Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ ДІЙ КОРИСТУВАЧА ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА З МЕТОЮ АНАЛІЗУ ВИТРАЧЕНОГО ЧАСУ ТА ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ
(тема)

Виконав:
здобувач 4 року навчання,

групи ІТІНФ-22-2

Панчук В. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Панчуку Всеволоду Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для відстеження дій користувача персонального комп'ютера з метою аналізу витраченого часу та підвищення продуктивності

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 29 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література в галузі комп'ютерного зору, оптичного розпізнавання символів та тайм-менеджменту; публікації у періодичних виданнях та матеріали профільних міжнародних конференцій; технічна документація та специфікації операційної системи Windows; офіційна документація до мови програмування Python та її бібліотек (PySide6, OpenCV, NumPy, pytesseract, PyGetWindow, mss, hashlib); стандарти проєктування реляційних баз даних та специфікація СКБД SQLite; архітектурні шаблони та принципи побудови модульного програмного забезпечення.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих методів аналізу активності користувача.
2. Проєктування застосунку для аналізу активності користувача.
3. Експерементальна реалізація застосунку аналізу активності користувача.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність теми, об'єкт, предмет, мета та задачі дослідження, порівняльний аналіз існуючих програмних рішень для моніторингу активності, загальна модульна архітектура розроблюваного застосунку, алгоритм функціонування інтелектуальної системи фонового моніторингу, схема інформаційної взаємодії модуля керування та функціональних модулів, алгоритм та логічна структура роботи аналітичного модуля класифікації, інфологічна модель та структура таблиць локальної бази даних, головне вікно програми в увімкненому та вимкненому станах, вікно налаштувань інтерфейсу користувача, графічний інтерфейс системи аналізу журналів та візуалізація звітності, результати експериментальних досліджень та порівняльні діаграми точності класифікації у різних режимах роботи, загальні висновки за результатами виконання кваліфікаційної роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-20.04.26	
3	Аналіз літератури з проблеми, що вирішується	21.04.26-29.04.26	
4	Аналіз існуючих методів розпізнавання	30.04.26-04.05.26	
5	Формування кроків вирішення проблеми	06.05.26-13.05.26	
6	Програмна реалізація	10.05.26-15.06.26	
7	Аналіз отриманих результатів	16.06.26-18.06.26	
8	Оформлення пояснювальної записки	13.06.26-19.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Яковлева О. В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 95 с., 10 табл., 28 рис., 2 дод., 31 джерело.

АНАЛІЗ АКТИВНОСТІ, ЗАСТОСУНОК, КОМП'ЮТЕРНИЙ ЗІР, МОНІТОРИНГ, ПРОДУКТИВНІСТЬ, OCR, PYTHON.

Об'єктом роботи є процес аналізу активності користувача персонального комп'ютера.

Метою роботи є розроблення застосунку для відстеження дій користувача персонального комп'ютера з метою аналізу витраченого часу та оцінки продуктивності.

Під час виконання роботи використано методи аналізу даних та комп'ютерного зору, зокрема обробку зображень, виділення ознак, розпізнавання тексту (OCR), а також методи програмної реалізації на мові Python. Практична цінність роботи полягає у створенні застосунку, який дозволяє автоматично відстежувати активність користувача, визначати тип діяльності та оцінювати ефективність використання часу.

У результаті роботи розроблено застосунок, що забезпечує: визначення активного вікна користувача; виявлення періодів неактивності (AFK); аналіз зображень екрана; розпізнавання текстової інформації; класифікацію діяльності користувача; оцінку продуктивності.

Область застосування: самоконтроль користувачів, підвищення продуктивності праці, аналіз робочого часу, навчальні та офісні середовища.

ACTIVITY ANALYSIS, APPLICATION, COMPUTER VISION, MONITORING, OCR, PRODUCTIVITY, PYTHON.

The object of the work is the process of analyzing user activity on a personal computer.

The goal of this work is to develop an application for tracking user actions on a personal computer in order to analyze time usage and evaluate productivity.

The following methods were used: data analysis methods, computer vision techniques (image processing, feature extraction), optical character recognition (OCR), and software implementation using Python. The practical value of the work lies in creating an application that automatically tracks user activity, determines the type of activity, and evaluates time efficiency.

As a result of the work, an application was developed that provides: active window detection; inactivity (AFK) detection; screen image analysis; text recognition; activity classification; productivity evaluation.

Application areas: personal productivity tracking, time management, workplace analysis, and educational environments.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

Activewindow – активне вікно операційної системи, з яким взаємодіє користувач

AFK – Away From Keyboard (стан неактивності користувача, коли відсутня взаємодія з комп’ютером)

API – Application Programming Interface (інтерфейс програмного застосунку, що використовується для взаємодії між різними програмними компонентами)

Brightness – середня яскравість зображення

Classification – процес визначення типу діяльності користувача на основі аналізу даних

Contrast – міра відмінності між світлими та темними ділянками зображення

CPU – Central Processing Unit (центральний процесор комп’ютера)

CSV – це простий текстовий формат, призначений для зберігання та обміну табличними даними

Edge detection – метод виділення контурів на зображенні

Feature (ознака) – характеристика зображення або даних, що використовується для аналізу

FPS – Frames Per Second (кількість кадрів за секунду при обробці зображень)

Framedifference – різниця між двома послідовними кадрами

GUI – Graphical User Interface (графічний інтерфейс користувача)

Hash (хеш) – числове або символьне представлення даних, що використовується для швидкого порівняння зображень

Idle time – час неактивності користувача

JSON – розширення файлу, для зберігання та передачі структурованої інформації

Keyword – ключове слово, що використовується для аналізу тексту

NumPy – бібліотека для обробки числових даних та масивів

OCR – Optical Character Recognition (технологія оптичного розпізнавання тексту на зображеннях)

OpenCV – бібліотека комп'ютерного зору для обробки зображень

Productivity – показник ефективності використання часу користувачем

Pytesseract – бібліотека для розпізнавання тексту (OCR)

Python – мова програмування, що використовується для реалізації застосунку

RAM – Random Access Memory (оперативна пам'ять комп'ютера)

Screenshot – знімок екрана, отриманий у процесі роботи застосунку

Time tracking – процес відстеження часу активності користувача

ВСТУП

У сучасному світі інформаційні технології відіграють важливу роль у повсякденному житті людини. Значна частина часу витрачається на роботу за персональним комп’ютером, що включає як продуктивну діяльність (робота, навчання), так і розважальні процеси.

Зі зростанням обсягів цифрової інформації та кількості програмного забезпечення, яким користуються люди, виникає проблема ефективного контролю та аналізу витраченого часу. Багато користувачів не усвідомлюють, скільки часу вони витрачають на непродуктивні дії, що негативно впливає на їхню ефективність. Сучасні технології штучного інтелекту та комп’ютерного зору відкривають нові можливості для аналізу активності користувача. Зокрема, застосування методів аналізу зображень, розпізнавання тексту та класифікації дозволяє автоматизувати процес визначення типу діяльності користувача.

Актуальність роботи полягає у необхідності створення інтелектуальної системи, яка здатна автоматично аналізувати активність користувача персонального комп’ютера та оцінювати ефективність використання часу.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ АНАЛІЗУ АКТИВНОСТІ КОРИСТУВАЧА

1.1 Задачі аналізу активності користувача

На сьогоднішній день аналіз активності користувача є важливою задачею в галузі інформаційних технологій. Він дозволяє визначати ефективність використання часу, виявляти непродуктивні дії та оптимізувати робочі процеси. Особливо актуальною ця задача є в умовах широкого використання персональних комп’ютерів для роботи, навчання та повсякденного спілкування.

Під активністю користувача розуміють сукупність дій, які виконуються під час роботи за комп’ютером: запуск і використання програм, введення тексту, перегляд вебсторінок, роботу з документами, перегляд мультимедійного контенту та інші види взаємодії з операційною системою. Аналіз цих дій дає змогу отримати об’єктивну інформацію про характер використання комп’ютера та структуру витрат часу.

Основні задачі аналізу активності користувача включають:

− відстеження активних застосунків;

− визначення періодів активності та неактивності;

− класифікацію діяльності користувача;

− оцінку продуктивності;

− формування статистики та звітів.

Відстеження активних застосунків полягає у визначенні програми або вікна, з яким користувач взаємодіє в конкретний момент часу. Це дозволяє встановити, які інструменти використовуються найчастіше та скільки часу витрачається на кожен із них.

Визначення періодів активності та неактивності дає змогу відокремити фактичний робочий час від перерв, під час яких користувач не взаємодіє з комп’ютером. Такий підхід забезпечує більш точну оцінку реального часу

продуктивної роботи.

Класифікація діяльності користувача передбачає автоматичне визначення типу поточної активності, наприклад програмування, навчання, робота з документами, спілкування або розваги. Для цього можуть використовуватися дані про активне вікно, вміст екрана та результати розпізнавання тексту.

Оцінка продуктивності полягає у визначенні співвідношення між продуктивною та непродуктивною діяльністю. На основі заданих правил або моделей система може формувати інтегральний показник ефективності використання часу.

Формування статистики та звітів забезпечує збереження результатів аналізу у зручному для користувача вигляді. Це можуть бути таблиці, графіки, діаграми та текстові рекомендації щодо покращення організації роботи.

Системи аналізу активності користувача застосовуються в різних сферах. Для особистого використання вони допомагають контролювати власну продуктивність та боротися з прокрастинацією. У корпоративному середовищі такі системи використовуються для аналізу робочого часу співробітників. В освітній сфері вони дозволяють оцінювати ефективність самостійної роботи учнів та студентів.

Особливістю задачі аналізу активності є необхідність обробки різнорідних даних: системної інформації, знімків екрана, текстових даних та часових показників. Тому для побудови ефективної системи необхідно поєднувати методи моніторингу операційної системи, комп’ютерного зору, оптичного розпізнавання символів та алгоритми класифікації.

Таким чином, аналіз активності користувача є комплексною задачею, спрямованою на автоматичне визначення характеру роботи за комп’ютером, оцінювання ефективності використання часу та формування рекомендацій щодо підвищення продуктивності. Рішення цих задач дозволяє підвищити ефективність роботи користувача та оптимізувати використання його часу.

1.2 Аналіз існуючих рішень

На сьогоднішній день існує велика кількість програмних засобів для відстеження активності користувача, серед яких найбільш відомими є RescueTime [1], ManicTime [2], ActivityWatch [3], WakaTime [4] та Toggl Track [5]. Дані системи призначені для автоматичного збору інформації про використання програм, вебсайтів та тривалість роботи за комп'ютером.

Більшість сучасних рішень забезпечують можливість:

− відстеження часу використання програм та вебресурсів;

− визначення періодів активності та неактивності;

− формування статистичних звітів;

− аналізу продуктивності користувача;

− експорту даних у різні формати.

Попри широкий функціонал, більшість існуючих систем мають недоліки:

− обмежену точність визначення типу діяльності;

− відсутність аналізу фактичного вмісту екрана;

− необхідність ручного налаштування категорій;

− складність інтерфейсу;

− обмежений функціонал у безкоштовних версіях.

Одним із найвідоміших сервісів є RescueTime. Даний застосунок автоматично відстежує використання програм та вебсайтів, формує звіти про продуктивність та дозволяє аналізувати структуру витрат часу. Основною перевагою RescueTime є простий та зрозумілий інтерфейс, а також зручна система візуалізації статистики. На рисунку 1.1 наведено головне вікно застосунку RescueTime.

Проте безкоштовна версія RescueTime має суттєві обмеження та надає доступ лише до базових функцій. Крім того, система не аналізує безпосередній вміст екрана, а класифікація діяльності виконується переважно на основі назв програм і вебсайтів.

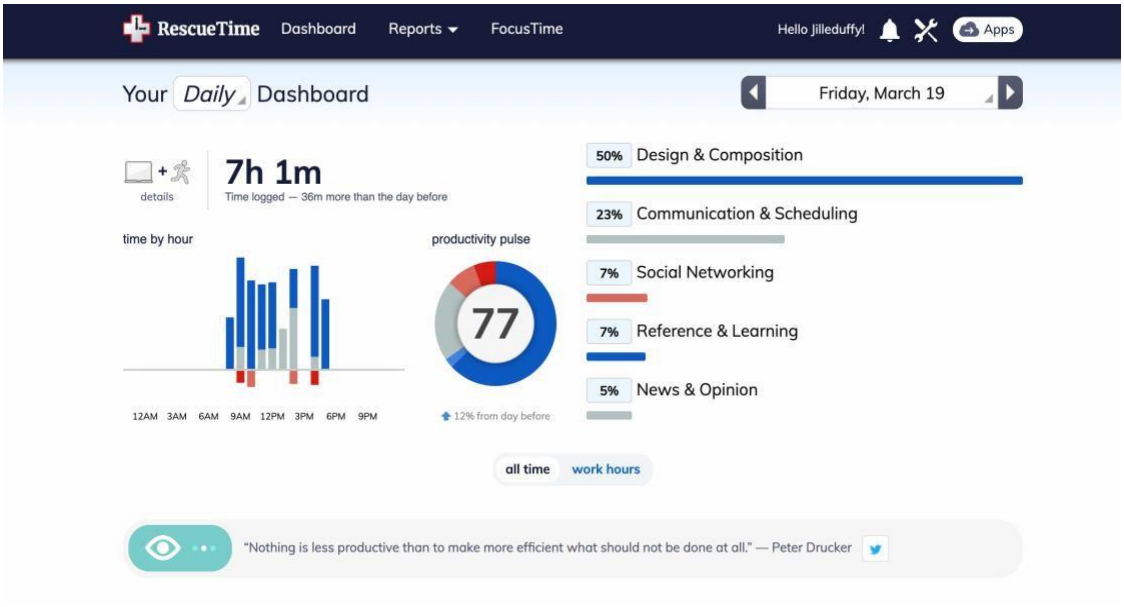

Рисунок 1.1 – Інтерфейс застосунку RescueTime

Програма ManicTime є потужним інструментом для локального моніторингу активності користувача. Вона зберігає всі дані на комп'ютері користувача, забезпечує детальний аналіз часових інтервалів та дозволяє будувати різноманітні звіти. Інтерфейс застосунку наведено на рисунку 1.2.

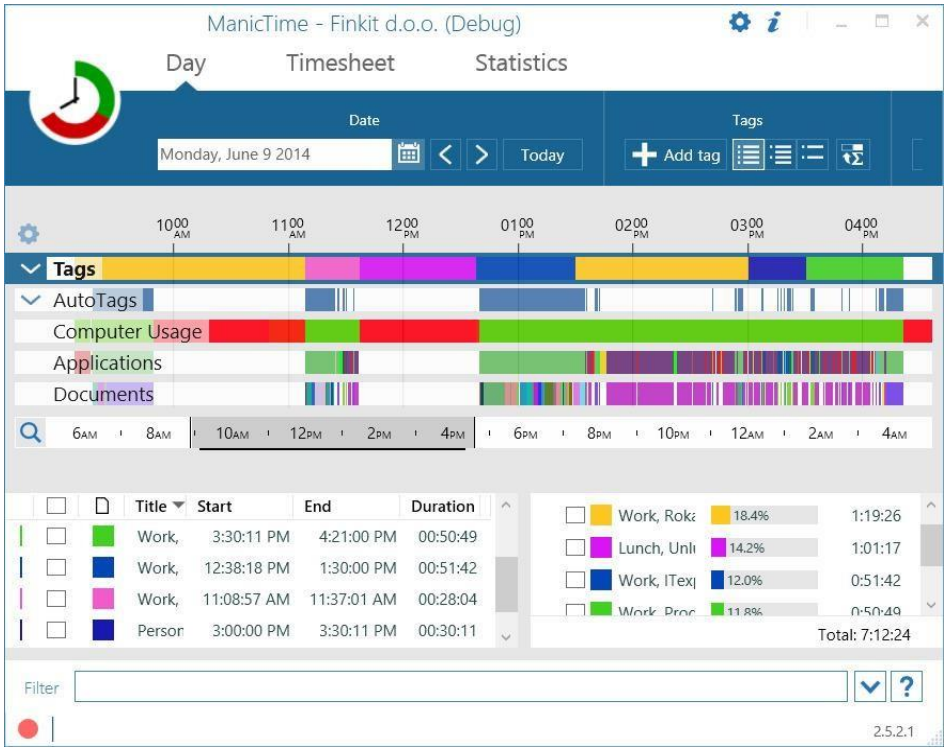

Рисунок 1.2 – Інтерфейс застосунку ManicTime

Перевагою ManicTime є широкий функціонал навіть у безкоштовній версії та високий рівень конфіденційності. Водночас інтерфейс програми є досить складним для нових користувачів, що ускладнює початок роботи без попереднього ознайомлення з документацією.

Ще одним популярним рішенням є ActivityWatch – безкоштовний застосунок із відкритим вихідним кодом. Він автоматично збирає інформацію про використання програм та вебсайтів, а також зберігає всі дані локально. Приклад інтерфейсу ActivityWatch наведено на рисунку 1.3.

