesense.api.dto.DailyUsageListDTO;
import org.fovvox.usagesense.model.domain.Entry;
import org.springframework.stereotype.Component;

@Component
public class DailyUsageMapper {

	public DailyUsageListDTO entriesListToDailyUsageListDTO(List<Entry> entries) {
		return new DailyUsageListDTO(entries.stream()
		.collect(
			Collectors.groupingBy(e -> e.getStartTime().toLocalDate()))
			.entrySet()
				.stream()
				.map(t ->
					new DailyUsageDTO(
							t.getKey(),
							t.getValue().stream()
								.mapToInt(Entry::getDuration).sum() /
3600.))
				.toList());
	}
}
```

```
package org.fovvox.usagesense.api.dto.mapper;

import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.fovvox.usagesense.api.dto.ProgramHourlyUsageDTO;
import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.springframework.stereotype.Component;

import lombok.extern.slf4j.Slf4j;

@Component
public class ProgramHourlyUsageMapper {
```

```
	public List<ProgramHourlyUsageDTO>
programmEntiesMapToProgramHourlyUsageDTOs(Map<Programm, List<Entry>> map) {
		List<ProgramHourlyUsageDTO> list = new ArrayList<>();
		map.entrySet().stream()
//			.sorted((e1, e2) -> e2.getValue().get(e2.getValue().size() - 1).getStartTime()
//				.compareTo(e1.getValue().get(e1.getValue().size() - 1).getStartTime()))
			.forEach(e -> {
				List<Entry> entries = splitEntries(e.getValue());
				Map<Integer, Double> hourlyUsage = new HashMap<>();
				entries.stream().sorted((o1, o2) ->
o1.getStartTime().compareTo(o2.getStartTime())).forEach(t -> {
					int hour = t.getStartTime().getHour();
					int durationSeconds = t.getDuration();
					double durationMinutes = durationSeconds / 60.0;

					hourlyUsage.put(hour, hourlyUsage.getOrDefault(hour, 0.0)
+ durationMinutes);
					for (int i = 0; i < 24; i++) {
						hourlyUsage.put(i,
Math.round(hourlyUsage.getOrDefault(i, 0.0) * 10) / 10.);
					}

				});

	list.add(ProgramHourlyUsageDTO.builder().id(e.getKey().getId()).name(e.getKey().getName())

	.iconPath(e.getKey().getLogoPath()).color(e.getKey().getDisplayColor())
							.hourlyUse(hourlyUsage).build());
			});
		list.sort((o1, o2) -> {
			double o1Sum = o1.getHourlyUse().entrySet().stream().mapToDouble((value) -> {
				return (Double) value.getValue();
			}).sum();
			double o2Sum = o2.getHourlyUse().entrySet().stream().mapToDouble((value) -> {
				return (Double) value.getValue();
			}).sum();
			return Double.compare(o1Sum, o2Sum);
		});

		return list;
	}

	private List<Entry> splitEntries(List<Entry> entries) {
		List<Entry> result = new ArrayList<>();

		for (Entry entry : entries) {
			LocalDateTime endTime = entry.getStartTime().plusSeconds(entry.getDuration());

			if (entry.getStartTime().getHour() == endTime.getHour()) {
				result.add(entry);
				continue;
			}

			int firtsHourDuration = (int) ChronoUnit.SECONDS.between(entry.getStartTime(),
LocalDateTime
					.of(entry.getStartTime().toLocalDate(),
```

```
LocalTime.of(entry.getStartTime().getHour(), 59, 59)));

		result.add(Entry.builder().startTime(entry.getStartTime()).duration(firtsHourDuration).build());
				int durationLeft = entry.getDuration() - firtsHourDuration;
				int endHour = endTime.getHour();
				if (endHour == 0) {
					endHour = 24;
				}
				for (int i = 1; i <= endHour - entry.getStartTime().getHour(); i++) {
					LocalDateTime startTime =
LocalDateTime.of(entry.getStartTime().toLocalDate(),
							LocalTime.of(entry.getStartTime().plusHours(i).getHour(),
0, 0));
					Entry entryToAdd = new Entry();
					entryToAdd.setStartTime(startTime);
					;
					if (durationLeft - 60 * 60 > 0) {
						entryToAdd.setDuration(60 * 60);
						;
						durationLeft -= 60 * 60;
					} else {
						entryToAdd.setDuration(durationLeft);
						;
					}

					result.add(entryToAdd);
				}

			}
			return result;
		}

}


package org.fovvox.usagesense.api.dto.mapper;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import org.fovvox.usagesense.api.dto.ProgrammUsageDTO;
import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.springframework.stereotype.Component;
import org.springframework.util.comparator.Comparators;

@Component
public class ProgrammUsageMapper {

		public List<ProgrammUsageDTO> programmEntriesMapToProgrammUsageDto(Map<Programm,
List<Entry>> map) {
			List<ProgrammUsageDTO> result = new ArrayList<ProgrammUsageDTO>();
			map.forEach((prog, useList) -> {
				int usage = useList.stream().mapToInt(e -> e.getDuration()).sum();
				result.add(ProgrammUsageDTO.builder()
					.id(prog.getId())
```

```
                                        .name(prog.getName())
                                        .color(prog.getDisplayColor())
                                        .usage(usage)
                                        .build());
                });
                result.sort((o1, o2) -> Integer.compare(o2.getUsage(), o1.getUsage()));
                return result;
        }


        public List<ProgrammUsageDTO> programmUsageMapToProgrammUsageDto(Map<Programm,
Integer>  map) {
                List<ProgrammUsageDTO> result = new ArrayList<ProgrammUsageDTO>();
                map.forEach((prog, usage) -> {
                        result.add(ProgrammUsageDTO.builder()
                                        .id(prog.getId())
                                        .name(prog.getName())
                                        .color(prog.getDisplayColor())
                                        .usage(usage)
                                        .build());
                });


                result.sort((o1, o2) -> Integer.compare(o2.getUsage(), o1.getUsage()));
                ProgrammUsageDTO dto = result.stream().filter(t -> t.getId() == -1).findFirst().get();
                result.remove(dto);
                result.add(dto);
                return result;
        }
}


package org.fovvox.usagesense.api;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;

import org.fovvox.usagesense.api.dto.ProgramHourlyUsageDTO;
import org.fovvox.usagesense.api.dto.ProgrammUsageDTO;
import org.fovvox.usagesense.api.dto.mapper.ProgramHourlyUsageMapper;
import org.fovvox.usagesense.api.dto.mapper.ProgrammUsageMapper;
import org.fovvox.usagesense.model.domain.Entry;
import org.fovvox.usagesense.model.domain.Programm;
import org.fovvox.usagesense.model.service.EntryNotFoundException;
import org.fovvox.usagesense.model.service.ProgrammService;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.server.ResponseStatusException;

import lombok.extern.slf4j.Slf4j;
```

```
@Slf4j
@RestController
@RequestMapping(ProgrammController.BASE_URL)
public class ProgrammController {
	public static final String BASE_URL = "/api/programm/";

	private final ProgrammService programmService;
	private final ProgramHourlyUsageMapper programHourlyUsageMapper;
	private final ProgrammUsageMapper programmUsageMapper;

	public ProgrammController(ProgramHourlyUsageMapper programHourlyUsageMapper,
ProgrammService programmService, ProgrammUsageMapper programmUsageMapper) {
		this.programmService = programmService;
		this.programHourlyUsageMapper = programHourlyUsageMapper;
		this.programmUsageMapper = programmUsageMapper;
		}