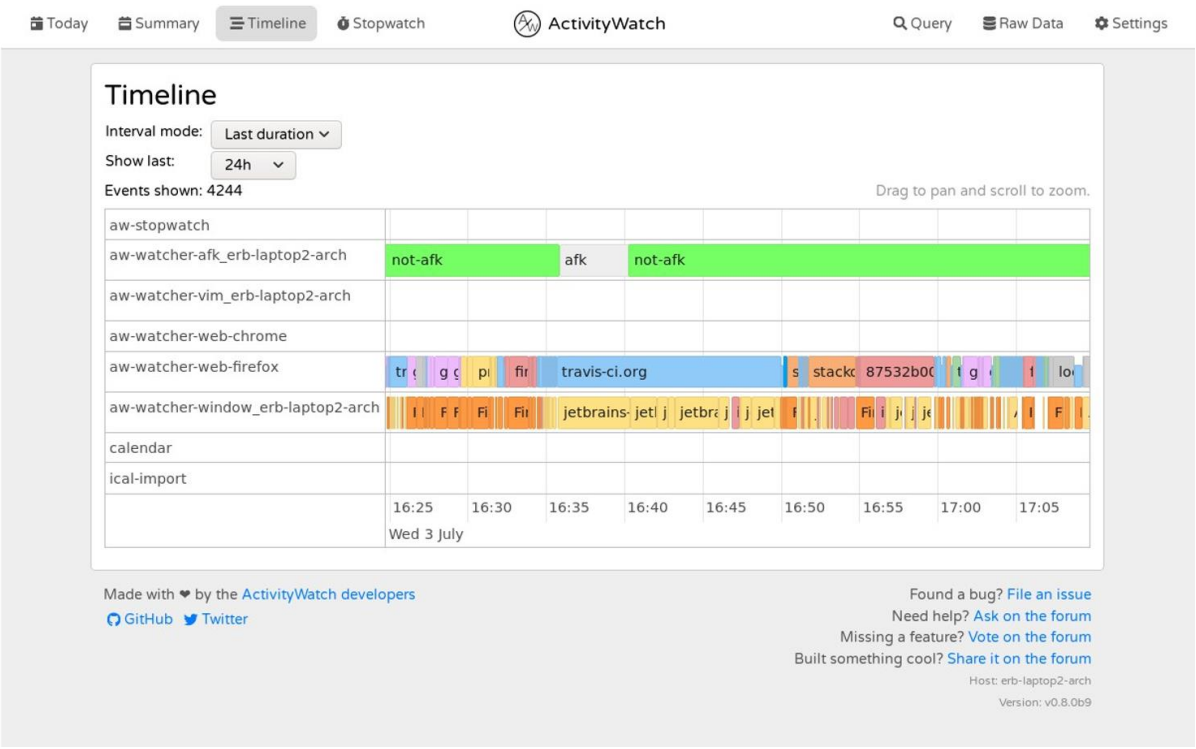

Рисунок 1.3 – Інтерфейс застосунку ActivityWatch

Основною перевагою застосунку ActivityWatch є повна безкоштовність і відкритість вихідного коду. Однак система не виконує аналіз зображень екрана та не використовує технології оптичного розпізнавання символів (Optical Character Recognition, OCR), що обмежує можливості точного визначення типу діяльності.

Сервіс WakaTime орієнтований переважно на розробників програмного забезпечення. Він інтегрується із середовищами розробки та автоматично

збирає статистику щодо часу програмування, використаних мов та проєктів. Інтерфейс сервісу наведено на рисунку 1.4.

Рисунок 1.4 – Інтерфейс сервісу WakaTime

WakaTime є ефективним інструментом для аналізу продуктивності програмістів, проте його функціональність обмежується переважно діяльністю у середовищах розробки.

Toggl Track є універсальною системою обліку часу, яка широко використовується для особистого планування та командної роботи. Основне вікно застосунку наведено на рисунку 1.5.

Перевагами Toggl Track є простий інтерфейс та можливість формування детальних звітів. Недоліком є необхідність ручного запуску та зупинки таймерів, що зменшує рівень автоматизації процесу моніторингу. Крім того, застосунок не забезпечує аналіз вмісту екрана та не підтримує методи комп’ютерного зору.

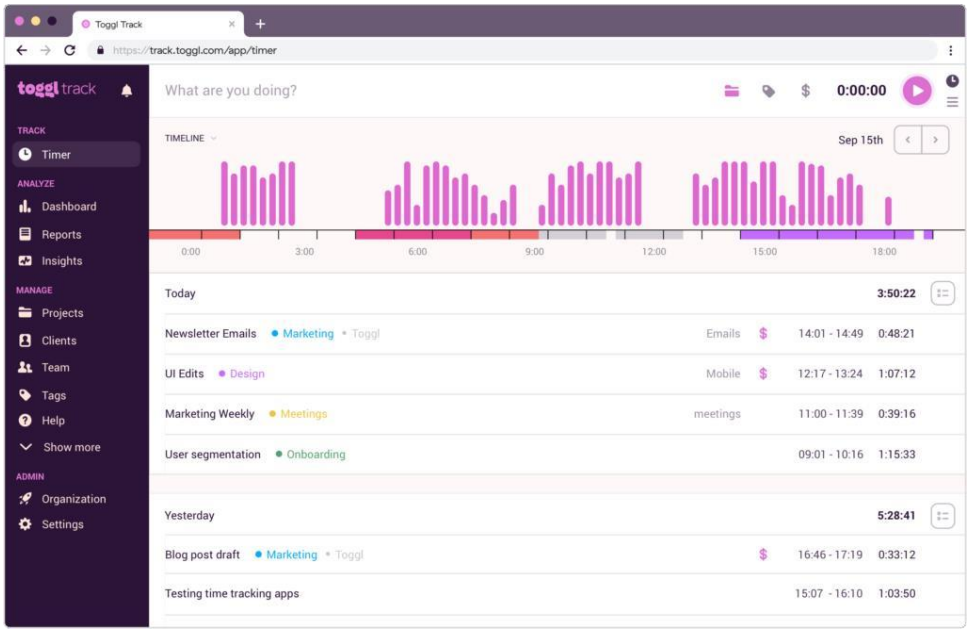

Рисунок 1.5 – Інтерфейс програми Toggl Track

Для наочного порівняння функціональних можливостей розглянутих систем доцільно використати таблицю 1.1.

Таблиця 1.1 – Порівняння існуючих систем аналізу активності користувача

Система	Відкритий код	Аналіз скріншотів	OCR	Безкоштовна версія
RescueTime	Ні	Ні	Ні	Частково
ManicTime	Ні	Ні	Ні	Частково
ActivityWatch	Так	Ні	Ні	Так
WakaTime	Частково	Ні	Ні	Частково
Toggl Track	Ні	Ні	Ні	Частково
Розроблюваний застосунок	Так	Так	Так	Так

Як видно з таблиці 1.1, жодне з розглянутих рішень не поєднує в собі безкоштовність, відкритий вихідний код, аналіз скріншотів та розпізнавання тексту. Більшість систем здійснює класифікацію діяльності лише на основі

назв програм або вебсайтів.

Таким чином, існує необхідність створення більш інтелектуального, безкоштовного та простого у використанні застосунку, який поєднуватиме аналіз активного вікна, обробку зображень екрана та OCR для більш точної оцінки активності користувача.

1.3 Методи аналізу активності користувача

У процесі розроблення системи аналізу активності користувача було використано комплексний підхід, що поєднує декілька методів обробки даних. Це дозволяє підвищити точність визначення типу діяльності користувача та забезпечити більш об’єктивну оцінку продуктивності [6].

До основних методів, що використовуються у розробці програмного застосунку, належать:

− відстеження активного вікна;

− визначення неактивності користувача;

− аналіз зображень;

− розпізнавання тексту.

1.3.1 Відстеження активного вікна

Одним із базових методів аналізу активності користувача є визначення активного вікна операційної системи. Цей підхід дозволяє встановити, який саме застосунок перебуває на передньому плані та з яким користувач взаємодіє в конкретний момент часу.

Для реалізації даного функціоналу в системі використовується бібліотека PyGetWindow [7], яка надає доступ до інформації про відкриті вікна операційної системи. За допомогою цієї бібліотеки система періодично

отримує поточне активне вікно та зчитує його заголовок.

Отриманий заголовок вікна може містити назву програми, відкритого документа, вебсторінки або вкладки браузера. Ця інформація передається до модуля аналізу, де використовується для визначення типу діяльності користувача. Наприклад, робота у середовищі розробки може класифікуватися як програмування, використання текстового редактора – як написання документів, а перегляд соціальних мереж – як розважальна активність.

Для забезпечення стабільної роботи системи передбачено обробку можливих помилок, які можуть виникати під час отримання інформації про активне вікно. Якщо визначити активне вікно не вдається, система повертає порожнє значення та продовжує роботу без переривання процесу моніторингу.

Перевагами даного методу є простота реалізації, низьке навантаження на систему та можливість отримання базової інформації про поточну активність користувача в режимі реального часу. Водночас точність цього підходу є обмеженою, оскільки він враховує лише назву активного вікна та не аналізує фактичний вміст екрана. Наприклад, одна й та сама програма може використовуватися як для виконання робочих завдань, так і для розважальних цілей, що ускладнює однозначну класифікацію діяльності лише на основі заголовка вікна.

1.3.2 Визначення неактивності користувача

Для підвищення точності аналізу активності користувача необхідно враховувати періоди, коли користувач фактично не взаємодіє з комп’ютером. З цією метою в системі реалізовано механізм визначення стану неактивності користувача (Away From Keyboard, AFK).

Основою даного підходу є аналіз послідовних знімків екрана. Система періодично отримує зображення робочого столу та порівнює його з попереднім кадром. Якщо протягом заданого проміжку часу вміст екрана не

змінюється, користувач вважається неактивним. Також система повинна враховувати присутність користувача біля комп’ютеру, якщо увімкнена взаємодія з вебкамерою та користувача немає у полі зору програма автоматично вважає що він неактивний.

Для зменшення обчислювального навантаження використовується бібліотека hashlib [8], за допомогою якої для кожного зображення обчислюється хеш-значення. Перед хешуванням зображення зменшується шляхом вибірки лише частини пікселів, що дозволяє значно прискорити обробку без суттєвої втрати точності. Якщо хеш поточного знімка відрізняється від попереднього, система фіксує момент останньої активності.

Для вимірювання часу неактивності використовується стандартний модуль time, який дозволяє визначити, скільки секунд минуло з моменту останньої зміни зображення. Якщо цей інтервал перевищує встановлений поріг (наприклад, 60 секунд), користувач вважається таким, що не взаємодіє з комп’ютером.

Додатково для більш точного аналізу використовується бібліотека NumPy [9], яка дозволяє обчислити середню абсолютну різницю між двома послідовними кадрами. Цей метод дає змогу враховувати незначні зміни на екрані та точніше оцінювати рівень активності користувача.

Застосування описаних методів дозволяє ефективно визначати періоди бездіяльності користувача та виключати їх із подальшого аналізу продуктивності. Це підвищує достовірність статистики та забезпечує більш коректну оцінку фактичного часу активної роботи за комп’ютером.

1.3.3 Аналіз зображень

Аналіз зображень є важливою складовою системи моніторингу активності, оскільки дозволяє отримати додаткові характеристики діяльності користувача на основі скріншотів екрана. На відміну від аналізу лише

активного вікна, цей підхід враховує безпосередній візуальний вміст екрана, що дає змогу точніше оцінювати характер виконуваної роботи [10].

Для обробки зображень використовується бібліотека OpenCV [11, 12], яка забезпечує широкий набір інструментів комп'ютерного зору, а також бібліотека NumPy для виконання числових обчислень.

На першому етапі отриманий скріншот зменшується до фіксованого розміру [13]. Це дозволяє значно зменшити обсяг даних та прискорити подальшу обробку без істотної втрати інформації. Після цього зображення перетворюється у відтінки сірого, що спрощує розрахунок базових статистичних характеристик [14].

У процесі аналізу визначаються такі показники:

− яскравість – середнє значенняінтенсивності пікселів, яке характеризує загальний рівень освітленості зображення;

− контраст – стандартне відхилення значень яскравості, що відображає ступінь різноманітності світлих і темних ділянок;

− колірна варіативність – дисперсія значень кольорових каналів, яка характеризує насиченість та різноманітність кольорів;

− щільність контурів – частка пікселів, що належать до меж об'єктів, визначених за допомогою алгоритму виявлення границь Canny.

Для додаткового аналізу активності виконується оцінка змін між послідовними кадрами. За допомогою функції абсолютної різниці між двома зображеннями визначається кількість пікселів, значення яких змінилося понад встановлений поріг. Отриманий показник характеризує рівень руху на екрані та дозволяє виявляти динамічні сцени, наприклад прокручування документів, перемикання між вікнами або перегляд відео, а відсутність зміни кадрів навпаки свідчить про відсутність активності [15].

Сукупність розрахованих характеристик використовується як набір ознак для подальшої класифікації діяльності користувача. Наприклад, висока щільність контурів і помірна колірна варіативність можуть бути характерними для програмного коду, тоді як значна кількість руху та висока насиченість

кольорів часто спостерігаються під час перегляду мультимедійного контенту.

Таким чином, аналіз зображень дозволяє суттєво розширити можливості системи моніторингу та підвищити точність визначення типу активності користувача [16–18].

1.3.4 Розпізнавання тексту

Для більш точного визначення типу діяльності користувача в системі використовується технологія OCR. Даний підхід дозволяє автоматично витягувати текстову інформацію зі скріншотів екрана та використовувати її для подальшого аналізу [19].

Для реалізації OCR застосовується бібліотека pytesseract [20], яка є Python-інтерфейсом до системи розпізнавання тексту Tesseract OCR. Вона дозволяє розпізнавати текст на зображеннях різними мовами та повертати його у вигляді звичайного текстового рядка [21].

У процесі роботи система передає попередньо оброблений скріншот до модуля OCR, після чого отримує текстовий вміст, присутній на екрані. Це можуть бути назви документів, фрагменти програмного коду, текст статей, повідомлення, елементи вебсторінок та інші текстові дані.

Отриманий текст використовується для пошуку ключових слів, характерних для певних типів діяльності. Наприклад:

− наявність конструкцій мов програмування може свідчити про розробку програмного забезпечення;

− слова, пов’язані з навчальними матеріалами, можуть вказувати на освітню діяльність;

− назви соціальних мереж, відеоплатформ або ігрових сервісів можуть свідчити про розважальне використання комп’ютера.

Додатково результати OCR можуть застосовуватися для оцінки обсягу тексту на екрані, визначення мови контенту та формування текстових ознак

для моделей машинного навчання [22].

Основною перевагою даного методу є можливість аналізу фактичного змісту екрана, що значно підвищує точність класифікації порівняно з підходами, які враховують лише назву активного вікна або загальні характеристики зображення.

Водночас ефективність OCR залежить від якості зображення, розміру шрифту, мови тексту та наявності графічних елементів. У випадку низької роздільної здатності або складного оформлення сторінки точність розпізнавання може знижуватися.

Таким чином, використання технології OCR дозволяє отримувати цінну текстову інформацію зі скріншотів та забезпечує більш глибоке розуміння характеру діяльності користувача.

Також для аналізу текстової інформації на екрані можна використовувати великі мовні моделі (Large Language Models, LLMs) [23].

1.4 Метод класифікації активності

Класифікація активності користувача є завершальним етапом аналізу даних, отриманих у процесі моніторингу роботи за комп’ютером. Її метою є автоматичне визначення типу поточної діяльності користувача на основі сукупності ознак, отриманих з різних джерел інформації.

У даній роботі класифікація здійснюється за допомогою декількох джерел даних:

− назви активного вікна;

− характеристик зображення екрана;

− текстової інформації, отриманої за допомогою OCR;

− інформації про періоди активності та неактивності користувача.

На основі отриманих даних система формує набір ознак [24], які характеризують поточний стан користувача. До таких ознак можуть належати

назва програми, кількість розпізнаного тексту, наявність ключових слів, рівень змін на екрані, щільність контурів, а також тривалість безперервної активності.

У межах роботи розглядаються такі основні категорії діяльності:

− програмування;

− робота з документами;

− навчання;

− перегляд вебсторінок;

− спілкування;

− розваги;

− неактивність.

Для кожної категорії діяльності задається набір правил та ключових ознак, які дозволяють оцінити ймовірність належності поточної активності до відповідного класу. Наприклад, наявність назв середовищ розробки, фрагментів програмного коду та спеціальних конструкцій мов програмування підвищує ймовірність того, що користувач займається програмуванням.

Якщо в розпізнаному тексті виявляються слова, пов’язані з навчальними матеріалами, а активним є браузер або програма для перегляду документів, система може класифікувати діяльність як навчання. Аналогічно, виявлення назв соціальних мереж, месенджерів або мультимедійних платформ може свідчити про спілкування або розважальну активність.

Для кожної категорії обчислюється числова оцінка, яка відображає ступінь відповідності поточних ознак заданим правилам. Категорія з найбільшим значенням оцінки вважається результатом класифікації.

Результати класифікації зберігаються у файл у форматі TXT. Для кожного запису фіксуються:

− дата і час спостереження;

− назва активного вікна;

− визначена категорія діяльності;

− додаткові характеристики аналізу.

Накопичені дані використовуються для побудови статистики та формування звітів, які відображають розподіл часу між різними видами діяльності. Це дозволяє користувачу оцінити власну продуктивність та виявити чинники, що впливають на ефективність роботи.

Перевагою такого підходу є поєднання кількох джерел інформації, що забезпечує вищу точність порівняно з методами, які аналізують лише назву активного вікна або часові інтервали. Крім того, система є гнучкою та може бути легко адаптована до нових категорій діяльності шляхом зміни правил класифікації.

Таким чином, використання методу класифікації активності дозволяє автоматично визначати характер діяльності користувача, оцінювати продуктивність роботи та формувати структуровану інформацію для подальшого аналізу.

1.5 Постановка задачі

У сучасному світі ефективний контроль цифрового часу та підвищення продуктивності праці за персональним комп’ютером є надзвичайно актуальними задачами. Проведений аналіз існуючих програмних рішень (RescueTime, ManicTime, ActivityWatch тощо) показав, що більшість із них мають суттєві обмеження: вони класифікують діяльність лише на основі назв запущених програм або заголовків вікон, не аналізують фактичний вміст екрана, мають комерційні обмеження або складний інтерфейс.

У зв’язку з цим ставиться завдання розроблення кросплатформеного (з первинною орієнтацією на Windows 10) інтелектуального застосунку з відкритим вихідним кодом, який поєднуватиме методи системного моніторингу, обробки зображень та оптичного розпізнавання символів для багаторівневого аналізу та точного визначення типу діяльності користувача.

Об’єктом роботи є процес аналізу активності користувача

персонального комп’ютера.

Метою роботи є розроблення застосунку для відстеження дій користувача персонального комп’ютера з метою аналізу витраченого часу та оцінки продуктивності.

Для досягнення поставленої мети необхідно вирішити такі завдання:

− проаналізувати існуючі методи, алгоритми та програмні рішення для відстеження активності користувача ПК, виявити їхні переваги та функціональні обмеження;

− дослідити методи комп'ютерного зору (обробки зображень, виділення ознак) та технології оптичного розпізнавання тексту для аналізу вмісту екрана;

− спроєктувати загальну модульну архітектуру системи, включаючи модулі системного моніторингу, створення скріншотів, аналізу зображень, OCR, взаємодії з вебкамерою та збереження даних;

− розробити алгоритм та евристичні правила для автоматичної класифікації діяльності користувача за категоріями (програмування, навчання, розваги, неактивність тощо) на основі комплексу отриманих ознак;

− реалізувати механізм визначення стану неактивності користувача на основі хешування послідовних знімків екрана та аналізу присутності в полі зору вебкамери;

− програмно реалізувати прототип застосунку на мові Python з використанням бібліотек PySide6, OpenCV, NumPy, pytesseract та СКБД SQLite для локального зберігання даних;

− розробити модулі статистики та аналізу журналів (логів) з інтеграцією елементів штучного інтелекту для формування звітів та рекомендацій користувачеві;

− провести експериментальне тестування розробленого застосунку в різних режимах (з використанням вебкамери/інтернету та без них), виконати порівняльний аналіз очікуваних і фактичних результатів та оцінити точність класифікації.

2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ АКТИВНОСТІ КОРИСТУВАЧА

2.1 Формування вимог до програмного застосунку

На основі проведеного аналізу предметної області було сформовано вимоги до програмного застосунку для аналізу активності користувача.

Застосунок повинен визначати активне вікно, виділяти головне з того, що є на екрані користувача, аналізувати отриману інформацію та записувати її у вайл для подальшого аналізу, тому з функціональних вимог можна виділити:

− визначення активного вікна операційної системи;

− створення скріншотів екрана через заданий інтервал часу;

− визначення періодів неактивності користувача;

− розпізнавання тексту зі скріншотів;

− аналіз характеристик зображення;

− класифікацію діяльності користувача;

− обчислення показників продуктивності;

− збереження результатів у файл;

− формування статистичних звітів.

З нефункціональних вимог розроблюваної програми можна одразу виділиит:

− працювати на персональних комп'ютерах під керуванням операційної системи Windows;

− не створювати значного навантаження на процесор та оперативну пам'ять;

− мати простий та зрозумілий інтерфейс;

− забезпечувати конфіденційність даних користувача;

− бути придатним до подальшого розширення.

Оскільки програма повинна аналізувати дії користувача на екрані та з

відео вебкамери, до вхідних даних системи можна віднести:

− заголовок активного вікна;

− зображення екрана;

− текст, отриманий за допомогою OCR;

− часові мітки;

− параметри конфігурації;

− відеопотік вебкамери.

А до результату роботи або вихідних даних системи:

− визначена категорія діяльності;

− оцінка продуктивності;

− статистичні показники;

− файли журналу та звітів.

2.2 Проєктування загальної архітектури системи

Програмний застосунок побудовано за модульним принципом. Кожен модуль відповідає за виконання окремої функції, що спрощує розроблення, тестування та супровід системи, загальна архітектура розроблюваної системи зображена на рисунку 2.1.

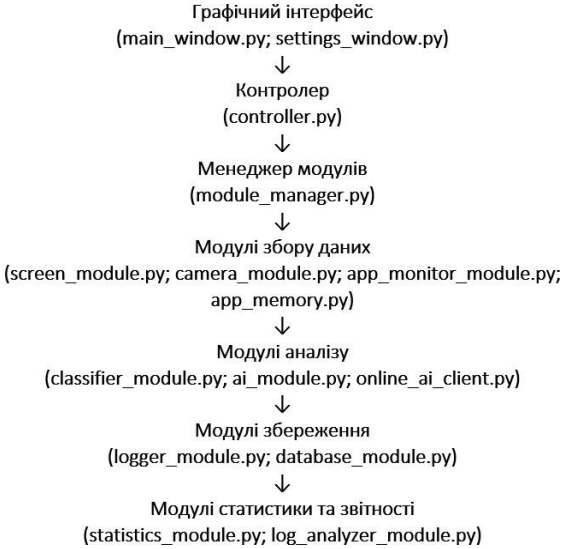

Рисунок 2.1 – Загальна архітектура системи

Опис основних компонентів:

− MainWindow забезпечує взаємодію користувача із системою, запуск і зупинку моніторингу, налаштування параметрів та перегляд поточного стану. SettingsWindow забезпечує взаємодію користувача з налаштуваннями системи, дозволяє змінювати частоту знімків екрану, перемикати світлу та темну теми, вмикати та вимикати використанні онлайн LLM та камери;

− Controller координує роботу системи, запускає цикли моніторингу та керує обміном даними між модулями;

− ModuleManager створює та ініціалізує всі функціональні модулі відповідно до конфігурації застосунку;

− AppMonitorModule отримує назву активного вікна. ScreenModule створює скріншоти екрана;

− CameraModule отримує зображення з вебкамери для визначення присутності користувача;

− ClassifierModule виконує AFK-аналіз, OCR, обчислення ознак та класифікацію активності;

− AIModule забезпечує можливість використання локальних або онлайн-моделей штучного інтелекту для покращення класифікації;

− LoggerModule зберігає результати у текстові файли. DatabaseModule записує інформацію до бази даних SQLite. StatisticsModule обчислює підсумкові статистичні показники;

− AppMemoryModule керує збереженням та доступом до внутрішньої пам’яті застосунку (контекст сесій, кеш для ШІ, налаштування);

− Модуль знімків екрану періодично зберігає зображення екрана;

− Модуль OCR виконує розпізнавання текстової інформації;

− Модуль аналізу зображень обчислює статистичні характеристики скріншота;

− Модуль класифікації визначає тип діяльності користувача;

− Модуль збереження даних записує результати до файлів TXT;

− Модулі звітності формують підсумкові статистичні звіти.

2.3 Розроблення алгоритму аналізу активності користувача

На основі сформованих вимог можемо розробити алгоритм роботи програми.

Після запуску програмного застосунку виконується ініціалізація всіх необхідних компонентів системи. Головний модуль програми завантажує конфігураційні параметри з файлу settings.json, створює графічний інтерфейс користувача та передає керування центральному контролеру.

На першому етапі контролер за допомогою модуля ModuleManager створює та ініціалізує всі функціональні модулі системи, зокрема модулі моніторингу активного вікна, створення скріншотів, аналізу зображень,розпізнавання тексту, класифікації активності, ведення журналу та збереження даних у базу даних.

Після завершення ініціалізації запускається основний цикл роботи програми, який виконується з певним часовим інтервалом. Під час кожної ітерації циклу система спочатку отримує інформацію про поточне активне вікно за допомогою модуля AppMonitorModule. Отримані дані містять назву програми або документа, з яким у даний момент працює користувач.

Далі модуль ScreenModule створює знімок екрана, який використовується для подальшого аналізу. Отриманий скріншот передається до модуля класифікації, де виконується перевірка на неактивність користувача. Для цього поточне зображення порівнюється з попереднім кадром. Якщо зміни на екрані відсутні протягом заданого проміжку часу, система визначає стан AFK.

Якщо користувач є активним, виконується оптичне розпізнавання тексту за допомогою OCR, а також обчислення візуальних характеристик зображення, таких як яскравість, контраст, щільність контурів та варіативність кольорів. Одночасно аналізується назва активного вікна та здійснюється пошук ключових слів у розпізнаному тексті.

На основі отриманих даних формується набір ознак, який передається до

модуля класифікації. Модуль ClassifierModule визначає найбільш ймовірну категорію діяльності користувача, наприклад програмування, навчання, роботу з документами, спілкування або розваги. Якщо в налаштуваннях активовано використання вебкамери, результати аналізу можуть додатково уточнюватися модулем CameraModule.

Після завершення класифікації результати аналізу зберігаються двома способами. Модуль LoggerModule записує інформацію до текстового файлу журналу, а модуль DatabaseModule зберігає структуровані дані у базі даних SQLite. До запису включаються дата і час спостереження, назва активного вікна, визначена категорія діяльності, оцінка продуктивності та допоміжні характеристики.

Загальний алгоритм роботи системи представлено на рисунку 2.2.

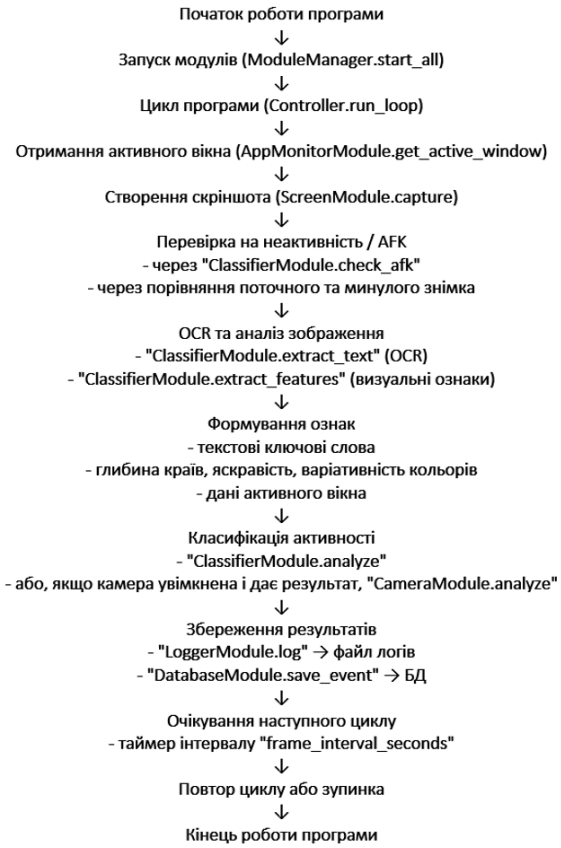

Рисунок 2.2 – Алгоритм роботи системи

Після збереження результатів програма переходить у режим очікування на час, визначений параметром frame_interval_seconds. Після завершення

цього інтервалу запускається наступна ітерація циклу моніторингу. Робота системи продовжується до моменту, коли користувач зупиняє моніторинг або завершує роботу програми.

Також розробимо псевдокод алгоритму для демонстрації, псевдокод продемонстровано на рисунку 2.3.

```
while monitoring_enabled:
active_window = AppMonitorModule.get_active_window() screenshot =
ScreenModule.capture()
if ClassifierModule.check_afk(screenshot) or
compare_screenshots(screenshot, previous_screenshot):
category = "afk" else:
text = ClassifierModule.extract_text(screenshot)  image_features =
ClassifierModule.extract_features(screenshot)
features = combine_features(active_window, text, image_features) category
= ClassifierModule.analyze(screenshot, active_window)
if camera_enabled and CameraModule.analyze() is not None: category =
CameraModule.analyze()
LoggerModule.log(category) DatabaseModule.save_event(category)
wait(frame_interval_seconds)
```

Рисунок 2.3 – Псевдокод алгоритму

2.4 Проєктування основних модулів застосунку

Програмний застосунок Activity Monitor побудовано за модульним принципом. Кожен функціональний компонент реалізовано у вигляді окремого програмного модуля, який відповідає за виконання певного набору завдань.

Такий підхід спрощує розроблення, тестування та подальше розширення системи, оскільки дозволяє змінювати окремі компоненти без впливу на інші частини програми та запобігає виходу з ладу усієї системи одразу. Це також забезпечує високий рівень гнучкості при інтеграції нових

інструментів аналізу та дозволяє легко адаптувати застосунок під специфічні потреби користувача.

Основні модулі застосунку розташовані в каталозі modules, а їх взаємодія координується класами Controller та ModuleManager, що знаходяться в каталозі core.

2.4.1 Модуль керування системою

Центральним компонентом програми є клас Controller, реалізований у файлі core/controller.py. Він відповідає за запуск і зупинку моніторингу, створення окремого потоку виконання та організацію основного циклу роботи програми.

Під час ініціалізації Controller завантажує параметри конфігурації з файлу settings.json, створює об’єкт ModuleManager та встановлює внутрішні параметри системи, зокрема інтервал моніторингу, поріг визначення неактивності та налаштування окремих модулів.

Метод start() запускає всі необхідні модулі та створює фоновий потік, у якому виконується метод run_loop(). У цьому методі з певною періодичністю викликається run_cycle(), що реалізує повний цикл аналізу активності користувача.

Таким чином, Controller виконує роль центрального координатора, який керує всіма етапами роботи застосунку. Схема роботи класу зображена на рисунку 2.4.

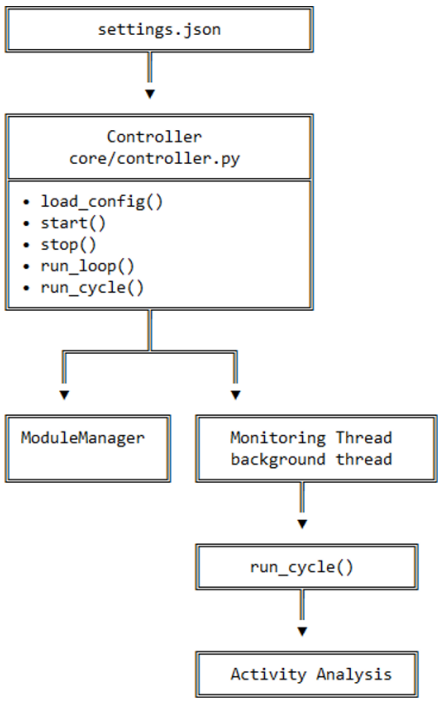

Рисунок 2.4 – Схема роботи класу Controller

2.4.2 Модуль керування функціональними компонентами

Клас ModuleManager, реалізований у файлі core/module_manager.py, відповідає за створення, ініціалізацію та запуск усіх функціональних модулів системи.

Під час створення об’єкта ModuleManager формуються екземпляри таких модулів:

− ScreenModule;

− AppMonitorModule;

− ClassifierModule;

− CameraModule;

− LoggerModule;

− DatabaseModule;

− StatisticsModule.

Метод start_all() послідовно запускає всі модулі, які увімкнені у конфігураційному файлі. Це дозволяє гнучко налаштовувати склад системи.

Наприклад, користувач може вимкнути використання вебкамери або збереження до бази даних без внесення змін до програмного коду.

Завдяки використанню ModuleManager процес керування модулями централізований і не потребує дублювання коду, принцип зображено на рисунку 2.5.

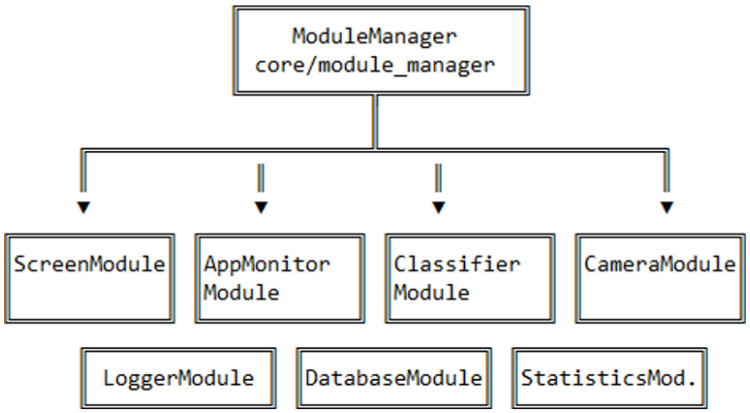

Рисунок 2.5 – Взаємодія ModuleManager з функціональними модулями

2.4.3 Модуль визначення активного вікна

Модуль AppMonitorModule, реалізований у файлі modules/app_monitor_module.py, призначений для визначення активного вікна операційної системи.

Для реалізації цього функціоналу використовується бібліотека PyGetWindow, яка дозволяє отримати об’єкт активного вікна та його заголовок. У заголовку міститься назва програми, документа або вебсторінки, з якою в даний момент працює користувач.

Щоб зменшити навантаження на систему, модуль використовує кешування результату протягом короткого проміжку часу. Якщо повторний запит виконується раніше встановленого інтервалу, повертається попереднє значення без повторного звернення до операційної системи.

Отримана назва активного вікна є однією з основних ознак для

подальшої класифікації діяльності користувача, схема роботи модуля зображена на рисунку 2.6.

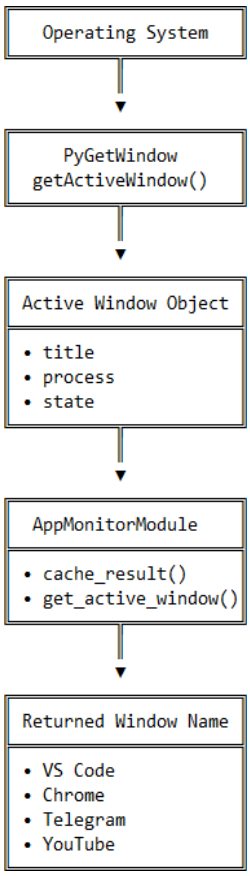

Рисунок 2.6 – Схема отримання назви активного вікна

2.4.4 Модуль створення скріншотів

Модуль ScreenModule, реалізований у файлі modules/screen_module.py, відповідає за створення знімків екрана через задані проміжки часу.