	@GetMapping("all/{date}")
	public List<ProgramHourlyUsageDTO> getProgramHourlyUsage(@PathVariable String date) {
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
		LocalDate lDate = LocalDate.parse(date, formatter);

		return
programHourlyUsageMapper.programmEntiesMapToProgramHourlyUsageDTOs(programmService.getAllP
rogrammEntriesbyDay(lDate.atStartOfDay()));
	}

	@GetMapping("all/year/{year}")
	public List<ProgrammUsageDTO> getProgramYearlyUsage(@PathVariable String year) {
		int y = Integer.parseInt(year);

		return
programmUsageMapper.programmEntriesMapToProgrammUsageDto(programmService.getAllProgrammEn
triesbyYear(y));
	}

	@GetMapping("{programmID}")
	public Programm getProgramm(@PathVariable String programmID) {
		try {
			Programm programm = programmService.getByID(Long.parseLong(programmID));
			programm.getEntryList().forEach(e -> e.setProgramm(null));
			return programm;
		} catch (EntryNotFoundException e) {
			log.warn(e.getMessage());
			throw new ResponseStatusException(HttpStatus.NOT_FOUND);
		}
	}

	@GetMapping("{programmID}/pie")
	public List<ProgrammUsageDTO> getProgramPieData(@PathVariable String programmID) {
		long id = Long.parseLong(programmID);

		return
programmUsageMapper.programmUsageMapToProgrammUsageDto(programmService.getPopularPrograms
(id));
	}
```

```
}

<!DOCTYPE HTML>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
<link rel="icon" type="image/x-icon" href="icons/favicon.ico">
<title>UsageSense</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<!-- this is where we refer the css file from src/main/resources/static/main.css -->
<link th:href="@{/css/main.css}" rel="stylesheet" />
</head>
<body class="pt-2 pl-[8vw] pr-[8vw] ">
        <div class="flex flex-row">
                <div class=" flex flex-col mr-2 bg-clip-border rounded-xl bg-white text-gray-700 h-[98vh]
min-h-[30rem] w-full max-w-[20rem] p-4 shadow-xl shadow-gray-500">
                        <div class="p-4 mb-2">
                                <h1
                                        class="block font-sans text-2xl antialiased font-semibold leading-
snug tracking-normal text-gray-900">UsageSense</h5>
                        </div>
                        <nav
                                class="flex flex-col gap-1 min-w-[240px] p-2 font-sans text-base font-
normal text-gray-700">
                                <a href="/">
                                        <div role="button" tabindex="0"
                                        class="flex items-center w-full p-3 leading-tight text-blue-900
transition-all rounded-lg group text-start bg-blue-50 bg-opacity-80">
                                        <div class="grid mr-4 place-items-center">
                                                <img src="icons/home.png" alt="home" class="h-8">
                                        </div>
                                        Огляд
                                </div>
                                </a>
                                <a href="/stats">
                                        <div role="button" tabindex="0"
                                        class="flex items-center w-full p-3 leading-tight transition-all
rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                                        <div class="grid mr-4 place-items-center">
                                                <img src="icons/stats.png" alt="home" class="h-8">
                                        </div>
                                        Щоденна статиска
                                </div>
                                </a>

                                <div role="button" tabindex="0"
                                        class="flex items-center w-full p-3 leading-tight transition-all
rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                                        <div class="grid mr-4 place-items-center">
                                                <img src="icons/exit.png" alt="home" class="h-8">
                                        </div>
                                        Вихід
                                </div>
                        </nav>
                </div>
                <div class="relative flex flex-col bg-clip-border rounded-xl bg-slate-50 text-gray-700  min-
h-[30rem] h-[98vh] w-[calc(99vw-20rem)] p-7 shadow-xl shadow-gray-500">
                        <h2 class="block mb-5 font-sans text-2xl antialiased font-semibold leading-snug
```

```
tracking-normal text-gray-900">Огляд</h2>
                    <div class="container shadow-xl rounded-xl bg-slate-100">
                        <div class="container flex items-center font-sans text-2xl font-normal
leading-none ">
                            <div id="previousButton" role="button" class="p-4 m-auto
hover:bg-blue-100 hover:bg-opacity-80 hover:text-blue-900" >
                                <img src="icons/arrow.png" alt="arrow" class="rotate-180
h-7">
                            </div>
                            <div id="yearDisplay" class="pl-7 pr-7"></div>
                            <div id="nextButton" role="button" class="p-4 m-auto hover:bg-
blue-100 hover:bg-opacity-80 hover:text-blue-900" >
                                <img src="icons/arrow.png" alt="arrow" class="h-7">
                            </div>
                        </div>
                        <div class="p-3 m-3 font-sans leading-tight bg-slate-200 rounded-xl">
                            <table id="heatmap" class="">
                                <thead>
                                    <tr id="headerRow">
                                        <td></td>
                                        <td>
                                            <span >Січ</span>

                        </td>
                                        <td>
                                            <span>Лют</span>
                                        </td>
                                        <td>
                                            <span>Бер</span>
                                        </td>
                                        <td>
                                            <span>Кві</span>
                                        </td>
                                        <td>
                                            <span>Тра</span>
                                        </td>
                                        <td>
                                            <span>Чер</span>
                                        </td>
                                        <td>
                                            <span>Лип</span>
                                        </td>
                                        <td>
                                            <span>Сер</span>
                                        </td>
                                        <td>
                                            <span>Вер</span>
                                        </td>
                                        <td>
                                            <span>Жов</span>
                                        </td>
                                        <td>
                                            <span>Лис</span>
                                        </td>
                                        <td>
                                            <span>Гру</span>
                                        </td>
```

```
                                        </tr>
                                    </thead>
                                    <tbody id="rows">
                                        <tr id="mondays">
                                            <td  class="pr-4">Пн</td>
                                        </tr>
                                        <tr id="tuesdays">
                                            <td class="w-[1vw] h-[1vw]">Вт</td>
                                        </tr>
                                        <tr id="wednesdays">
                                            <td>Ср</td>
                                        </tr>
                                        <tr id="thursdays">
                                            <td>Чт</td>
                                        </tr>
                                        <tr id="fridays">
                                            <td>Пт</td>
                                        </tr>
                                        <tr id="saturdays">
                                            <td>Сб</td>
                                        </tr>
                                        <tr id="sundays">
                                            <td>Нд</td>
                                        </tr>
                                    </tbody>
                                </table>
                            </div>
                        </div>
                        <div class="container flex flex-col p-3 mt-2 shadow-xl rounded-xl bg-slate-100 ">
                            <div class="flex flex-row rounded-md bg-slate-200">
                                <div class="w-[6.8%]"></div>
                                <div class="w-[72%]">Назва</div>
                                <div > Час використання</div>
                            </div>
                            <h2 id="noData" class="hidden m-auto text-2xl font-semibold text-slate-
800">В цьому році не було зареєстровано ніякої активності</h2>
                            <div class=" overflow-y-scroll max-h-[30vh]">
                                <table id="programmTable" class="w-[100%]">
                                    <tr class="group">
                                        <td class="rounded-l-md group-hover:bg-blue-100
group-hover:bg-opacity-80">
                                            <a href=""><div class="h-[1vw] w-[1vw]
rounded-md m-auto" style="background-color: #838ee8;"></div></a>
                                        </td >
                                        <td class="group-hover:bg-blue-100 group-
hover:bg-opacity-80" tabindex="-1">
                                            <a href="">Visual Studio Code</a>
                                        </td>
                                        <td class="rounded-r-md group-hover:bg-blue-100
group-hover:bg-opacity-80" tabindex="-1">
                                            <a href="">00:50:00</a>
                                        </td>
                                    </tr>
                                </table>
                            </div>
```

```
                    </div>
                </div>
        </div>
        <div class="rounded-md bg-cyan-50 bg-cyan-100 bg-cyan-200 bg-cyan-300 bg-cyan-400 bg-cyan-500 bg-cyan-600"></div>
        <div id="tooltip" class="absolute hidden flex-col group-hover:flex translate-x-[-43.65%] -top-2 -translate-y-full w-40 px-2 py-1 bg-gray-700 rounded-lg text-center text-white text-sm after:content-['] after:absolute after:left-1/2 after:top-[100%] after:-translate-x-1/2 after:border-8 after:border-x-transparent after:border-b-transparent after:border-t-gray-700"></div>
        <script type="module" src="js/heatmap.js"></script>
</body>
</html>


<!DOCTYPE HTML>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
<link rel="icon" type="image/x-icon" href="icons/favicon.ico">
<title>UsageSense</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<!-- this is where we refer the css file from src/main/resources/static/main.css -->
<link th:href="@{/css/main.css}" rel="stylesheet" />
</head>
<body class="pt-2 pl-[8vw] pr-[8vw] ">
        <div class="flex flex-row">
                <div class=" flex flex-col mr-2 bg-clip-border rounded-xl bg-white text-gray-700 min-h-[30rem] w-full max-w-[20rem] p-4 shadow-xl shadow-gray-500">
                    <div class="p-4 mb-2">
                        <h1
                            class="block font-sans text-2xl antialiased font-semibold leading-snug tracking-normal text-gray-900">UsageSense</h5>
                    </div>
                    <nav
                        class="flex flex-col gap-1 min-w-[240px] p-2 font-sans text-base font-normal text-gray-700">
                        <a href="/">
                            <div role="button" tabindex="0"
                            class="flex items-center w-full p-3 leading-tight transition-all rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                            <div class="grid mr-4 place-items-center">
                                <img src="icons/home.png" alt="home" class="h-8">
                            </div>
                            Огляд
                        </div>
                        </a>
                        <a href="/stats">
                            <div role="button" tabindex="0"
                            class="flex items-center w-full p-3 leading-tight text-blue-900 transition-all rounded-lg group text-start bg-blue-50 bg-opacity-80 ">
                            <div class="grid mr-4 place-items-center">
                                <img src="icons/stats.png" alt="home" class="h-8">
                            </div>
                            Щоденна статиска
                        </div>
                        </a>
```

```
                        <div role="button" tabindex="0"
                            class="flex items-center w-full p-3 leading-tight transition-all
rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                            <div class="grid mr-4 place-items-center">
                                <img src="icons/exit.png" alt="home" class="h-8">
                            </div>
                            Вихід
                        </div>
                    </nav>
                </div>
                <div class="relative flex flex-col bg-clip-border rounded-xl bg-slate-50 text-gray-700  min-
h-[30rem]  w-[calc(99vw-20rem)] p-7 shadow-xl shadow-gray-500">
                    <h2 class="block mb-5 font-sans text-2xl antialiased font-semibold leading-snug
tracking-normal text-gray-900">Щоденна статистика</h2>
                    <div class="container p-3 shadow-xl rounded-xl bg-slate-100">
                        <div class="container flex items-center font-sans text-2xl font-normal
leading-none ">
                            <div id="previousButton" role="button" class="p-4 m-auto
hover:bg-blue-100 hover:bg-opacity-80 hover:text-blue-900" >
                                <img src="icons/arrow.png" alt="arrow" class="rotate-180
h-7">
                            </div>
                            <div id="dateDisply" class="pl-7 pr-7"></div>
                            <div id="nextButton" role="button" class="p-4 m-auto hover:bg-
blue-100 hover:bg-opacity-80 hover:text-blue-900" >
                                <img src="icons/arrow.png" alt="arrow" class="h-7">
                            </div>
                        </div>
                        <div class="flex flex-col items-center">
                            <h2 id="noData" class="hidden m-auto text-2xl font-semibold text-
slate-800">В цей день не було зареєстровано ніякої активності</h2>
                            <canvas id="hourlyChart"></canvas>
                        </div>
                    </div>
                    <div class="container p-3 mt-2 shadow-xl rounded-xl bg-slate-100">
                        <div class="flex flex-row rounded-md bg-slate-200">
                            <div class="w-[6.8%]"></div>
                            <div class="w-[72%]">Назва</div>
                            <div > Час використання</div>
                        </div>
                        <div class=" overflow-y-scroll max-h-[30vh]">
                            <table id="programmTable" class="w-[100%]">
                                <tr>
                                    <td>
                                        <div class="h-[1vw] w-[1vw] rounded-md
m-auto" style="background-color: #838ee8;"></div>
                                    </td>
                                    <td>
                                        Visual Studio Code
                                    </td>
                                    <td>
                                        00:50:00
                                    </td>
                                </tr>
                            </table>
                        </div>
```

```
                </div>
            </div>
                <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
                <script type="module" src="js/dailyCharts.js"></script>

</body>
</html>


<!DOCTYPE HTML>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">

<head>
    <link rel="icon" type="image/x-icon" href="icons/favicon.ico">
    <title>UsageSense</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <!-- this is where we refer the css file from src/main/resources/static/main.css -->
    <link th:href="@{/css/main.css}" rel="stylesheet" />
</head>

<body class="pt-2 pl-[8vw] pr-[8vw] ">
    <div class="flex flex-row">
        <div
            class=" flex flex-col mr-2 bg-clip-border rounded-xl bg-white text-gray-700 min-h-
[30rem] w-full max-w-[20rem] p-4 shadow-xl shadow-gray-500">
            <div class="p-4 mb-2">
                <h1
                    class="block font-sans text-2xl antialiased font-semibold leading-
snug tracking-normal text-gray-900">
                    UsageSense</h5>
            </div>
            <nav class="flex flex-col gap-1 min-w-[240px] p-2 font-sans text-base font-normal
text-gray-700">
                <a href="/">
                    <div role="button" tabindex="0"
                        class="flex items-center w-full p-3 leading-tight transition-
all rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                        <div class="grid mr-4 place-items-center">
                            <img src="icons/home.png" alt="home" class="h-
8">
                        </div>
                        Огляд
                    </div>
                </a>
                <a href="/stats">
                    <div role="button" tabindex="0"
                        class="flex items-center w-full p-3 leading-tight transition-
all rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                        <div class="grid mr-4 place-items-center">
                            <img src="icons/stats.png" alt="home" class="h-8">
                        </div>
                        Щоденна статиска
                    </div>
                </a>