Для захоплення зображення використовується бібліотека mss, яка забезпечує швидке отримання скріншотів із мінімальним навантаженням на систему. Отримане зображення перетворюється у формат NumPy-масиву для подальшої обробки.

Модуль також підтримує збереження останніх кадрів у буфері, що використовується для визначення неактивності користувача та аналізу змін на екрані.

Скріншоти є основним джерелом даних для OCR, аналізу зображень та визначення стану AFK, принцип реалізації модулю зображено на схемі на рисунку 2.7.

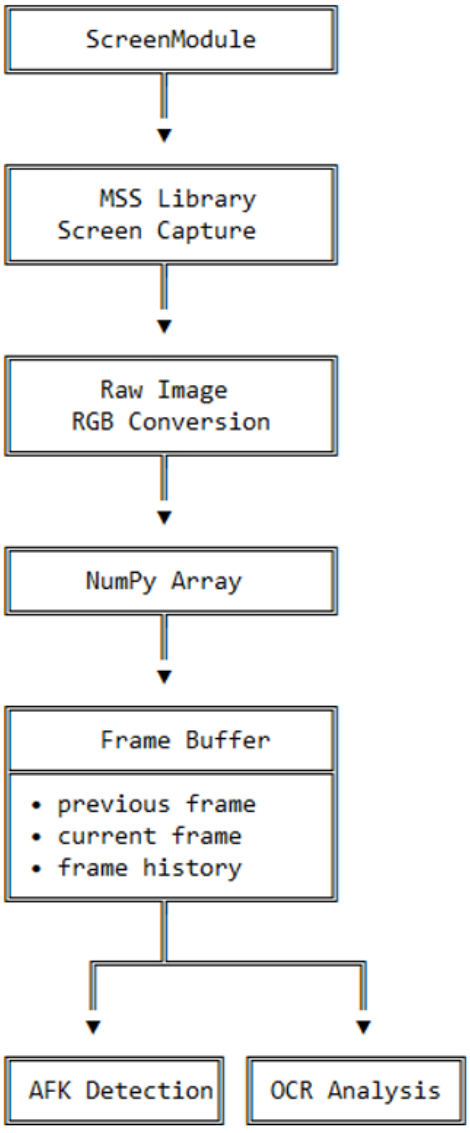

Рисунок 2.7 – Схема роботи модулю ScreenModule

2.4.5 Модуль класифікації активності

ClassifierModule, реалізований у файлі modules/classifier_module.py, є центральним аналітичним компонентом системи.

Модуль виконує такі функції:

− визначення стану неактивності користувача;

− розпізнавання тексту зі скріншотів;

− обчислення візуальних характеристик зображення;

− формування набору ознак;

− класифікацію діяльності користувача.

Для визначення стану AFK поточний скріншот порівнюється з попереднім. Якщо зміни на екрані відсутні протягом заданого проміжку часу, користувач вважається неактивним.

Під час аналізу зображення обчислюються такі характеристики:

− середня яскравість;

− контраст;

− щільність контурів;

− варіативність кольорів;

− рівень змін між послідовними кадрами.

Для розпізнавання тексту використовується бібліотека pytesseract. Отриманий текст аналізується на наявність ключових слів, що можуть вказувати на конкретний тип діяльності.

На основі сукупності ознак модуль визначає категорію активності, наприклад:

− programming;

− study;

− work;

− entertainment;

− communication;

− idle.

Більш детально структуру роботи ClassifierModule зображено на рисунку 2.8.

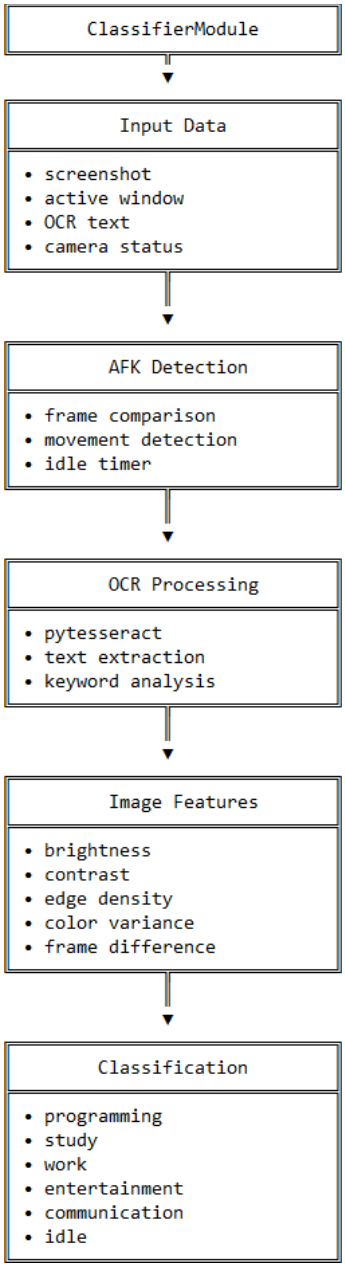

Рисунок 2.8 – Структура роботи ClassifierModule

2.4.6 Модуль роботи з вебкамерою

Модуль CameraModule, реалізований у файлі modules/camera_module.py, використовується для визначення присутності користувача біля комп'ютера.

Якщо модуль активований, програма періодично отримує кадри з вебкамери та аналізує їх. У разі відсутності користувача в полі зору камера може передавати інформацію до системи класифікації, яка позначає

відповідний період як неактивний.

Використання вебкамери дозволяє підвищити точність визначення реальної активності користувача, схему роботи модуля можна побачити на рисунку 2.9.

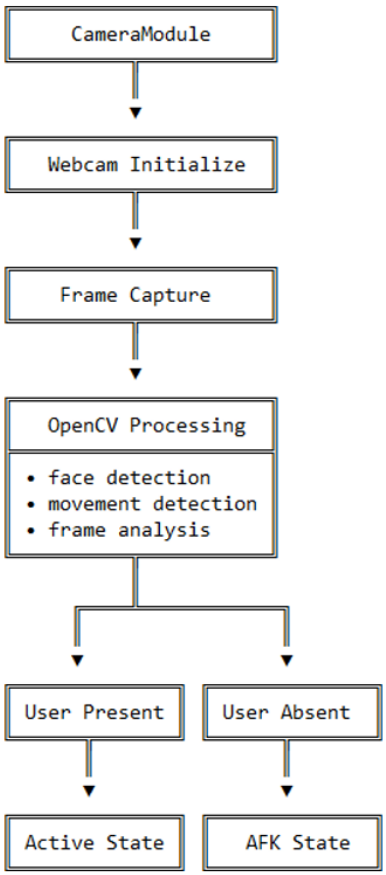

Рисунок 2.9 – Схема роботи модуля CameraModule

2.4.7 Модулі збереження результатів

Для збереження результатів аналізу використовуються два модулі: LoggerModule та DatabaseModule. Структуру зображено на рисунку 2.10.

LoggerModule створює текстові файли журналів у каталозі logs. До журналу записуються дата і час події, назва активного вікна, визначена категорія діяльності, стан AFK та інші допоміжні показники. При кожному новому запуску програми створюється новий log файл, для зручності кожен

створений файл підписується log_dd_mm_yy_h_m.txt, де:

− dd це поточний день;

− mm це поточний місяць;

− yy це поточний рік;

− h години створення файлу;

− m хвилини створення файлу.

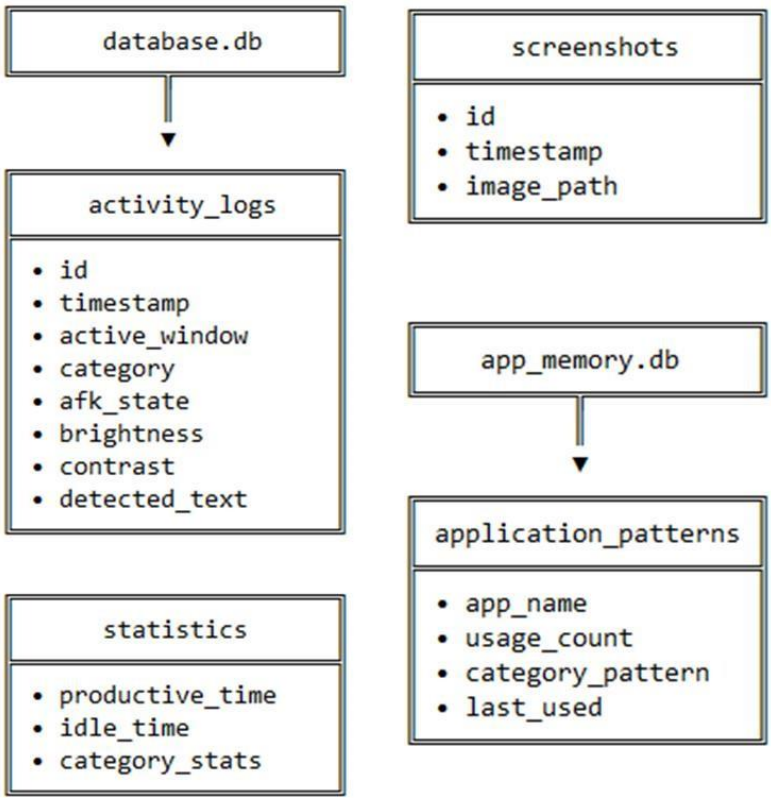

Рисунок 2.10 – Структура таблиць бази даних

Файли журналів можуть використовуватися для налагодження роботи системи та перевірки коректності визначення активності користувача. Такий підхід спрощує аналіз помилок та контроль роботи окремих модулів застосунку.

DatabaseModule забезпечує збереження структурованих даних у базу даних SQLite database.db. Це дозволяє виконувати подальший аналіз інформації та формувати статистичні звіти. Використання SQLite дозволяє забезпечити швидкий доступ до збережених даних без необхідності встановлення окремого серверного програмного забезпечення.

Крім основної бази даних, застосунок використовує файл app_memory.db для збереження інформації про попередню поведінку користувача та накопичені шаблони активності. Збережена інформація може використовуватися для аналізу продуктивності користувача за певний період часу та подальшого вдосконалення алгоритмів класифікації.

2.4.8 Модулі статистики та аналізу журналів

Модуль StatisticsModule виконує обчислення підсумкових показників продуктивності, зокрема тривалості роботи в різних категоріях активності та частки продуктивного часу.

Модуль LogAnalyzerModule дозволяє аналізувати накопичені журнали та формувати статистичні звіти на основі текстових логів і даних із бази даних.

Результати роботи цих модулів можуть використовуватися для побудови графіків, діаграм і зведених таблиць, схему роботи модуля зображено на рисунку 2.11.

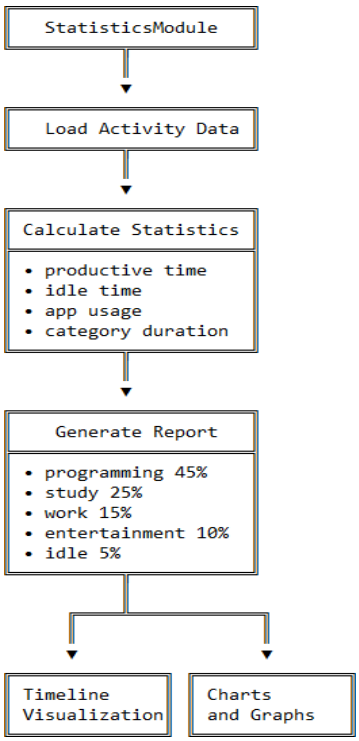

Рисунок 2.11 – Схема модуля звітування

2.4.9 Модуль штучного інтелекту

Модулі ai_module.py та online_ai_client.py забезпечують можливість інтеграції з локальними або онлайн-моделями штучного інтелекту [25].

Їх призначення полягає у вдосконаленні процесу класифікації за рахунок аналізу контексту, розпізнаного тексту та візуальних ознак зображення.

У поточній версії застосунку ці модулі закладають основу для подальшого розвитку системи та впровадження більш складних інтелектуальних алгоритмів.

Таким чином, програмний застосунок Activity Monitor складається з набору взаємопов'язаних модулів, кожен із яких виконує окрему функцію. Модульна архітектура забезпечує гнучкість, масштабованість та зручність подальшого вдосконалення системи.

2.5 Обгрунтування вибору технологій реалізації

Для реалізації програмного застосунку Activity Monitor було обрано мову програмування Python. Даний вибір зумовлений простотою синтаксису, великою кількістю готових бібліотек для обробки зображень, роботи з базами даних, створення графічних інтерфейсів та інтеграції з технологіями штучного інтелекту.

Однією з основних переваг Python є можливість швидкого створення прототипів та поступового розширення функціональності без суттєвого ускладнення програмного коду. Крім того, Python має активну спільноту розробників та добре документовані бібліотеки, що значно спрощує процес розроблення.

Для створення графічного інтерфейсу користувача було використано бібліотеку PySide6 [26]. Вона є офіційним Python-інтерфейсом до фреймворку Qt та надає широкий набір засобів для побудови сучасних настільних

застосунків. За допомогою PySide6 реалізовано головне вікно програми, вікно налаштувань, таблиці статистики та елементи керування процесом моніторингу.

Для створення скріншотів екрана використовується бібліотека mss. Вона забезпечує високу швидкість захоплення зображень та мінімальне навантаження на систему, що є важливим для програм, які працюють у фоновому режимі.

Аналіз зображень виконується за допомогою бібліотеки OpenCV. Вона дозволяє виконувати перетворення зображень, знаходити контури, обчислювати статистичні характеристики та оцінювати зміни між послідовними кадрами.

Для числових обчислень застосовується бібліотека NumPy, яка забезпечує ефективну роботу з багатовимірними масивами та використовується під час аналізу скріншотів.

Розпізнавання тексту реалізовано за допомогою бібліотеки pytesseract [27], яка є Python-інтерфейсом до системи Tesseract OCR [28]. Використання OCR дозволяє отримувати текстову інформацію зі скріншотів та використовувати її під час класифікації активності.

Для визначення активного вікна операційної системи використовується бібліотека PyGetWindow.

Зберігання структурованих даних здійснюється у базі даних SQLite [29]. Даний формат не потребує встановлення окремого серверу та добре підходить для настільних застосунків.

Текстові журнали роботи програми створюються за допомогою стандартних засобів Python. Це дозволяє зберігати повну історію подій та використовувати її для подальшого аналізу.

Для роботи з часовими інтервалами, потоками виконання та конфігураційними файлами використовуються стандартні модулі Python [30]: time, threading, json, os та pathlib.

Отже, обраний набір технологій забезпечує необхідний функціонал для

простої реалізації системи аналізу активності користувача та створює основу для її подальшого вдосконалення шляхом додавання нових модулів або покращення та оптимізації існуючих.

Більш детально використані технології та їх призначення розглянуто у таблиці 2.1.

Таблиця 2.1 – Використані технології та їх призначення

Технологія	Призначення
Python 3	Основна мова програмування для реалізації всіх компонентів застосунку
PySide6	Розроблення графічного інтерфейсу користувача
Qt	Базовий фреймворк, на основі якого працює PySide6
mss	Створення скріншотів екрана
OpenCV (cv2)	Обробка зображень, обчислення візуальних характеристик
NumPy	Числові обчислення та робота з масивами даних
pytesseract	Розпізнавання тексту зі скріншотів (OCR)
Tesseract OCR	Система оптичного розпізнавання символів
PyGetWindow	Отримання інформації про активне вікно операційної системи
SQLite	Локальне зберігання структурованих даних
JSON	Зберігання конфігураційних параметрів у файлі settings.json
threading	Виконання моніторингу в окремому потоці

Продовження таблиці 2.1

Технологія	Призначення
time	Робота з часовими інтервалами та мітками часу
hashlib	Обчислення хешів для визначення стану AFK
logging / файловий запис	Створення текстових журналів роботи програми
pathlib, os	Робота з файлами та каталогами
OpenAI API (опціонально)	Додатковий аналіз активності за допомогою моделей штучного інтелекту

2.6 Структура проєкту та організація даних

Програмний застосунок Activity Monitor має ієрархічну структуру каталогів, яка забезпечує логічне розділення програмного коду, конфігураційних файлів, баз даних та результатів роботи. Структуру застосунку зображено на рисунку 2.12.

У кореневому каталозі проєкту розміщено файли main.py, settings.json, database.db та app_memory.db.

Файл main.py є точкою входу до програми. У ньому виконується створення об’єкта графічного застосунку та запуск головного вікна.

Файл settings.json містить параметри конфігурації, такі як інтервал моніторингу, налаштування OCR, параметри бази даних та прапорці увімкнення окремих модулів.

Файл database.db є основною базою даних SQLite, у якій зберігаються результати моніторингу.

Файл app_memory.db використовується для накопичення інформації про попередню активність користувача.