                <div role="button" tabindex="0"
```

```
                                        class="flex items-center w-full p-3 leading-tight transition-all
rounded-lg text-start hover:bg-blue-50 hover:bg-opacity-80 hover:text-blue-900 ">
                                        <div class="grid mr-4 place-items-center">
                                                <img src="icons/exit.png" alt="home" class="h-8">
                                        </div>
                                        Вихід
                                </div>
                        </nav>
                </div>
                <div
                        class="relative flex flex-col bg-clip-border rounded-xl bg-slate-50 text-gray-700
min-h-[30rem]  w-[calc(99vw-20rem)] p-7 shadow-xl shadow-gray-500">
                        <h2 id="programmDisplay"
                                class="block mb-5 font-sans text-2xl antialiased font-semibold leading-snug
tracking-normal text-gray-900">
                                </h2>
                        <h3 id="comapnyDisplay"
                                class="block mb-5 font-sans text-xl antialiased font-semibold leading-snug
tracking-normal text-gray-800">
                        </h3>
                        <div class="container shadow-xl rounded-xl bg-slate-100">
                                <h3 class="p-3 m-auto text-xl antialiased text-center text-gray-
800">Використання за весь час</h3>
                                <div  class="container flex flex-row p-6 shadow-xl rounded-xl bg-slate-
100">
                                        <div class="p-5 m-auto bg-slate-200 rounded-xl">
                                                <div id="weeklyLabel" class="mb-5 text-lg text-center text-
gray-800 "></div>
                                                <canvas id="weeklyChart" class="min-h-[25vh]"></canvas>
                                        </div>
                                        <div class="p-5 m-auto bg-slate-200 rounded-xl">
                                                <div id="percentageLabel" class="mb-5 text-lg text-center
text-gray-800 "></div>
                                                <canvas id="percentageChart" class="min-h-
[25vh]"></canvas>
                                        </div>
                                </div>
                        </div>

                </div>
                <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>
                <script type="module" src="js/programmCharts.js"></script>

</body>

</html>


import * as rest from "./rest.js"

const heatmap = document.getElementById("heatmap")
const mondays = document.getElementById("mondays")
const tuesdays = document.getElementById("tuesdays")
const wednesdays = document.getElementById("wednesdays")
const thursdays = document.getElementById("thursdays")
const fridays = document.getElementById("fridays")
const saturdays = document.getElementById("saturdays")
```

```
const sundays = document.getElementById("sundays")
const yearDisplay = document.getElementById("yearDisplay")
const headerRow = document.getElementById("headerRow")
const rows = document.getElementById("rows")
const noData = document.getElementById("noData");

const MONDAY = 1;
const TUESDAY  = 2;
const WEDNESDAY = 3;
const THURSDAY = 4;
const FRIDAY = 5;
const SATURDAY = 6;
const SUNDAY = 0;


let year = new Date().getFullYear();




yearDisplay.textContent=year;

let renderTable = (year) => {
   rest.getUsagesByYear(year).then(
      (data) => {
         if (data.length == 0) {
            noData.classList.remove("hidden")
         } else {
            console.log("hidden");
            noData.classList.add("hidden")
         }
         const dates = getAllDatesInYear(year);
         let map = [];
         dates.forEach((date) => {
            date.setHours(date.getHours() - date.getTimezoneOffset() / 60)
            let match = data.find(item => item.date === date.toISOString().slice(0, 10));
            if (!match) {
               match = 0
            } else {
               match = Math.round(match.hours * 10) /10
            }
            map.push({ date: date, hours: match});
         });

         let lastMonth =  map[0].date.getMonth();
         let index = 0;
         fillEmpty(map[0].date);
         map.forEach(use => {
            if (lastMonth != use.date.getMonth()) {
               headerRow.cells[lastMonth + 1].colSpan = index;
               lastMonth = use.date.getMonth();
               index=0;
            }
            if (use.date.getDay() == MONDAY) {
               index++;
            }
            createCell(use.date,  use.hours)
```

```
            });
            headerRow.cells[12].colSpan = index;
        }
    )
}



let getAllDatesInYear= (year) => {
    const dates = [];
    const startDate = new Date(year, 0, 1); // January 1st of the given year
    const endDate = new Date(year + 1, 0, 0); // December 31st of the given year

    for (let date = startDate; date <= endDate; date.setDate(date.getDate() + 1)) {
        dates.push(new Date(date)); // Push a copy of the date object
    }

    return dates;
}

let createCell = (date, hours) => {
    let td = document.createElement("td");
    td.classList.add("m-auto")
    let cell = document.createElement("div");
    cell.classList.add("h-[1vw]")
    cell.classList.add("w-[1vw]")
    cell.classList.add("rounded-md")
    cell.classList.add("relative")
    cell.classList.add("group")
    if (hours == 0) {
        cell.classList.add("bg-cyan-50")
        cell.appendChild(createToolTip(date, hours));
    } else if (hours>0 && hours<=4) {
        cell.classList.add("bg-cyan-200")
        cell.appendChild(createToolTip(date, hours));
    } else if (hours>4 && hours<=8) {
        cell.classList.add("bg-cyan-300")
        cell.appendChild(createToolTip(date, hours));
    } else if (hours>8 && hours<=12) {
        cell.classList.add("bg-cyan-400")
        cell.appendChild(createToolTip(date, hours));
    } else if (hours>12 && hours<=16) {
        cell.classList.add("bg-cyan-500")
        cell.appendChild(createToolTip(date, hours));
    }else if (hours>16 && hours<=24) {
        cell.classList.add("bg-cyan-600")
        cell.appendChild(createToolTip(date, hours));
    }
    let ceellWrapper = document.createElement("a");
    ceellWrapper.href = `http://localhost:39999/stats/day/${date.toISOString().split('T')[0]}`
    ceellWrapper.appendChild(cell);
    td.appendChild(ceellWrapper);
    if (date.getDay() == SUNDAY) {
        rows.children[6].appendChild(td);
    } else {
        rows.children[date.getDay() - 1].appendChild(td)
    }
```

```
}

let createToolTip = (date, hours) =>{
    let tooltip = document.getElementById("tooltip").cloneNode(true);
    tooltip.id = "";
    let dateStr = document.createElement("div");
    dateStr.textContent = date.toDateString();
    let hoursStr = document.createElement("div");
    hoursStr.textContent = hours + " Годин витрачено";
    tooltip.appendChild(dateStr);
    tooltip.appendChild(hoursStr);
    return tooltip;
}

let fillEmpty = (date) => {
    if (date.getDay() == SUNDAY) {
        for (let index = 1; index < 7; index++) {
            let d = new Date(date);
            d.setDate(d.getDate() - index);
            createCell(d);
        }
        return
    }
    for (let index = 1; index < date.getDay(); index++) {
        let d = new Date(date);
        d.setDate(d.getDate() - index);
        createCell(d);
    }
};

let clearTable = () => {
    for (let i = 0; i < rows.children.length; i++) {
        const row = rows.children[i];
        const lable = row.children[0];
        row.innerHTML="";
        row.appendChild(lable);
    }
}

let programmTable = document.getElementById("programmTable");

function formatSeconds(seconds) {
  // Calculate hours, minutes, and remaining seconds (within 24 hours)
  const hours = Math.floor(seconds / 3600);
  const minutes = Math.floor((seconds % 3600) / 60);
  const secondsLeft = seconds % 60;