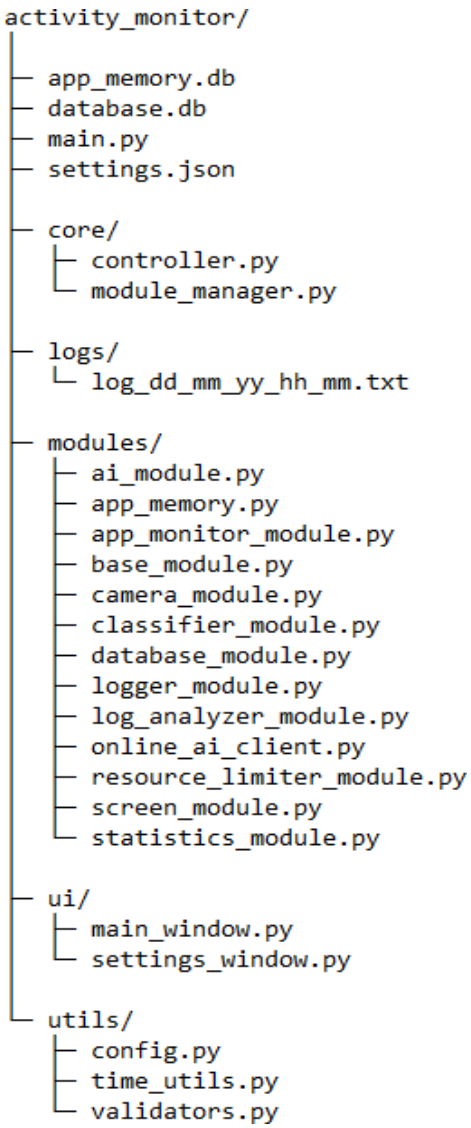

Рисунок 2.12 – Структура застосунку Activity Monitor

Каталог logs призначений для зберігання текстових журналів роботи програми.

Такий розподіл файлів забезпечує зручність супроводу проєкту та спрощує пошук необхідних компонентів.

3 ЕКСПЕРИМЕНТАЛЬНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ АНАЛІЗУ АКТИВНОСТІ КОРИСТУВАЧА

3.1 Обмеження проєкту

Оскільки програма буде працювати у фоновому режимі інтерфейс програми повинен бути простим та водночас ефіктивним. Для зручності використання усі головні функції повинні бути доступні з головного вікна.

Саме тому у головному вікні повинні бути кнопки «Start» та «Stop» які відповідають за початок та зупинення роботи програми. Також у швидкому доступі повинні бути кнопки закриття програми, згортання та переходу до налаштувань.

Також до головної сторінки програми потрібно винести вивід вебкамери (якщо вона увімкнена), показник присутності людини та показник роботи програми. Показник присутності потрібно розташувати біля виводу відео з вебкамери, а маркер роботи застосунку потрібно зробити мінімалістичним так, щоб не заважав користувачу. На приклад зробити його у якості рамки, що змінює колір. Синій значить працює, червоний – ні.

Приклад реалізованого головного вікна програми зображено на рисунках 3.1 та 3.2. На прикладах можна побачити що у вікні всього присутні всього шість кнопок (Start, Stop, дві кнопки налаштувань, кнопка згорнення програми та кнопка закриття), назва самої програми та покажчики роботи програми та присутності людини.

Також реалізовано зміну кольору рамки програми при увімкненні. А вебкамера використовується лише з дозволу користувача та тільки при зчитуванні активності, тому коли програму вимкнена відео з неї недоступно, отже місце виводу залишається порожнім. У якості заглушки використано надпис «Camera disabled».

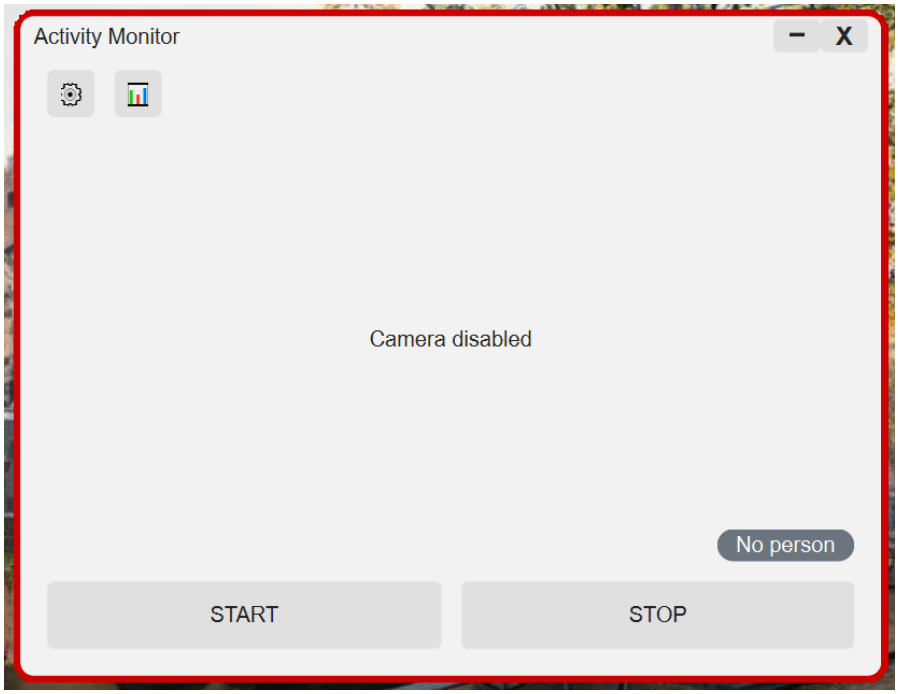

Рисунок 3.1 – Вікно застосунку у вимкненому стані

Окрім головного вікна застосунку потрібно розробити зручні способи взаємодії користувача з налаштуваннями. Для цього потрібно зробити вікно налаштувань, доступ до якого буде відбуватися з основного меню натисненням кнопки налаштувань.

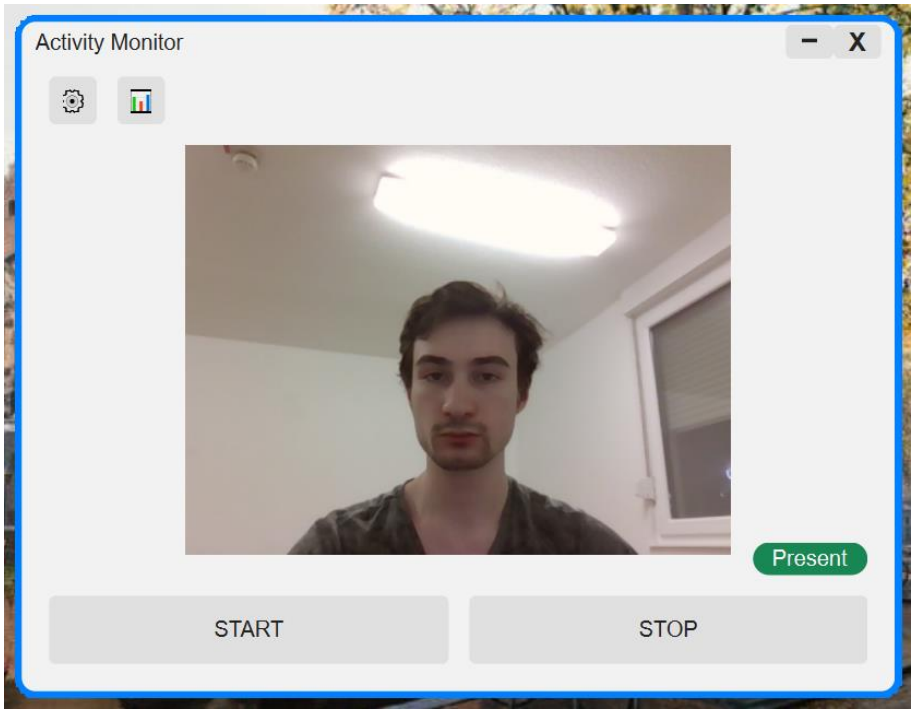

Рисунок 3.2 – Вікно застосунку в увімкненому стані

У меню налаштувань потрібно реалізувати взаємодію користувача з усіма необхідними функціями. До них входять:

− зміна частоти знімків екрану та виведення поточної частоти;

− дозвіл на використання вебкамери;

− дозвіл на звернення до більш просунутого ШІ у мережі інтернет;

− поля для введення ключів більш просунутого ШІ;

− можливість ввімкнути/вимкнути згортання до області сповіщень за бажанням користувача;

− зміну теми застосунку з світлої на темну за бажанням користувача;

− вибір та кнопка автоналаштування камери, а також можливість її відзеркалити;

− кнопка для аналізу журналів;

− кнопки повернення до головного меню та збереення налаштувань.

Для зручності вводу частоти знімків екрану використано слайдер з послідовною зміною інтервалів. Від 1 до 60 збільшується на 1, а від 60 до 300 секунд збільшення вже відбувається на 5 одиниць. Щоб пришвидшити введення поруч зі слайдером було розміщено вікно для вводу числа. При введення застосунок автоматично округлює число до найбжтжчого дозволеного значення. Інтерфейс вікна налаштувань з усіма необхідними засобами для взаємодії зображено на рисунку 3.3.

Для аналізу журналів використовується окреме вікно, доступ до якого користувач може отримати з меню налаштувань, за допомого. Кнопки «Analyze logs».

При натисненні кнопки відкривається власне меню, у якому користувач може обрати журнали, які хоче проаналізувати. При натисненні лівої кнопки миші обирається один журнал, при натисненні сполучення клавіш «Shift» та лівої кнопки миші обираються усі журнали між обраним попередньо та тим, що було натиснуто зараз. Також є можливість обрати усі достуні журнати, для цього розроблено кнопку «Select all». Інтерфейс вікна можна побачити на рисунку 3.4.

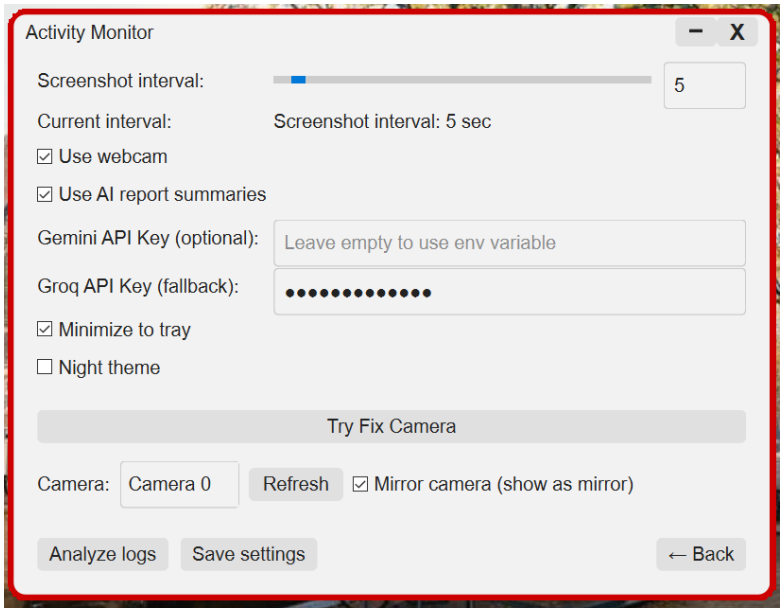

Рисунок 3.3 – Вікно налаштувань застосунку

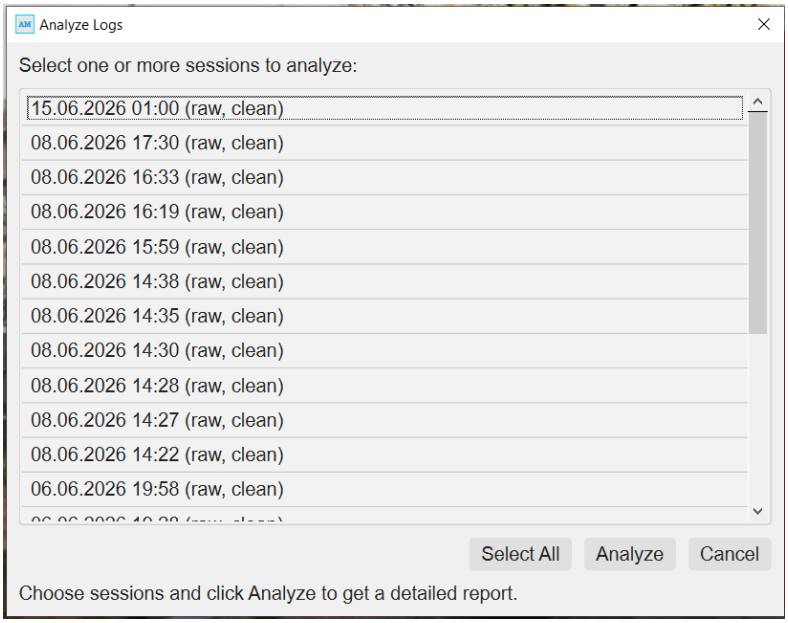

Рисунок 3.4 – Інтерфейс вікна вибору журналів

3.2 Особливості реалізації модулів

Практична реалізація програмного застосунку Activity Monitor базується на принципах модульного об’єктно-орієнтованого програмування із чітким розділенням зон відповідальності між окремими компонентами системи. Головною метою проєктування архітектури було створення гнучкого,

розширюваного та стійкого до відмов інструменту, здатного функціонувати у фоновому режимі з мінімальним залученням апаратних ресурсів.

Програмний комплекс розроблено на мові програмування Python з використанням набору спеціалізованих низькорівневих системних бібліотек, засобів комп’ютерного зору та систем розпізнавання тексту.

3.2.1 Реалізація модуля керування системою

Модуль керування системою реалізовано у файлі controller.py у вигляді класу Controller. Він є центральним компонентом програмного застосунку Activity Monitor та забезпечує координацію роботи всіх функціональних модулів системи. Через даний клас виконується запуск моніторингу, створення циклів аналізу, взаємодія між модулями та збереження отриманих результатів.

Під час створення об’єкта Controller відбувається завантаження конфігурації із файлу налаштувань. Конфігурація містить параметри інтервалу моніторингу, тривалості визначення стану AFK, параметри роботи OCR та інші службові налаштування. Використання окремого конфігураційного файлу дозволяє змінювати поведінку програми без модифікації вихідного коду.

Для зберігання історії станів екрана використовується кільцевий буфер на основі структури deque. Розмір буфера визначається автоматично відповідно до інтервалу моніторингу. Приклад реалізації у проєкті наведено у лістингу 3.1.

Лістинг 3.1 Ініціалізація буфера історії кадрів:

```
interval = max(
    1.0,
    float(
```

```
            self.config.get(
                "frame_interval_seconds",
                5
            )
        )
    )
    buffer_len = max(
        12,
        int(round(60.0 / interval)) + 2
    )
    self.screen_buffer = deque(
        maxlen=buffer_len
    )
```

Даний підхід дозволяє зберігати приблизно одну хвилину історії змін екрана незалежно від встановленого інтервалу моніторингу. Використання буфера фіксованого розміру запобігає неконтрольованому збільшенню використання оперативної пам’яті.

Для забезпечення безперервної роботи програми без блокування графічного інтерфейсу використовується окремий потік виконання. Після запуску моніторингу створюється фоновий потік, у якому виконується основний цикл аналізу. Приклад реалізації у проєкті наведено у лістингу 3.2.

Лістинг 3.2 Створення потоку моніторингу:

```
    self.thread = threading.Thread(
        target=self.run_loop,
        daemon=True
    )
    self.thread.start()
```

Використання окремого потоку дозволяє виконувати операції OCR, аналізу зображень та запису даних без впливу на роботу користувацького інтерфейсу.

Основна логіка роботи реалізована методом run_loop(), який виконує періодичний виклик методу run_cycle(). Кожна ітерація циклу відповідає одному повному етапу моніторингу активності. Приклад реалізації у проєкті наведено у лістингу 3.3.

Лістинг 3.3 Основний цикл роботи контролера:

```
while self.running:
    interval = self.config.get(
        "frame_interval_seconds",
        5
    )
    self.run_cycle()
    time.sleep(interval)
```

Під час виконання run_cycle() система створює скріншот екрана, визначає активне вікно, виконує класифікацію діяльності користувача та передає результати до модулів журналювання і збереження даних.

Таким чином, Controller виступає центральним координатором усіх процесів системи та забезпечує узгоджену взаємодію між її компонентами.

3.2.2 Реалізація модуля моніторингу активного вікна

Отримання інформації про активне вікно реалізовано у файлі app_monitor_module.py за допомогою класу AppMonitorModule. Даний модуль відповідає за визначення програми, з якою користувач працює в поточний момент часу.

Для доступу до інформації про активні вікна використовується бібліотека PyGetWindow. Під час кожного циклу моніторингу система звертається до операційної системи та отримує заголовок поточного активного вікна.

З метою зменшення кількості системних викликів реалізовано механізм кешування результатів. Якщо між двома запитами проходить менше визначеного часу, система повертає попередньо отриманий результат. Приклад реалізації у проєкті наведено у лістингу 3.4.

Лістинг 3.4 Перевірка кешованого результату:

```
now = time.time()
if now - self._last_check_time < 0.5:
    return self._last_window
```

Завдяки такому підходу значно зменшується навантаження на систему під час частих звернень до модуля.

Отримання назви активного вікна виконується за допомогою функцій бібліотеки PyGetWindow. Приклад реалізації у проєкті наведено у лістингу 3.5.

Лістинг 3.5 Отримання активного вікна:

```
window = gw.getActiveWindow()
if window:
    self._last_window = (
        window.title.lower()
    )
else:
    self._last_window = ""
```

Після отримання назви вікна виконується перетворення тексту до нижнього регістру, що спрощує подальший пошук ключових слів та

порівняння назв програм.

Особливістю реалізації є наявність механізму обробки помилок. У випадку неможливості отримання інформації про активне вікно система повертає порожній рядок, що дозволяє уникнути аварійного завершення роботи програми.

Результат роботи AppMonitorModule використовується модулем класифікації як одна з головних ознак визначення поточного типу діяльності користувача.