  // Format with leading zeros
  const formattedHours = hours.toString().padStart(2, '0');
  const formattedMinutes = minutes.toString().padStart(2, '0');
  const formattedSeconds = secondsLeft.toString().padStart(2, '0');
  console.log(seconds + " : " + `${formattedHours}:${formattedMinutes}:${formattedSeconds}`);
  return `${formattedHours}:${formattedMinutes}:${formattedSeconds}`;
  }

function generateTableRow(id, color, name, duration) {
    const tableRow = document.createElement('tr');
```

```
    tableRow.classList.add("group")
    const link = `http://localhost:39999/stats/programm/${id}`;

    const cell1 = document.createElement('td');
    cell1.classList.add("rounded-l-md","group-hover:bg-blue-100","group-hover:bg-opacity-80");
    const coloredSquare = document.createElement('div');
    coloredSquare.classList.add('h-[1vw]', 'w-[1vw]', 'rounded-md', 'm-auto');
    coloredSquare.style.backgroundColor = color;
    const a1 = document.createElement("a");
    a1.href=link;
    a1.appendChild(coloredSquare);
    cell1.appendChild(a1);
    const cell2 = document.createElement('td');
    cell2.classList.add("group-hover:bg-blue-100","group-hover:bg-opacity-80");
    cell2.tabIndex = -1;
    const a2 = document.createElement("a");
    a2.href=link;
    a2.textContent = name;
    cell2.appendChild(a2);

    const cell3 = document.createElement('td');
    cell3.classList.add("rounded-r-md","group-hover:bg-blue-100","group-hover:bg-opacity-80");
    cell3.tabIndex = -1;
    const a3 = document.createElement("a");
    a3.href=link;
    a3.textContent = formatSeconds(duration);
    cell3.appendChild(a3);

    tableRow.appendChild(cell1);
    tableRow.appendChild(cell2);
    tableRow.appendChild(cell3);




    return tableRow;
  }

async function renderProgrammTable(year) {
    programmTable.innerHTML = "";
    let data = await rest.getProgrammUsagesByYear(year)
    console.log(data);
    data.forEach(program => {
        programmTable.appendChild(generateTableRow(program.id, program.color, program.name,
program.usage));
      })
  }

function renderPage(year){
    renderTable(year);
    renderProgrammTable(year);
}



let previousYear = () => {
    year--;
```

```
    yearDisplay.textContent = year;
    clearTable();
    renderPage(year);
}

let nextYear = () => {
    year++;
    yearDisplay.textContent = year;
    clearTable();
    renderPage(year);
}

renderPage(year);

document.getElementById("previousButton").addEventListener("click" , previousYear);
document.getElementById("nextButton").addEventListener("click" ,nextYear);


export async function getUsagesByYear(year) {
  const response = await fetch(`http://localhost:39999/api/entry/year/${year}`);
  const dailyUsage = await response.json();
  return dailyUsage.dailyUsageDTOs;
}

export async function getProgrammUsagesByYear(year) {
  const response = await fetch(`http://localhost:39999/api/programm/all/year/${year}`);
  return await response.json();
}


let dateDisplay = document.getElementById("dateDisply")
let programmTable = document.getElementById("programmTable");




let extractDateFromUrl = (url) => {
    // Use regular expression to match the date format (YYYY-MM-DD)
    const regex = /\/day\/(\d{4}-\d{2}-\d{2})/;
    const match = url.match(regex);

    // Check if there's a match
    if (match) {
      // Extract the captured group (the date string)
      return match[1];
    } else {
      // Handle the case where the URL doesn't match the format
      console.error("Invalid URL format. Date not found.");
      return null;
    }
}

async function retriveData(date) {
  const response = await fetch(`http://localhost:39999/api/programm/all/${date.toISOString().split('T')[0]}`);
  const data = await response.json();
```

```
  return data;
}


let date = new Date(extractDateFromUrl(window.location.href));
dateDisplay.textContent = date.toLocaleDateString('uk')

function renderNothing() {
  document.getElementById("noData").classList.remove("hidden")
}


function parseDatasets(data) {
  let dbData = data;
  let datasets = []
  if (dbData.length == 0 ) {
    renderNothing();
  }
  console.log(dbData);

  dbData.forEach((programm) => {
    let datapoints = [];
    for (const hour in programm.hourlyUse) {
      if (Object.hasOwnProperty.call(programm.hourlyUse, hour)) {
        const element = programm.hourlyUse[hour];
        datapoints.push(element)
      }
    }
    datasets.push({
      label: programm.name,
      data: datapoints,
      borderColor: programm.color,
      backgroundColor:programm.color+"50",
      hiden:false,
      borderWidth:1,
      fill: '-1'
    })
  })
  datasets[0].fill = "origin"
  return datasets;
}

async function generateConfig(data) {
  const parsedDatasets = parseDatasets(data);

  return {
    type: 'line',
    data: {
      labels: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23],
      datasets: parsedDatasets
    },
    options: {

      scales: {
        yAxes: [
          {
            stacked:true,
```

```
          grid:{
            display:false
          }
        }
      ]
    }
  },
 }
}



function formatMinutesToTime(minutes) {
 if (isNaN(minutes) || minutes < 0) {
  return "Invalid time";
 }

 const seconds = Math.floor(minutes * 60);

 const hours = Math.floor(seconds / 3600);
 const remainingMinutes = Math.floor((seconds % 3600) / 60);
 const remainingSeconds = seconds % 60;

 const paddedHours = hours.toString().padStart(2, '0');
 const paddedMinutes = remainingMinutes.toString().padStart(2, '0');
 const paddedSeconds = remainingSeconds.toString().padStart(2, '0');

 return `${paddedHours}:${paddedMinutes}:${paddedSeconds}`;
}

function generateTableRow(id, color, name, duration) {
   const tableRow = document.createElement('tr');
   tableRow.classList.add("group")
   const link = `http://localhost:39999/stats/programm/${id}`;

   const cell1 = document.createElement('td');
   cell1.classList.add("rounded-l-md","group-hover:bg-blue-100","group-hover:bg-opacity-80");
   const coloredSquare = document.createElement('div');
   coloredSquare.classList.add('h-[1vw]', 'w-[1vw]', 'rounded-md', 'm-auto');
   coloredSquare.style.backgroundColor = color;
   const a1 = document.createElement("a");
   a1.href=link;
   a1.appendChild(coloredSquare);
   cell1.appendChild(a1);
   const cell2 = document.createElement('td');
   cell2.classList.add("group-hover:bg-blue-100","group-hover:bg-opacity-80");
   cell2.tabIndex = -1;
   const a2 = document.createElement("a");
   a2.href=link;
   a2.textContent = name;
   cell2.appendChild(a2);

   const cell3 = document.createElement('td');
   cell3.classList.add("rounded-r-md","group-hover:bg-blue-100","group-hover:bg-opacity-80");
   cell3.tabIndex = -1;
   const a3 = document.createElement("a");
   a3.href=link;
```

```
    a3.textContent = formatMinutesToTime(duration);
    cell3.appendChild(a3);

    tableRow.appendChild(cell1);
    tableRow.appendChild(cell2);
    tableRow.appendChild(cell3);




    return tableRow;
  }



async function renderChart(data) {
  const config = await generateConfig(data);

  new Chart(
    document.getElementById('hourlyChart'),
    config
  );
}

async function renderTable(data) {
  programmTable.innerHTML = "";
  data.reverse().forEach(program => {
    let duration = 0;
    for (const key in program.hourlyUse) {
      if (Object.hasOwnProperty.call(program.hourlyUse, key)) {
        const element = program.hourlyUse[key];
        duration += element;
      }
    }
    programmTable.appendChild(generateTableRow(program.id, program.color, program.name, duration));
  })
}

async function renderPage() {
  const dbData = await retriveData(date);
  renderChart(dbData);
  renderTable(dbData);
}

function nextDay(){
  date =  new Date(date.getTime() + (24 * 60 * 60 * 1000));
  dateDisplay.textContent = date.toLocaleDateString('uk')
  document.location.href = `http://localhost:39999/stats/day/${date.toISOString().split('T')[0]}`
  renderPage();
}

function previousDay() {
  date = new Date(date.getTime() - (24 * 60 * 60 * 1000));
  dateDisplay.textContent = date.toLocaleDateString('uk')
  document.location.href = `http://localhost:39999/stats/day/${date.toISOString().split('T')[0]}`
  renderPage();
```

```
}

renderPage();

document.getElementById("previousButton").addEventListener("click" , previousDay);
document.getElementById("nextButton").addEventListener("click" ,nextDay);


let programmDisplay = document.getElementById("programmDisplay");
let comapnyDisplay = document.getElementById("comapnyDisplay");
let percentageLabel = document.getElementById("percentageLabel");
let weeklyLabel = document.getElementById("weeklyLabel");

const daysOfWeek = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Нд'];
const daysOfWeekFull = ['понеділкх', 'вівторках', 'середах', 'четвергах', 'п\'ятницях', 'суботах', 'неділях'];

let extractProgrammIdFromUrl = (url) => {
  // Use regular expression to match the date format (YYYY-MM-DD)
  const regex = /\/programm\/(\d+)/;
  const match = url.match(regex);

  // Check if there's a match
  if (match) {
    // Extract the captured group (the date string)
    return match[1];
  } else {
    // Handle the case where the URL doesn't match the format
    console.error("Invalid URL format. Date not found.");
    return null;
  }
}

function formatMinutesToTime(minutes) {
  if (isNaN(minutes) || minutes < 0) {
    return "Invalid time";
  }

  const seconds = Math.floor(minutes * 60);

  const hours = Math.floor(seconds / 3600);
  const remainingMinutes = Math.floor((seconds % 3600) / 60);
  const remainingSeconds = seconds % 60;

  const paddedHours = hours.toString().padStart(2, '0');
  const paddedMinutes = remainingMinutes.toString().padStart(2, '0');
  const paddedSeconds = remainingSeconds.toString().padStart(2, '0');

  return `${paddedHours}:${paddedMinutes}:${paddedSeconds}`;
}

const programmId = extractProgrammIdFromUrl(window.location.href);

async function retriveData(id) {
  const response = await fetch(`http://localhost:39999/api/programm/${id}`);
  const data = await response.json();
  return data;
}
```

```
async function retrivePieData(id) {
  const response = await fetch(`http://localhost:39999/api/programm/${id}/pie`);
  const data = await response.json();
  return data;
}

function parseData(programm) {
  let usage = [0, 0, 0, 0, 0, 0, 0];
  let entryList = programm.entryList;
  entryList.forEach(entry => {
    let day = new Date(entry.startTime).getDay();
    day--;
    if (day == -1) {
      day = 6
    }
    usage[day] += entry.duration;
  });
  for (let index = 0; index < usage.length; index++) {
    usage[index] = Math.round((usage[index] / 60 / 60) * 100) / 100
  }
  console.log(usage);
  return {
    label: programm.name,
    data: usage,
    borderColor: programm.displayColor,
    backgroundColor: programm.displayColor + "40",
    hiden: false,
    borderWidth: 1
  }
}

function generateWeeklyChartConfig(data) {
  let config = {

    type: 'horizontalBar',
    data: {
      labels: daysOfWeek,
      datasets: [
        data
      ]
    },
    options: {
      maintainAspectRatio: true,
      aspectRatio: 1.2,
      legend: {
        display: false
      },
      scales: {
        xAxes: [{
          display: false

        }],
        yAxes: [{
          gridLines: {
            display: false
          }
```

```
      }]
    },


  },
 }
 console.log(config);
 return config
}

function generateWeeklyChart(programmData) {
 let data = parseData(programmData);
 let mostUsedDay = data.data.reduce((maxIndex, currentValue, currentIndex) => currentValue >
data.data[maxIndex] ? currentIndex : maxIndex, 0);

 weeklyLabel.textContent = `${programmData.name} найбільше використовувався по
${daysOfWeekFull[mostUsedDay]}`;

 const config = generateWeeklyChartConfig(data);
 console.log(config);
 new Chart(
  document.getElementById('weeklyChart'),
  config
 );
}

function generatePieChartConfig(pieData) {
 console.log(programmId);
 let borderColors = pieData.map(d => d.color);
 let bgColors = pieData.map((d) => {
  if (d.id == programmId) {
   return d.color ;
  } else {
   return d.color+ "50";
  }

 })


 let config = {
  type: 'doughnut',
  data: {
   labels: pieData.map(d => d.name),
   datasets: [{
    data: pieData.map(d => d.usage),
    borderColor: borderColors,
    backgroundColor: bgColors,
    hiden: false,
    borderWidth: 1
   }]
  },
  options: {
   maintainAspectRatio: true,
   aspectRatio: 1.2,
   legend: {
    display: false
   },
```

```
      tooltips: {
        callbacks:{
          label: function(tooltipItem, data) {
            return `${data.labels[tooltipItem.index]}: ${formatMinutesToTime(
data.datasets[tooltipItem.datasetIndex].data[tooltipItem.index]/60)}`;
          },
          afterLabel: function(tooltipItem, data) {
            let allDuration = data.datasets[tooltipItem.datasetIndex].data.reduce((acc, current) => acc+current,
0);
            let quotient = data.datasets[tooltipItem.datasetIndex].data[tooltipItem.index] / allDuration;

            let percentage = Math.round(quotient * 10000)/100;
            return `${percentage}%`;
          }
        }
      }
    },
  }
  return config;
}


function generatePieChart(pieData) {
  let program = pieData.find(p => p.id == programmId);
  let fullDuration = pieData.reduce((acc, prog) => acc + prog.usage,0);
  let percentage = Math.round(program.usage / fullDuration * 10000)/100;
  percentageLabel.textContent = `Всього ${program.name} використовувався ${percentage}% часу`

  let config = generatePieChartConfig(pieData);
  console.log(config);
  new Chart(
    document.getElementById('percentageChart'),
    config
  );
}

function countUsage(programmData) {
  return formatMinutesToTime(programmData.entryList.map(entry => entry.duration).reduce((acc, current)
=> acc+current, 0)/ 60);
}


async function renderPage() {
  let programmData = await retriveData(programmId)
  let pieData = await retrivePieData(programmId);

  programmDisplay.textContent = `${programmData.name} | ${countUsage(programmData)} `;
  comapnyDisplay.textContent = programmData.company;


  generateWeeklyChart(programmData);
  generatePieChart(pieData);