3.2.3 Реалізація модуля створення скріншотів

Для отримання інформації про поточний стан екрана використовується модуль ScreenModule, реалізований у файлі screen_module.py. Даний модуль забезпечує захоплення зображення робочого столу та передачу отриманих даних іншим компонентам системи.

Для створення скріншотів використовується бібліотека MSS. Дана бібліотека забезпечує високу швидкість отримання кадрів та низьке навантаження на систему порівняно з альтернативними рішеннями.

Процес створення скріншота реалізовано таким чином. Приклад реалізації у проєкті наведено у лістингу 3.6.

Лістинг 3.6 Отримання зображення екрана:

```
with mss.mss() as sct:
    monitor = sct.monitors[1]
    screenshot = sct.grab(
        monitor
    )
```

Бібліотека безпосередньо взаємодіє з буфером екрана операційної системи, що дозволяє отримувати зображення швидше за традиційні методи захоплення.

Отримане зображення надалі перетворюється у формат, придатний для обробки засобами OpenCV та NumPy. Приклад реалізації у проєкті наведено у лістингу 3.7.

Лістинг 3.7 Конвертація скріншота:

```
img = Image.frombytes(
    "RGB",
    screenshot.size,
    screenshot.rgb
return np.array(img)
```

Після конвертації зображення представляється у вигляді багатовимірного масиву NumPy. Такий формат є стандартним для більшості бібліотек комп’ютерного зору та машинного аналізу.

Одержані скріншоти використовуються одразу декількома підсистемами. Модуль OCR виконує розпізнавання тексту, модуль класифікації аналізує візуальні характеристики інтерфейсу, а механізм AFK порівнює поточне зображення з попередніми кадрами для визначення активності користувача.

Важливою перевагою реалізованого рішення є відсутність необхідності збереження скріншотів на диск. Усі операції виконуються безпосередньо в оперативній пам’яті, що підвищує швидкодію системи та зменшує навантаження на накопичувач.

Таким чином, ScreenModule забезпечує ефективне отримання графічної інформації та формує основу для роботи всіх механізмів аналізу активності користувача.

3.2.4 Реалізація модуля класифікації активності

Найбільш складним компонентом програмного застосунку є модуль класифікації активності користувача, реалізований у файлі classifier_module.py. Основним завданням даного модуля є аналіз інформації, отриманої від інших компонентів системи, та визначення поточного типу діяльності користувача.

Модуль використовує комбінований підхід до класифікації, який поєднує аналіз активного вікна, розпізнавання тексту, комп'ютерний зір та механізми визначення активності користувача. На відміну від простих систем моніторингу, що аналізують лише назву активної програми, розроблене рішення використовує декілька незалежних джерел даних, що дозволяє підвищити точність визначення діяльності.

Першим етапом роботи модуля є перевірка активності користувача. Для цього використовується механізм AFK-аналізу, який базується на порівнянні поточного скріншота з попередніми кадрами.

Для пришвидшення аналізу використовується механізм хешування зображень. Замість порівняння кожного пікселя система формує компактний MD5-хеш кадру. Приклад реалізації у проєкті наведено у лістингу 3.8.

Лістинг 3.8 Формування хешу зображення:

```
small = img[::20, ::20]
return hashlib.md5(
    small.tobytes()
).hexdigest()
```

Перед обчисленням хешу зображення додатково зменшується шляхом вибірки кожного двадцятого пікселя. Це дозволяє суттєво скоротити обсяг даних та прискорити процес аналізу без значного впливу на точність визначення змін.

Після отримання хешу система перевіряє, чи змінювався екран протягом останнього часу. Якщо зміни відсутні, починається підрахунок тривалості неактивності. Приклад реалізації у проєкті наведено у лістингу 3.9.

Лістинг 3.9 Визначення стану AFK:

```
idle = time.time() - self.last_change_time
if idle > 60:
    return True
```

Якщо протягом встановленого часу зміни на екрані не виявляються, користувач вважається неактивним. Отриманий результат використовується під час формування підсумкової оцінки діяльності.

Наступним етапом є аналіз текстової інформації. Для цього використовується система оптичного розпізнавання символів Tesseract OCR. Приклад реалізації у проєкті наведено у лістингу 3.10.

Лістинг 3.10 Отримання тексту зі скріншота:

```
text = pytesseract.image_to_string(
    img
)
return text.lower()
```

Після розпізнавання текст переводиться до нижнього регістру, що спрощує подальший пошук ключових слів незалежно від особливостей написання.

Отриманий текст використовується для пошуку ознак навчальної, робочої або розважальної діяльності. Для кожної категорії використовується окремий набір ключових слів. Приклад реалізації у проєкті наведено у лістингу 3.11.

Лістинг 3.11 Приклад словника ключових слів:

```
study_words = [
    "lecture",
    "theorem",
    "chapter",
    "exercise"]
work_words = [
    "code",
    "function",
    "class",
    "import"
]
```

Під час аналізу система підраховує кількість знайдених збігів та формує початкову оцінку ймовірності належності до певної категорії.

Крім текстового аналізу виконується аналіз візуальних характеристик зображення. Для цього використовується бібліотека OpenCV.

На першому етапі кадр переводиться у відтінки сірого. Приклад реалізації у проєкті наведено у лістингу 3.12.

Лістинг 3.12 Перетворення зображення:

```
gray = cv2.cvtColor(
    img,
    cv2.COLOR_BGR2GRAY
)
```

Отримане зображення використовується для подальшого аналізу контурів та статистичних характеристик.

Для визначення складності інтерфейсу застосовується алгоритм пошуку

контурів Canny. Приклад реалізації у проєкті наведено у лістингу 3.13.

Лістинг 3.13 Визначення контурів:

```
edges = cv2.Canny(
    gray,
    50,
    150
)
```

Щільність знайдених контурів використовується як одна з ознак під час класифікації. Наприклад, редактори програмного коду зазвичай мають велику кількість контурів та текстових елементів, тоді як відеоконтент характеризується більш плавними переходами кольорів.

Після завершення аналізу формується набір числових характеристик. Приклад реалізації у проєкті наведено у лістингу 3.14.

Лістинг 3.14 Формування набору ознак:

```
return {
    "edges": float(edge_density),
    "brightness": float(brightness),
    "colors": float(color_var)
}
```

У результаті роботи модуля всі отримані ознаки об’єднуються в єдину систему оцінювання. Для кожної категорії діяльності формується рейтинг, який враховує результати OCR, аналіз активного вікна, характеристики зображення та інформацію про активність користувача.

Категорія з найбільшим рейтингом вважається найбільш імовірною та використовується як підсумковий результат класифікації. Разом із категорією система зберігає рівень впевненості, який надалі використовується для

статистичного аналізу.

Таким чином, ClassifierModule реалізує багаторівневий механізм аналізу активності користувача та є основним інтелектуальним компонентом системи.

3.2.5 Реалізація модуля роботи з базою даних

Для довготривалого зберігання результатів роботи використовується модуль DatabaseModule. У якості системи керування базами даних обрано SQLite, що дозволяє використовувати локальну базу без встановлення додаткового серверного програмного забезпечення.

Під час запуску застосунку модуль автоматично створює необхідні таблиці, якщо вони відсутні у файлі бази даних. Приклад реалізації у проєкті наведено у лістингу 3.15.

Лістинг 3.15 Створення таблиці подій:

```
CREATE TABLE IF NOT EXISTS events (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT NOT NULL,
    state TEXT NOT NULL,
    afk INTEGER NOT NULL,
    work INTEGER NOT NULL,
    study INTEGER NOT NULL,
    entertainment INTEGER NOT NULL,
    diff REAL
)
```

Така структура дозволяє накопичувати історію роботи користувача та використовувати її для подальшого аналізу.

Для додавання нових записів використовується параметризований SQL-

запит. Приклад реалізації у проєкті наведено у лістингу 3.16.

Лістинг 3.16 Додавання запису до бази даних:

```
self.cursor.execute(
    """
    INSERT INTO events
    (
        timestamp,
        state,
        afk,
        work,
        study,
        entertainment,
        diff
    )
    VALUES (?, ?, ?, ?, ?, ?, ?)
    """
)
```

Після виконання запиту результати записуються до бази даних та стають доступними для подальшого статистичного аналізу.

Окремою особливістю реалізації є використання додаткової бази app_memory.db, яка застосовується для кешування результатів аналізу та зберігання допоміжної інформації, необхідної для роботи інтелектуальних алгоритмів.

Використання SQLite забезпечує високу швидкість роботи та мінімальне споживання ресурсів системи.

3.2.6 Реалізація модуля журналювання

Для ведення історії роботи застосунку використовується модуль LoggerModule. Його основним призначенням є збереження інформації про результати моніторингу та службові події системи.

Під час кожного циклу аналізу формується текстовий запис журналу, який містить інформацію про поточний стан користувача та результати класифікації. Приклад реалізації у проєкті наведено у лістингу 3.17.

Лістинг 3.17 Формування запису журналу:

```
line = (
    f"[{timestamp}] "
    f"STATE:{probs.get('state', 'unknown')} "
    f"AFK:{probs.get('afk', 0)} "
    f"WORK:{probs.get('work', 0)} "
    f"STUDY:{probs.get('study', 0)} ")
```

У сформованому записі містяться дата та час події, визначена категорія діяльності та числові оцінки окремих категорій.

Для побудови спрощених звітів система додатково визначає категорію з найбільшим рівнем впевненості. Приклад реалізації у проєкті наведено у лістингу 3.18.

Лістинг 3.18 Визначення основної категорії:

```
candidates = [
    ("work", int(probs.get("work", 0))),
    ("study", int(probs.get("study", 0))),
    ("entertainment", int(probs.get("entertainment", 0)))
]
```

Журнали зберігаються у вигляді текстових файлів у каталозі logs. Для

кожного дня створюється окремий файл, що спрощує пошук інформації та аналіз роботи програми.

Використання журналювання значно спрощує тестування та налагодження системи, оскільки дозволяє відстежувати послідовність виконання всіх операцій.

3.2.7 Реалізація модуля штучного інтелекту

Для підвищення точності аналізу активності у проєкті реалізовано окремий модуль штучного інтелекту, представлений файлами ai_module.py та online_ai_client.py.

Основним призначенням AIModule є використання додаткових механізмів аналізу даних та скорочення кількості повторних обчислень.

Перед виконанням аналізу система перевіряє наявність раніше отриманих результатів для аналогічних зображень. Приклад реалізації у проєкті наведено у лістингу 3.19.

Лістинг 3.19 Перевірка кешу результатів:

```
screenshot_hash = (
    self.memory
    .get_screenshot_hash(
        screenshot
    )
)
cached_result = (
    self.memory
    .get_cached_screenshot_result(
        screenshot_hash
    )
```

```
)
```

Якщо результат уже присутній у кеші, система може використати його повторно без запуску повного циклу аналізу.

Для швидкої класифікації застосовуються евристичні правила, які аналізують назву активного вікна. Приклад реалізації у проєкті наведено у лістингу 3.20.

Лістинг 3.20 Визначення розважальної активності:

```
if any(
    k in title
    for k in [
        "youtube",
        "netflix",
        "twitch"
    ]
):
    return {
        "work": 0,
        "study": 0,
        "entertainment": 100
    }
```

Подібні правила дозволяють швидко визначати очевидні типи діяльності без виконання складного аналізу зображення та тексту.

Окремий модуль online_ai_client.py забезпечує можливість інтеграції із зовнішніми сервісами штучного інтелекту. Завдяки цьому архітектура системи залишається відкритою для подальшого розвитку та інтеграції сучасних мовних моделей.

Таким чином, AIModule створює основу для впровадження

інтелектуальних алгоритмів аналізу та підвищення точності класифікації активності користувача в майбутніх версіях програмного застосунку.

3.3 Демонстрація роботи програми

Робота програмного застосунку демонструється на прикладі реального сеансу моніторингу активності користувача. Для перевірки коректності функціонування програми було виконано тестовий запуск, під час якого користувач послідовно виконав усі види активності, а саме: розваги, навчання, роботу та демонстрацію відсутності активності. Для тестового запуску використовувалися такі налаштування:

− інтервал між знімками екрану – 5 секунд;

− камера №1;

− світла тема застосунку.

Після запуску застосунок ініціалізує головне вікно та відновлює всі збережені налаштування з минулої сессії. Користувач може розпочати або зупинити моніторинг за допомогою кнопок «Start» та «Stop» відповідно, а також перейти до налаштувань або завершити роботу програми Головне вікно після запуску зображено на рисунку 3.5.

Після натискання кнопки «Start» застосунок починає аналіз активності, отримані результати класифікації програма записує у два журнали, чистий та технічний. Для кожної сессії створюється окремі журнали у папці «logs», це допомагає запобігати переповненню журналів та винекненню завеликих файлів. Зображення змісту папки зображено на рисунку 3.6.

Під час роботи програми у головному вікні з’являється зображення з обраної камери, якщо користувач дав згоду на її використання, а також рамка змінює колір з червоного на синій. Це свідчить про початок безпосереднього аналізу, це зображено на рисунку 3.7.

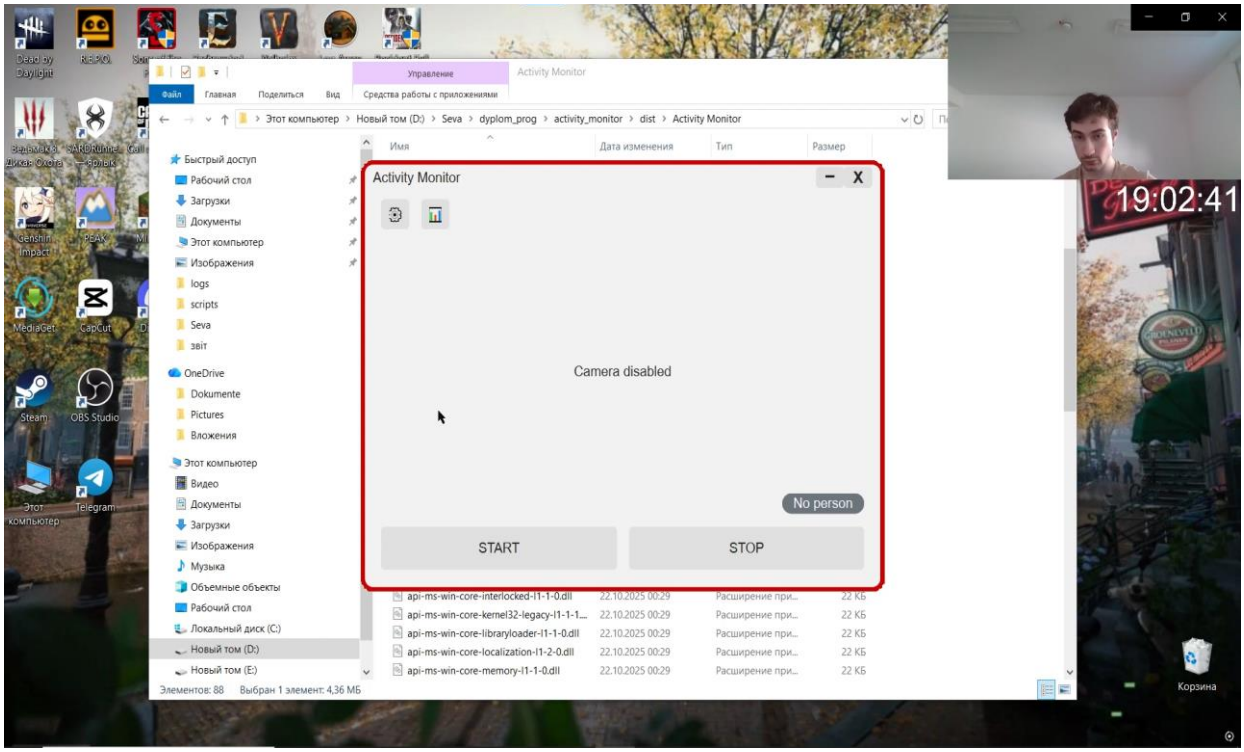

Рисунок 3.5 – Головне вікно програмного застосунку після запуску

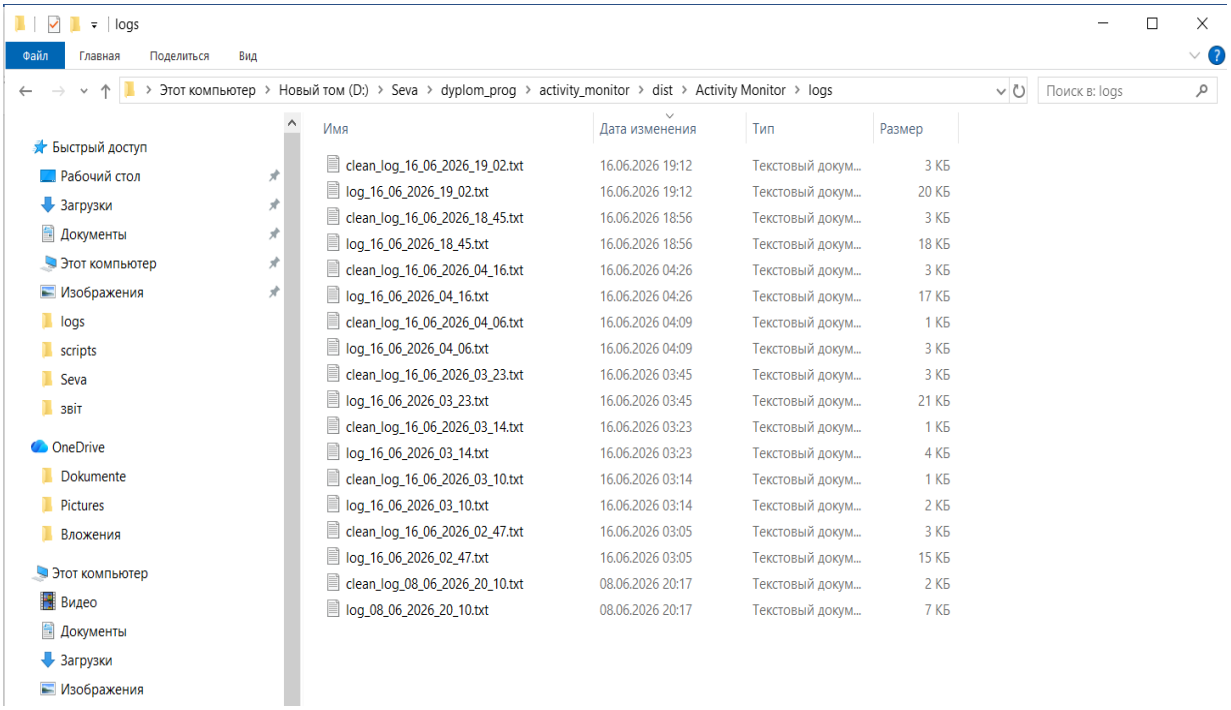

Рисунок 3.6 – Зміст папки з журналами

Під час роботи програми у головному вікні з’являється зображення з обраної камери, якщо користувач дав згоду на її використання, а також рамка змінює колір з червоного на синій. Це свідчить про початок безпосереднього

аналізу, це зображено на рисунку 3.7.

Під час роботи система отримує інформацію про активне вікно операційної системи та використовує її як одне з джерел даних для класифікації діяльності.

У процесі моніторингу система виконує аналіз заголовків вікон, тексту на екрані та візуальних характеристик зображення. На основі отриманих даних формується поточна категорія діяльності користувача.

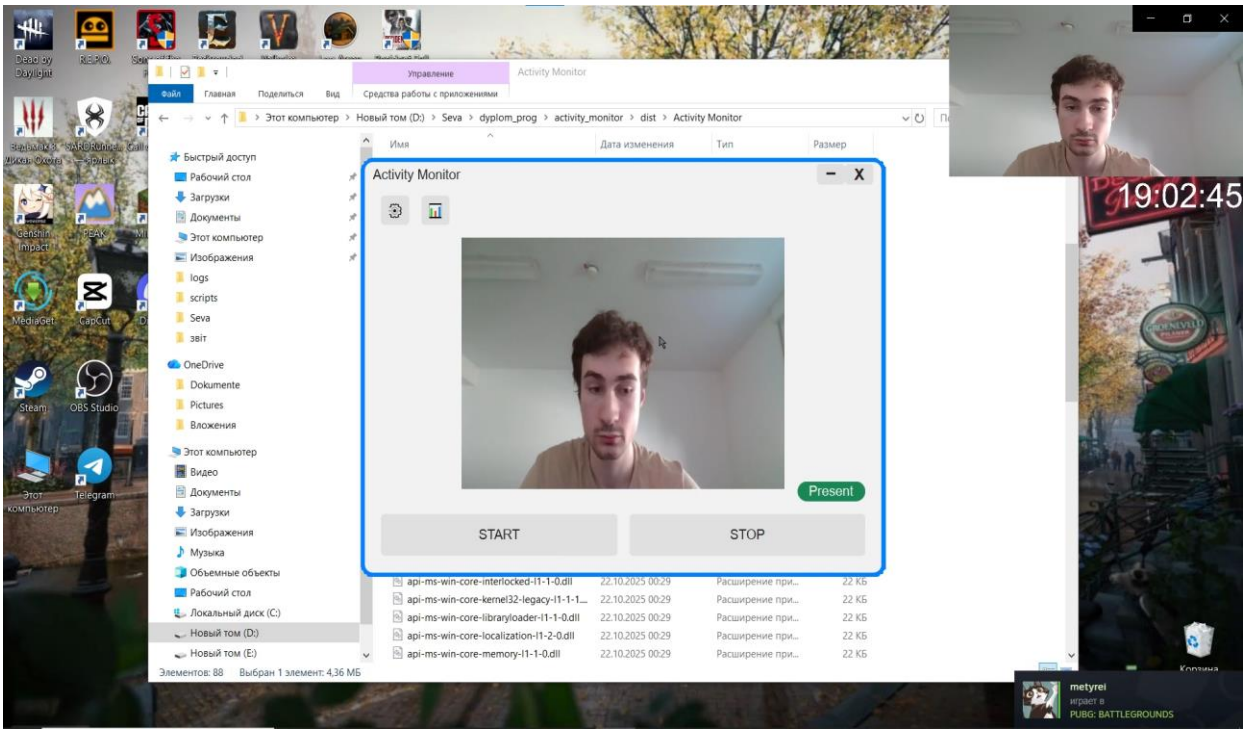

Рисунок 3.7 – Головне вікно програмного застосунку під час аналізу

Як зазначалося вище, усі результати аналізу автоматично записуються до двох журналів. У «чистому» журналі міститься лише часова позначка та головний тип визначенної активності, а у «технічному» також містяться технічні дані такі як назва застосунку що використовується, значення усів типів активності та інша інформація, необхідна для подальшого аналізу на рисунку 3.8 можна побачити зміст «чистого» журналу, а на рисунку 3.9 зображено технічний журнал.

Також у користувача є можливість отримати звіт по його активності безпосередньо у застосунку. Для цього є окрема кнопка у вікні налаштувань

«Analyze logs».

Після натискання користувач може обрати журнали, по яким бажає отримати звіт, та при натисканні кнопки «Analyze» отримати звіт з графіками та рекомендаціями від ШІ. Приклад рекомендацій можна побачити на рисунку 3.10, а приклад стовбчатої діаграми на рисунку 3.11.

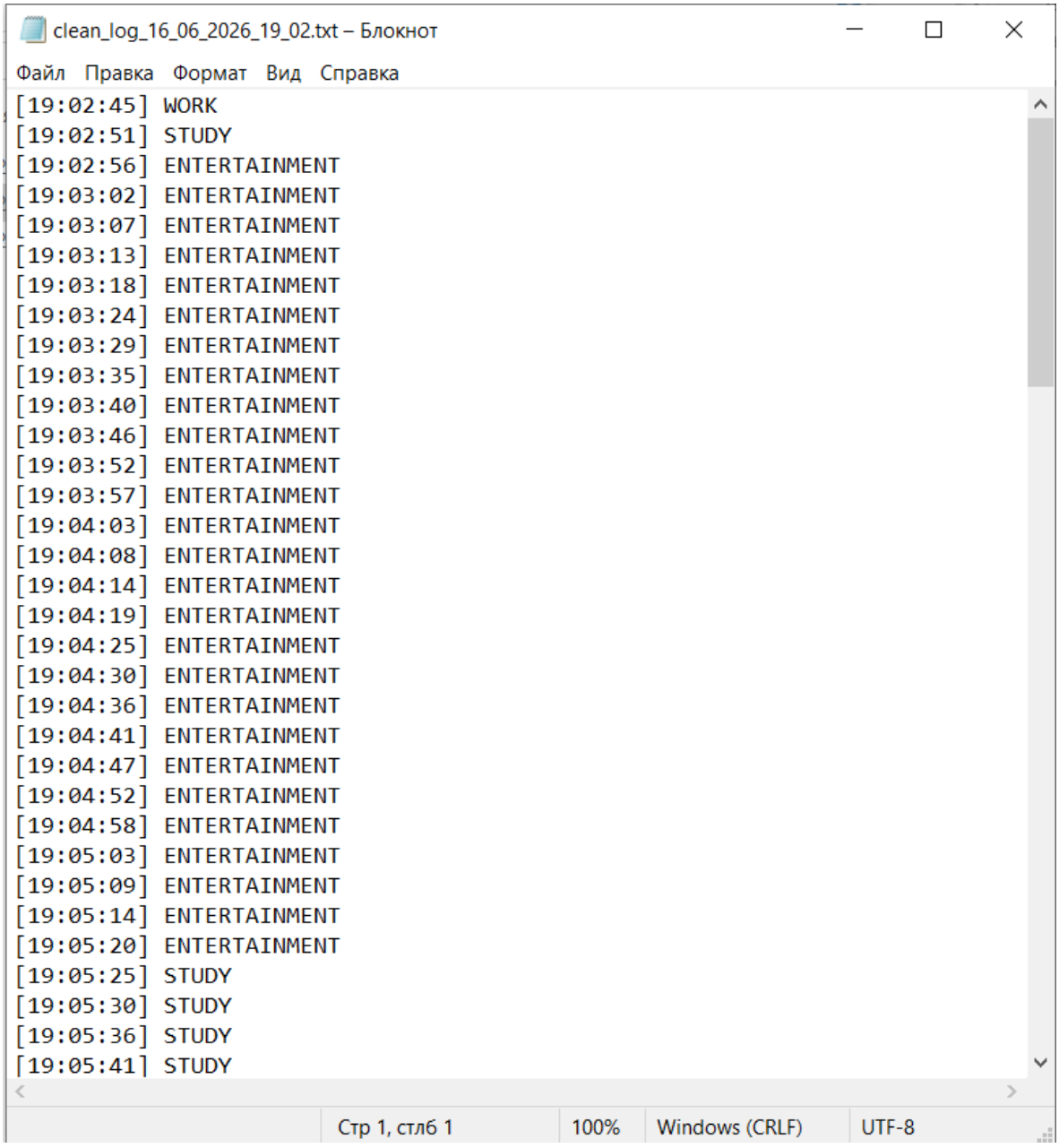

Рисунок 3.8 – Зміст спрощеного журналу

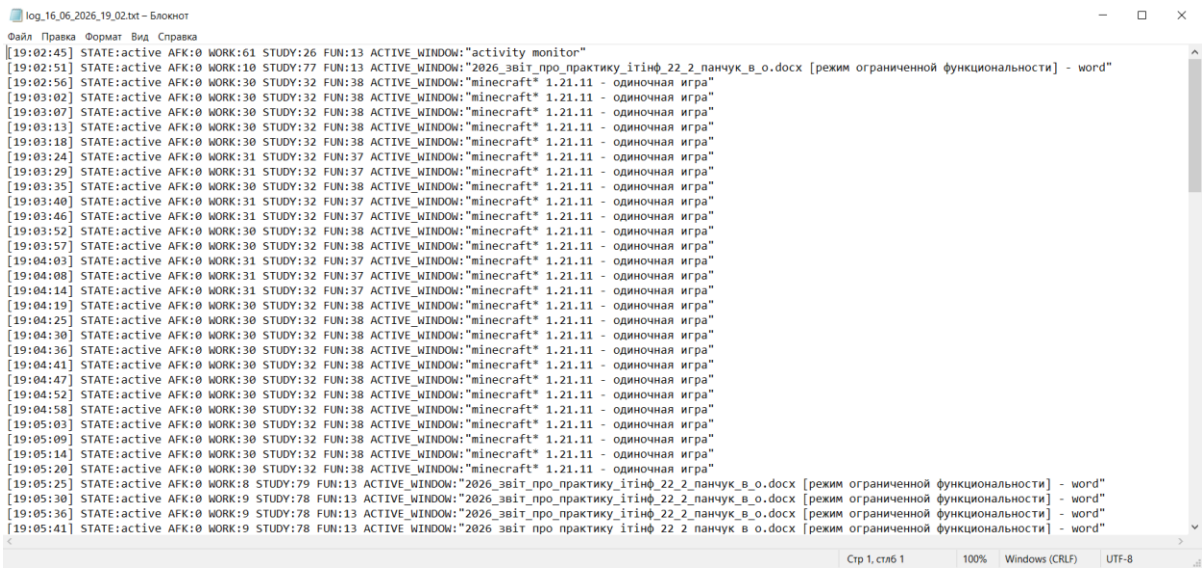

Рисунок 3.9 – Зміст технічного журналу

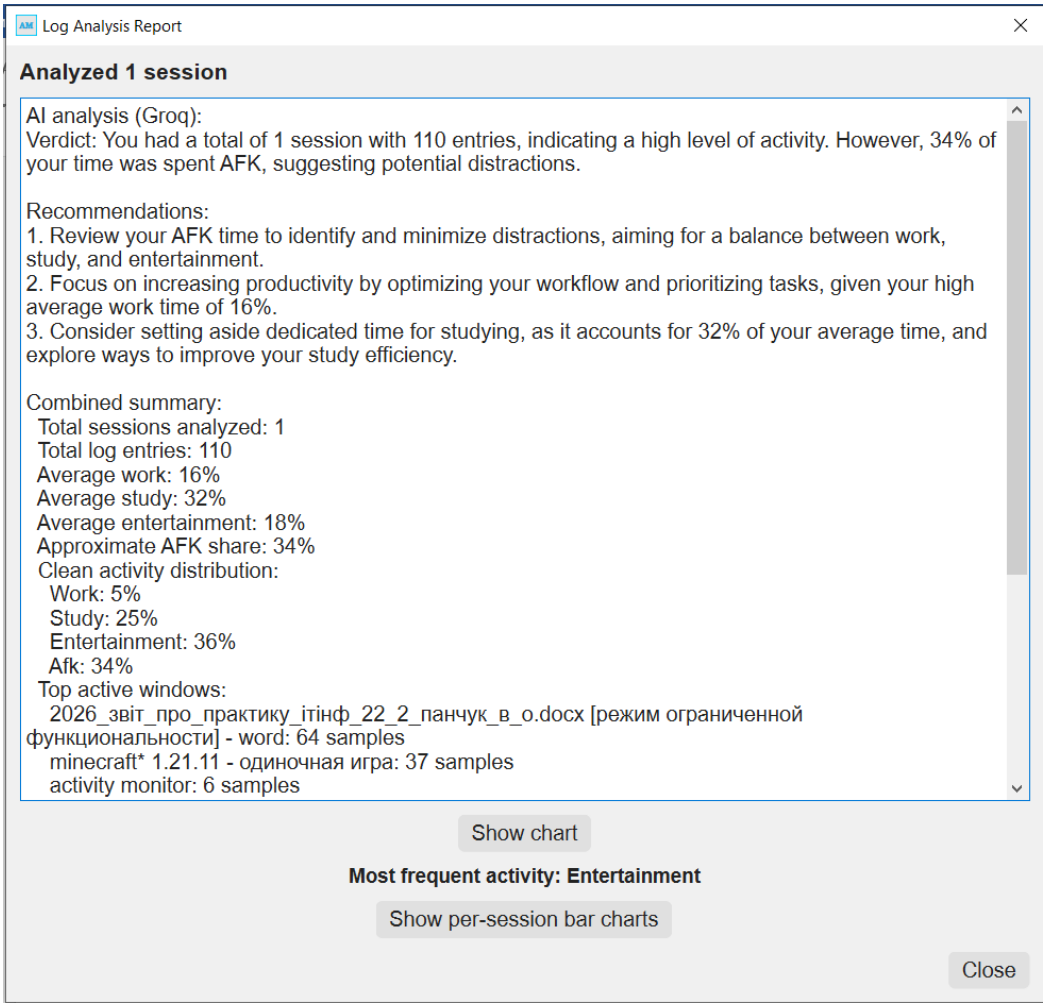

Рисунок 3.10 – Результат аналізу журнала штучним інтелектом

Рисунок 3.11 – Стовбчата діаграма побудована за данними з журналу

3.4 Порівняння очікуваних та фактичних результатів, оцінка якості

Для оцінки якості розробленого програмного застосунку було проведено перевірку реалізації функціональних вимог, сформульованих на етапі постановки задачі. Перевірка виконувалася шляхом аналізу журналів роботи програми та експериментального тестування системи.

Під час тестування одночасно здійснювався запис відео робочого столу користувача, відеопотоку з вебкамери та поточного часу. Відеозапис використовувався як еталонне джерело даних, що дозволяло визначати фактичний тип активності користувача в кожний момент часу та порівнювати його з результатами автоматичної класифікації.

Додатково аналізувалися записи журналів роботи програми, що дало можливість перевірити коректність функціонування окремих модулів

системи, точність визначення активності користувача, роботу механізму AFK-детекції та правильність формування результатів моніторингу.

Результати перевірки виконання функціональних вимог наведено в таблиці 3.1.

Таблиця 3.1 – Перевірка виконання функціональних вимог

Функціональна вимога	Реалізований модуль	Спосіб перевірки	Результат
Отримання інформації про активне вікно	AppMonitorModule	Аналіз записів журналу та назв активних вікон	Виконано
Створення скріншотів екрана	ScreenModule	Перевірка формування скріншотів під час роботи програми	Виконано
Розпізнавання тексту за допомогою OCR	ClassifierModule	Аналіз результатів OCR у журналі роботи	Виконано
Визначення стану AFK	ClassifierModule, AIModule	Порівняння результатів класифікації з відеозаписом тестування	Виконано
Класифікація активності користувача	ClassifierModule	Порівняння визначених категорій з фактичною діяльністю користувача	Виконано
Збереження результатів моніторингу	DatabaseModule	Перевірка записів у базі даних SQLite	Виконано

Продовження таблиці 3.1

Функціональна вимога	Реалізований модуль	Спосіб перевірки	Результат
Формування логів роботи системи	LoggerModule	Аналіз створених журналів та їхнього вмісту	Виконано
Координація роботи модулів системи	Controller	Контроль запуску та взаємодії компонентів системи	Виконано
Формування статистики активності	DatabaseModule, LoggerModule	Аналіз накопичених даних моніторингу	Виконано

Проведене тестування підтвердило коректну реалізацію всіх функціональних вимог, визначених на етапі проєктування системи. У процесі перевірки не було виявлено випадків втрати даних, помилок запису журналів або збоїв під час взаємодії між окремими модулями застосунку.