}

renderPage();
```

ДОДАТОК Г – ІЛЮСТРАТИВНА ЧАСТИНА

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ ТА АНАЛІЗУ АКТИВНОСТІ КОРИСТУВАЧА

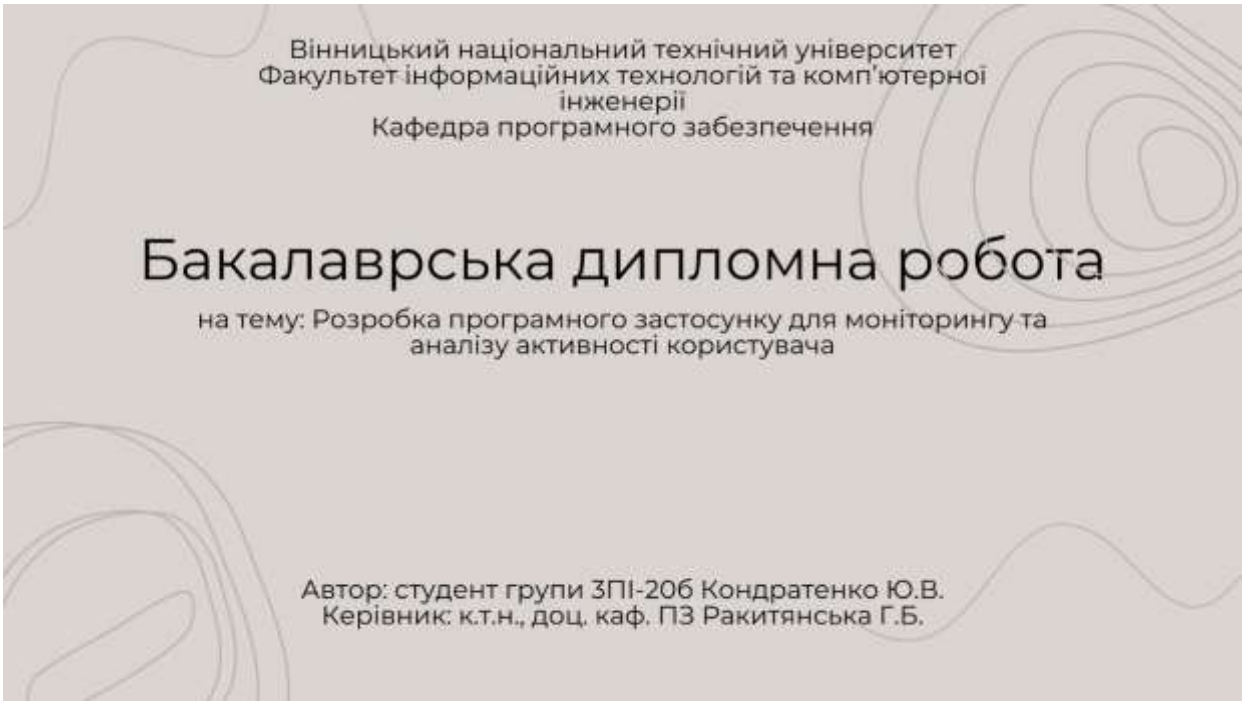


Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

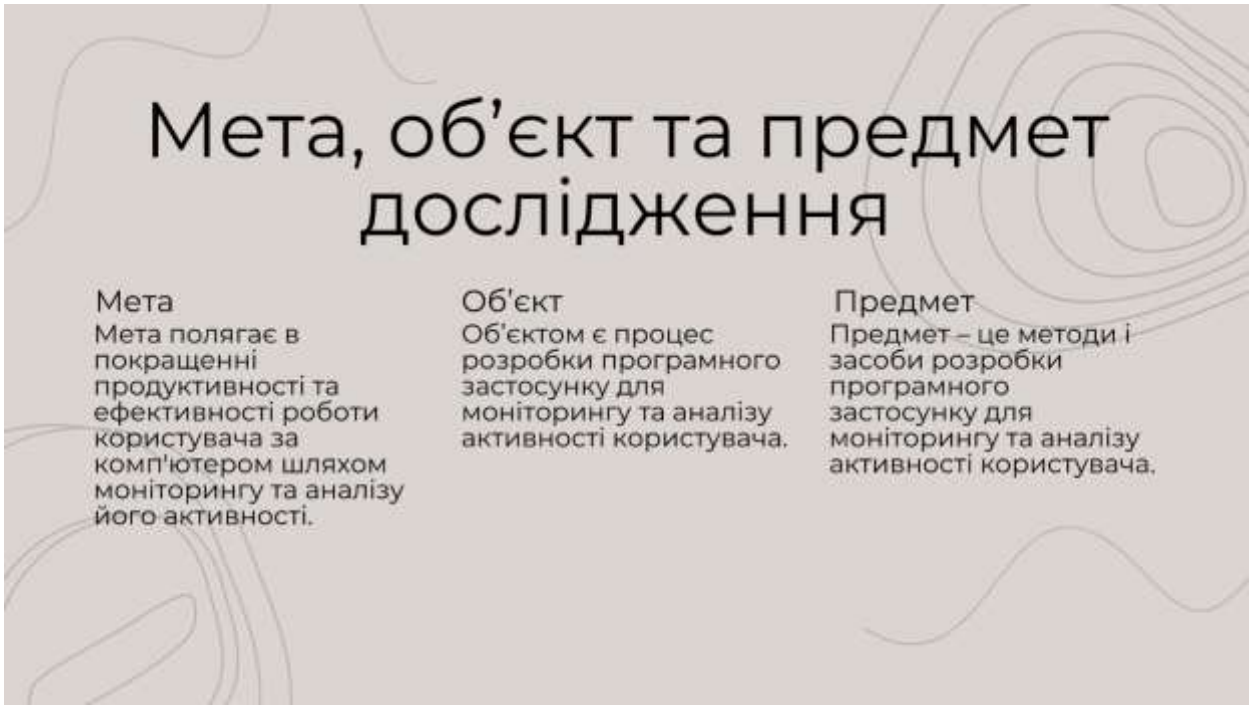


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Рисунок Г.4 – Новизна отриманих результатів

Порівняльний аналіз аналогів

Характеристика	ManicTime	Toggl Track	RescueTime	Clockify	UsageSense
Український інтерфейс	+	-	-	-	+
Автоматичне відстеження	+	-	+	+	+
Детальні звіти і аналітика	+	+	+	+	+
Категоризація діяльності	+	+	-	+	+
Безкоштовний доступ	-	-	+	+	+
Відкритий код	-	-	-	-	+
Можливості спільної роботи	-	+	+	+	+
Загальна оцінка	66%	50%	66%	83%	100%

Рисунок Г.5 – Порівняльний аналіз аналогів

Розробка алгоритмів роботи системи

Алгоритм збережнння даних про активне вікно