Окрім функціональних вимог було виконано перевірку нефункціональних вимог, які стосуються продуктивності, безпеки та стабільності роботи системи. Результати перевірки наведено у таблиці 3.2.

Таблиця 3.2 – Перевірка нефункціональних вимог

Нефункціональна вимога	Спосіб перевірки	Результат
Навантаження на процесор не більше 5 %	Моніторинг використання CPU під час роботи застосунку	Виконано
Локальне зберігання даних користувача	Аналіз журналів роботи та структури системи зберігання даних	Виконано
Коректне формування журналів роботи	Перевірка створених log-файлів та їх вмісту	Виконано

Продовження таблиці 3.2

Нефункціональна вимога	Спосіб перевірки	Результат
Стабільна робота протягом тривалого часу	Безперервне тестування застосунку під час сеансу моніторингу	Виконано
Простий та зрозумілий інтерфейс користувача	Перевірка доступності основних функцій без додаткового налаштування	Виконано
Можливість аналізу результатів моніторингу	Перевірка журналів роботи та накопичених даних у базі	Виконано

Результати перевірки показали, що застосунок відповідає встановленим нефункціональним вимогам. Під час тестування середнє навантаження на центральний процесор становило 1,4%, що знаходиться в межах встановленого обмеження та не перевищувало 5 %.

Порівняння результатів виконувалося у декілька етапів:

− аналіз журналу роботи програми;

− виділення часових інтервалів окремих типів активності;

− порівняння результатів класифікації з фактичною діяльністю користувача, зафіксованою на відеозаписі.

3.4.1 Експеримент та результати з використанням вебкамери та підключенням до інтернету

Після проведення експерименту необхідно виконати обробку журналу роботи програми та розподілити записи за окремими часовими інтервалами активності. Як приклад використовується файл журналу «clean_log_16_06_2026_19_02.txt». Повний вміст файлу наведено у лістингу А.1 додатку А.

Цей журнал можна розділити на відрізки з однаковим типом активності:

− [19:02:45] WORK;

− [19:02:51] STUDY;

− [19:02:56 – 19:05:20] ENTERTAINMENT;

− [19:05:25 – 19:05:46] STUDY;

− [19:05:51] AFK;

− [19:05:56 – 19:06:27] STUDY;

− [19:06:32] AFK;

− [19:06:37] STUDY;

− [19:06:42 – 19:08:24] AFK;

− [19:08:29 – 19:09:30] ENTERTAINMENT;

− [19:09:35 – 19:10:36] STUDY;

− [19:10:41 – 19:11:47] AFK;

− [19:11:53 – 19:12:09] WORK;

− [19:12:14] ENTERTAINMENT;

− [19:12:19] WORK.

Отримавши час для кожного виду активності та розділивши на відрізки можемо одразу відокремити безпосередній час зміни активності.

На приклад якщо взяти відрізки «[19:02:56 – 19:05:20] ENTERTAINMENT» та «[19:05:25 – 19:05:46] STUDY» можемо побачити що програма виявила зміну активності о 19:05:25. Це і буде точкою зміни активності.

Оскільки програма робить знімок екрану кожні 5 секунд похибку можна взяти також 5 секунд. Отже якщо активність зміниться і програма відобразить це у журналі у наступні 5 секунд це буде вважатися точною оцінкою, якщо затримка зміни буде більше 5 то буде вважатися що програма некоректно класифікувала діяльність.

У таблицю 3.3 будуть занесені час фактичної зміни активності, час зміни активності який виявив застосунок та результат їх порівняння, тобто збіг часу та класифікації активності. Час зміни, що виявила програма взято безпосередньо з журналу, а фактичний час зміни активності розраховано у відповідності з вихідним відеофайлом «2026-06-16 19-02-34.mkv».

Таблиця 3.3 – Результати порівнянь класифікації змін

Час зміни, яку виявила програма	Фактичний час зміни активності	Збіг
[19:02:45] WORK	[19:02:45] WORK	Збігається
[19:02:51] STUDY	[19:02:50] STUDY	Збігається
[19:02:56] ENTERTAINMENT	[19:02:52] ENTERTAINMENT	Збігається
[19:05:25] STUDY	[19:05:22] STUDY	Збігається
[19:05:51] AFK	Зміни активності не було	Не збігається
[19:05:56] STUDY	[19:05:56] STUDY	Збігається
[19:06:32] AFK	Зміни активності не було	Не збігається
[19:06:37] STUDY	[19:06:37] STUDY	Збігається
[19:06:42] AFK	[19:06:40] AFK	Збігається
[19:08:29] ENTERTAINMENT	[19:08:29] ENTERTAINMENT	Збігається
[19:09:35] STUDY	[19:09:31] STUDY	Збігається
[19:10:41] AFK	[19:10:41] AFK	Збігається
[19:11:53] WORK	[19:11:51] WORK	Збігається
[19:12:14] ENTERTAINMENT	[19:12:12] ENTERTAINMENT	Збігається
[19:12:19] WORK	[19:12:15] WORK	Збігається

За результатами першого порівняння можна побачити що застосунок помилково виділив 2 зміни активності на «Відсітність активності», це пов’язано з тим, що програма враховує не лише зміст екрану, а ще й відео з вебкамери. Якщо алгоритм не може розпізнати людину на відео з вебкамери вважається що активності немає.

Для більш точного результату будуть переглянуті та порівняні усі події з журналу подій. Результати порівняння представлені у таблиці Б.1 додатку Б.

За результатами порівняння програма правильно виявила та класифікувала 13 з 15 змін активності. Правильними виявилися 108 з 110 записів у журналі, 8 з яких вважаються правильними з урахуванням похибки затримки між знімками екрану, це свідчить про дуже високу точність класифікації, близько 98,18%. Загальні результати експерементального тестування наведено у таблиці 3.4.

Таблиця 3.4 – Загальні результати експерементального тестування

Показник	Значення
Загальна кількість записів	110
Правильно класифікованих	108
Помилкових	2
Точність	98,18 %
Кількість змін активності	15
Правильно визначених змін	13

3.4.2 Експеримент та результати без використання вебкамери та без інтернет-з’єднання

Як і для попереднього порівняння нам потрібно обробити журнал, отриманий у ході тестового запуску програми. Як приклад використовується файл журналу «clean_log_18_06_2026_07_58_10.txt». Повний вміст файлу наведено у лістингу А.2 додатку А.

Цей журнал можна так само розділити на відрізки з по активності:

− [07:58:10 – 07:59:10] ENTERTAINMENT;

− [07:59:16 – 08:01:03] STUDY;

– [08:01:08 – 08:02:10] AFK;

– [08:02:16 – 08:02:39] ENTERTAINMENT.

Отримавши час для кожного виду активності та розділивши на відрізки можемо одразу відокремити безпосередній час зміни активності.

Умови збігу залишилися такими самими, тому що показник частоти знімків екрану під час обох тестувань не змінювався. За отриманими даними було заповнено таблицю 3.5.

Таблиця 3.5 – Результати порівнянь класифікації змін

Час зміни, яку виявила програма	Фактичний час зміни активності	Збіг
[07:58:10] ENTERTAINMENT	[07:58:10] ENTERTAINMENT	Збігається
[07:59:16] STUDY	[07:59:12] STUDY	Збігається
[08:01:08] AFK	[08:01:07] AFK	Збігається
[08:02:16] ENTERTAINMENT	[08:02:12] ENTERTAINMENT	Збігається

За результатами порівняння можна побачити що застосунок правильно виділив усі зміни активності та класифікував їх.

Для більш точного результату будуть переглянуті та порівняні усі події з журналу подій. Результати порівняння представлені у таблиці Б.2 додатку Б.

За результатами порівняння програма правильно виявила та класифікувала 4 з 4 змін активності. Правильними виявилися 48 з 48 записів у журналі, 3 з яких вважаються правильними з урахуванням похибки затримки між знімками екрану, це свідчить про дуже високу точність класифікації, близько 100%. Загальні результати експерементального тестування наведено у таблиці 3.6.

Таблиця 3.6 – Загальні результати експерементального тестування

Показник	Значення
Загальна кількість записів	48
Правильно класифікованих	48
Помилкових	0
Точність	100 %
Кількість змін активності	4
Правильно визначених змін	4

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та реалізовано кросплатформений інтелектуальний застосунок для автоматичного фонового відстеження дій користувача персонального комп’ютера з метою детального аналізу витраченого часу, оцінювання продуктивності та формування аналітичних звітів.

Для досягнення поставленої мети в роботі було вирішено такі завдання:

– проаналізовано існуючі методи, алгоритми та програмні рішення для відстеження активності користувача ПК, виявлено їх переваги, недоліки та функціональні обмеження;

– досліджено методи комп’ютерного зору (обробки зображень, виділення візуальних ознак) та технології оптичного розпізнавання тексту (OCR) для багаторівневого аналізу вмісту екрана;

– спроєктовано модульну архітектуру системи, що включає модулі системного моніторингу, створення скріншотів, аналізу візуальних характеристик, розпізнавання тексту, взаємодії з вебкамерою та локального збереження даних;

– розроблено алгоритм та евристичні правила для автоматичної класифікації діяльності користувача за основними категоріями (програмування, навчання, розваги, робота з документами тощо) на основі комплексу отриманих системних і текстових ознак;

– реалізовано механізм визначення стану неактивності користувача на основі швидкого хешування послідовних знімків екрана та аналізу його присутності в полі зору вебкамери;

– програмно реалізовано прототип (MVP) застосунку мовою Python із використанням сучасних бібліотек PySide6, OpenCV, NumPy, pytesseract та СКБД SQLite для безпечного локального зберігання інформації;

– розроблено модулі статистики та аналізу журналів (логів) із можливістю інтеграції елементів штучного інтелекту для формування персоналізованих рекомендацій користувачеві;

– проведено експериментальне тестування розробленого застосунку в різних режимах функціонування (зокрема, з використанням вебкамери/інтернету та в автономному режимі) й оцінено точність роботи модулів класифікації.

Проведений аналіз існуючих рішень для тайм-трекінгу (RescueTime, ManicTime, ActivityWatch) підтвердив, що більшість із них обмежуються лише аналізом назв процесів чи заголовків вікон, не враховуючи фактичного візуального та текстового наповнення екрана. Застосування в розробленому рішенні бібліотеки PyGetWindow дозволило стабільно отримувати системні заголовки вікон у режимі реального часу з мінімальним навантаженням на систему. Для усунення обмежень класичного підходу було успішно інтегровано технологію OCR на базі pytesseract, що надало можливість здійснювати контекстний пошук за ключовими словами безпосередньо зі скріншотів, суттєво підвищивши точність розпізнавання типу діяльності.

Під час реалізації модуля визначення стану AFK та аналізу динаміки змін на екрані ефективно використано бібліотеку NumPy для математичного обчислення абсолютної різниці між послідовними кадрами. Це, у поєднанні з оптимізованим алгоритмом хешування через модуль hashlib, дозволило оперативно фіксувати бездіяльність або зміну фокусу користувача без значних обчислювальних затрат.

Експериментальна перевірка розробленого прототипу підтвердила його високу ефективність та відповідність усім сформованим нефункціональним вимогам. Середнє навантаження на центральний процесор (CPU) під час активного фонового моніторингу становило всього 1,4%, що повністю задовольняє встановлений ліміт (не більше 5%) і робить роботу застосунку непомітною для користувача. Тестування в автономному режимі (без вебкамери та інтернет-з’єднання) продемонструвало безпомилкове виділення

та класифікацію всіх інтервалів активності користувача («STUDY», «WORK», «ENTERTAINMENT», «AFK»).

Результати кваліфікаційної роботи мають високу практичну цінність і можуть бути використані як інструмент персонального менеджменту часу та самоконтролю, у системах корпоративного аналізу ефективності використання робочого часу або в освітніх середовищах для моніторингу самостійної роботи студентів. У подальшому є сенс допрацювати систему, розширивши набір правил класифікації за допомогою інтеграції сучасних локальних моделей машинного навчання та додавши модуль розширеної візуалізації довготривалої статистики.

Результати роботи апробовано у вигляді тез доповідей під час XXX Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. RescueTime. rescuetime.com. URL: https://www.rescuetime.com (дата звернення 11.03.2026).

2. ManicTime. manictime.com. URL: https://www.manictime.com (дата звернення 14.03.2026).

3. ActivityWatch. activitywatch.net. URL: https://activitywatch.net (дата звернення 16.03.2026).

4. WakaTime. wakatime.com. URL: https://wakatime.com (дата звернення 17.03.2026).

5. Toggl track: time tracking software for any workflow. Toggl Track: Time Tracking Software for Any Workflow. URL: https://toggl.com (дата звернення 19.03.2026).

6. Гороховатський, В., & Творошенко, І. (2026) Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія. Харків: ХНУРЕ.

7. PyGetWindow. PyPI · The Python Package Index. URL: https://pypi.org/project/PyGetWindow (дата звернення 20.03.2026).

8. Hashlib. Python documentation. URL: https://docs.python.org/uk/3/library/hashlib.html (дата звернення 22.03.2026).

9. NumPy. NumPy. URL: https://numpy.org (дата звернення 28.03.2026).

10. Yakovleva, O., Matúšová, S., Tvoroshenko, I., & Isaiev, Y. (2024) Visitor counting based on video stream analysis from surveillance cameras, *Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing»*, 20(1), pp. 67–87.

11. Gorokhovatskyi, O., & Yakovleva, O. (2024) Medoids as a packing of ORB image descriptors, *Advanced Information Systems*, 8(2), pp. 5–11.

12. Opencv-python. PyPI · The Python Package Index. URL: https://pypi.org/project/opencv-python (дата звернення 03.04.2026).

13. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025) Image description compression in classification structural methods, *IEEE Access*, 13, pp. 43631–43641.

14. Gorokhovatskyi, V., Tvoroshenko, I., & Yakovleva, O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, 33(1), pp. 113–125.

15. Yakovleva, O., & Nikolaieva, K. (2020) Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors, *Advanced Information Systems*, 4(4), pp. 89–101.

16. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023) Face Detection for Video Surveillance-based Security System. *Computational Linguistics and Intelligent Systems 2023 (Volume III) (COLINS-2023). CEUR Workshop Proceedings*, 3403, pp. 69-86.

17. Yakovleva, O., Kovač, M., Ardasov, V., & Yeremenko, I. (2023) Study on adding functionality to the Zoom online conference system for monitoring the participant activities, *Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing»*, 19(1), pp. 161–183.

18. Cherednichenko, O., Yakovleva, O., & Hrechyshkin, D. (2025) A data-centric approach to crowd counting: Synthetic dataset creation and evaluation. In S. Vladov, Y. Bodyanskiy, V. Lytvyn, I. Aizenberg, & L. Chyrun (Eds.), *Proceedings of the International Workshop on Applied Intelligent Security Systems in Law Enforcement (AISSLE-2025)*, *CEUR Workshop Proceedings*, pp. 22–34.

19. Остапчук, В.О. (2026) Порівняльний аналіз OCR та LLM підходів для перекладу тексту з зображення сторінки манґи. *30-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»*. Збірник матеріалів форуму, 22–24 квітня 2026 року, Харків, Україна, Т. 7, С. 127-129.

20. Pytesseract. PyPI · The Python Package Index. URL: https://pypi.org/project/pytesseract (дата звернення 07.04.2026).

21. Ковтуненко, А. Р., Яковлева, О. В., Любченко, В. А., & Янголенко, О. В. (2020) Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників, *Вісник Національного технічного університету ХПІ*, 1(3), С. 71–87.

22. Yakovleva, O., Matúšová, S., Liubchenko, V., & Maksimov, H. (2025) Innovative solutions based on AI in education, on the example of generating multimodal lecture notes, *Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing»*, 21(1), pp. 140-163.

23. Yakovleva, O., Sokolova, A., & Datsok, Y. (2026) Optimization of multimodal Gemini models for food product composition recognition and explanation with automated translation into Ukrainian. *Theoretical and empirical scientific research: concept and trends: proceedings of the X international scientific and practical conference*, May 8, 2026, Oxford, United Kingdom, pp. 117-120.

24. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2026) Feature space quantization and application of metrics in structural methods of image recognition, *IEEE Access*, 14, pp. 17812–17824.

25. Yakovleva, O., Matúšová, S., & Talakh, V. (2025) Gradio and Hugging capabilities for developing research AI applications. *Proceedings of the VII International Scientific and Practical Conference «Scientific practice: modern and classical research methods»*, Boston, USA, pp. 202–205.

26. PySide6. PyPI · The Python Package Index. URL: https://pypi.org/project/PySide6 (дата звернення 11.04.2026).

27. Pytesseract. PyPI · The Python Package Index. URL: https://pypi.org/project/pytesseract (дата звернення 12.04.2026).

28. Tesseract OCR – the world's best open source OCR engine. Tesseract OCR. URL: https://tesseractocr.org (дата звернення 15.04.2026).

29. SQLite home page. SQLite Home Page. URL: https://sqlite.org (дата звернення 19.04.2026).

30. The python standard library. Python documentation. URL: https://docs.python.org/3/library/index.html (дата звернення 24.04.2026).

31. Панчук, В.О. (2026) Розробка модульної системи аналізу активності користувача персонального комп’ютера з використанням методів штучного інтелекту. *30-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»*. Збірник матеріалів форуму, 22–24 квітня 2026 року, Харків, Україна, Т. 7, С. 130–132.