Алгоритм збирає дані про активність застосунку, перевіряючи активність та зчитуючи дані про вікно та процес. Якщо вікно не нове, дані записуються до буфера, який при заповненні зберігається до бази. У разі нового вікна створюється новий запис, перевіряється наявність програми в базі. Якщо є, дані з буфера зберігаються до бази, інакше реєструється нова програма, після чого дані з буфера зберігаються до бази. Процес завершується на етапі 15. Цей алгоритм ефективно збирає та зберігає дані про активність користувача, мінімізуючи навантаження на базу даних.

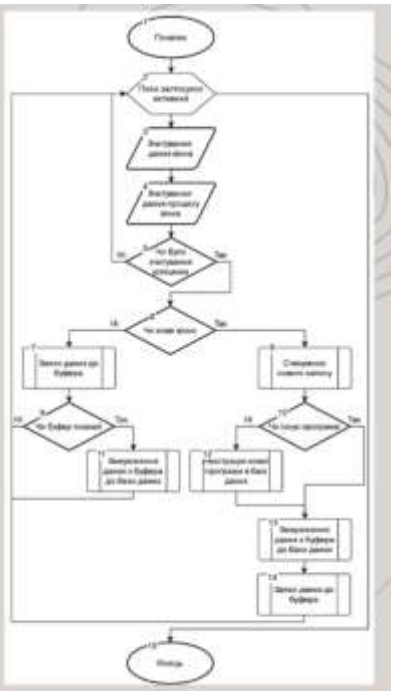


Рисунок Г.6 – Розробка алгоритмів роботи системи - Алгоритм збережнння даних про активне вікно

Рисунок Г.7 – Розробка алгоритмів роботи системи - Алгоритм створення таблиці головної сторінки

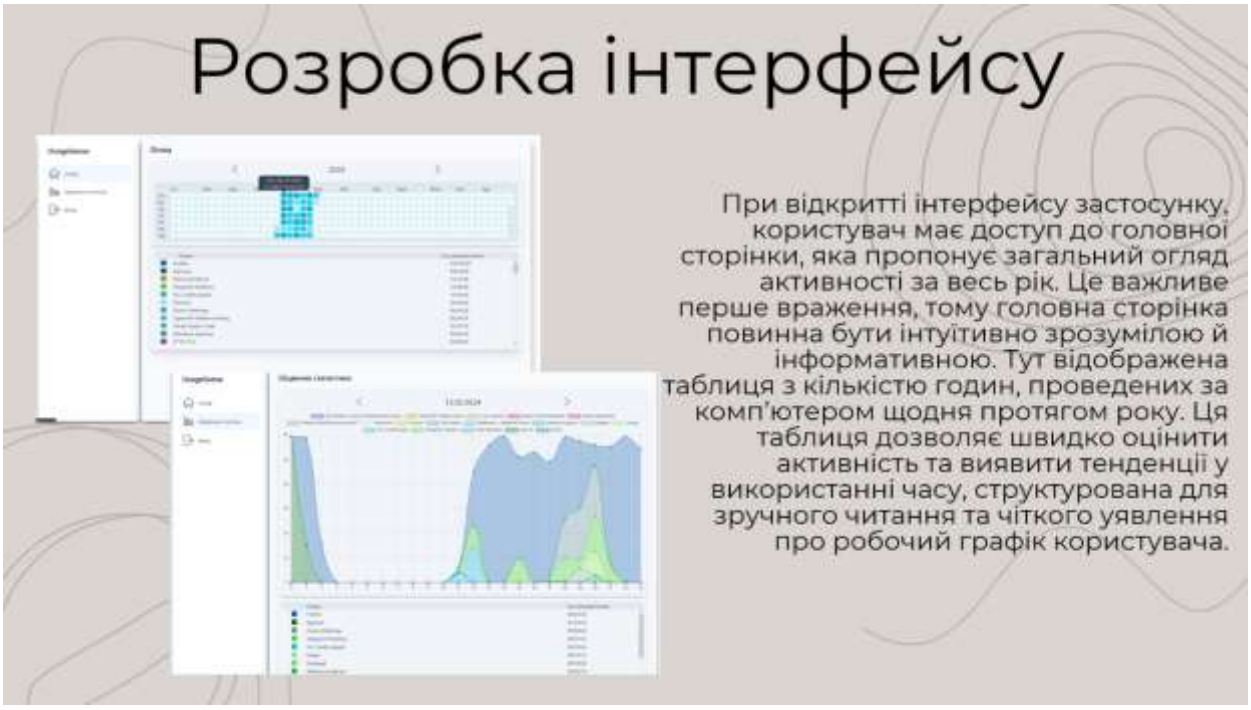


Рисунок Г.8 – Розробка інтерфейсу

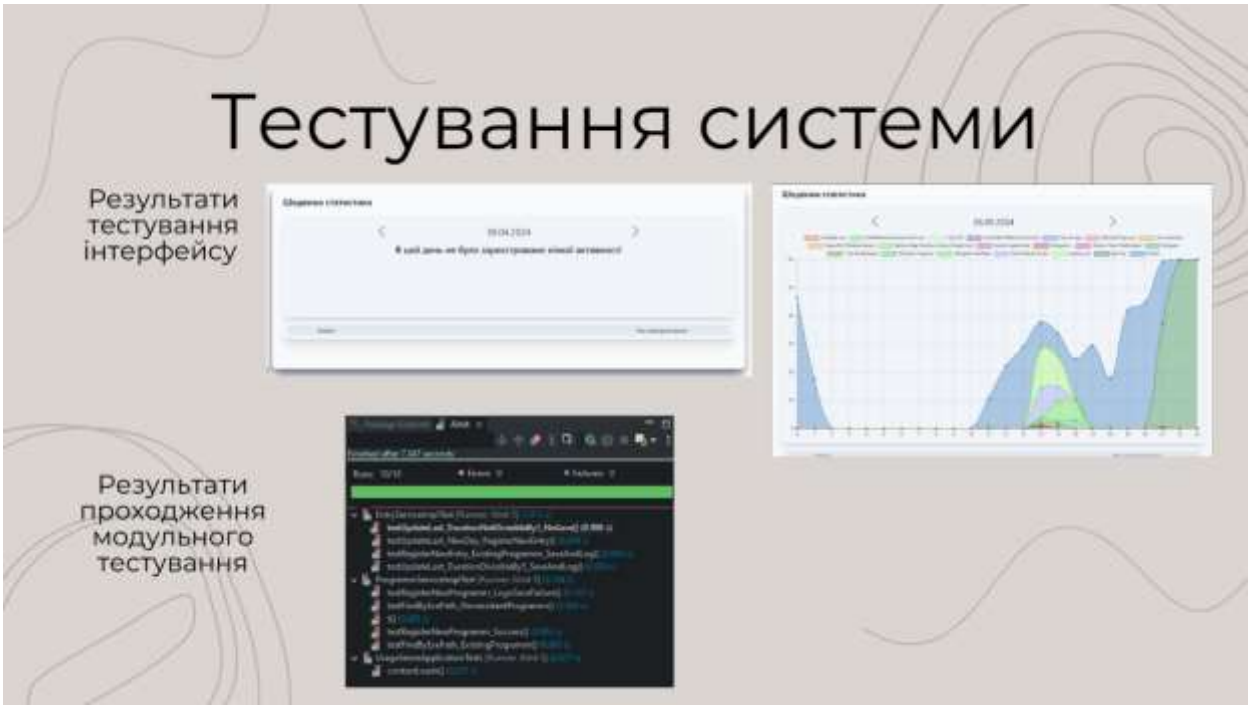


Рисунок Г.9 – Тестування системи

Рисунок Г.10 – Висновки