Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення програмного забезпечення оцінювання виконання робочих задач для раціонального розподілу часу
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТАКІТ-21-1
Олександр ПРИСТЕНСЬКИЙ
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)
Керівник ст. викл. Сергій ТЕСЛЮК
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я , Пристенський Олександр Романович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«13» червня 2025 р.  Олександр ПРИСТЕНСЬКИЙ

Харківський національний університет радіоелектроніки

Факультет АКТ

Кафедра КІТАР

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«19» травня 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Пристенському Олександру Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного забезпечення оцінювання виконання робочих задач для раціонального розподілу часу
затверджена наказом університету від 19.05.2025 р. №390Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 26.06 2025 р.
3. Вихідні дані до роботи 3.1 Операційна система – Windows 11;
3.2 Об'єм оперативної пам'яті – 8 Гб;
3.3 Мова написання програмного забезпечення – C#;
3.4 Фреймворк – WPF;
3.5 База даних – SQLite;
3.6 Графіки і діаграми;
3.7 Інформаційні панелі із статистикою.
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Провести аналіз аналогічних програмних рішень з оцінки продуктивності;
4.2 Розробити архітектуру та алгоритм роботи програмного забезпечення;
4.3 Розробити модель машинного навчання для класифікації типів активності;
4.4 Створити базу даних (БД) для зберігання та аналізу отриманих даних;
4.5 Реалізувати механізм збору даних про активність користувача;
4.6 Розробити систему візуалізації результатів;
4.7 Тестування та перевірка ПЗ;
4.8 Перевірити систему на стійкість;
4.9 Охорона праці.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Демонстраційний матеріал у вигляді презентації PowerPoint (*.pptx) – 15 с.

6. Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	07.04 – 13.04	Виконано
2	Аналіз матеріальної бази	14.04 – 23.04	Виконано
3	Розробка структури ПЗ	24.04 – 10.05	Виконано
4	Реалізація програмного забезпечення	11.05 – 30.05	Виконано
5	Оформлення кваліфікаційної роботи	31.05 – 15.06	Виконано
6	Подання роботи на нормо контроль	16.06	Виконано
7	Подання роботи на перевірку на плагіат	17.06	Виконано
8	Подання роботи на рецензію	19.06	Виконано
9	Подання роботи на підпис зав. кафедри	23.06	Виконано
10	Подання роботи до ЕК	26.06	Виконано

Дата видачі завдання 19.05.2025 р.

Здобувач  Олександр ПРИСТЕНСЬКИЙ

(підпис)

Керівник роботи _____ ст. викл. Сергій ТЕСЛЮК

(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 62 с., 3 табл., 26 рис., 4 додатків., 23 джерел.

АВТОМАТИЗАЦІЯ, АКТИВНІСТЬ КОРИСТУВАЧА, АНАЛІЗ ДАНИХ, ВІЗУАЛІЗАЦІЯ, ЕФЕКТИВНІСТЬ, МОНІТОРИНГ, КЛАСИФІКАЦІЯ, ПРОДУКТИВНІСТЬ, МАШИННЕ НАВЧАННЯ, ТАЙМ-МЕНЕДЖМЕНТ, ML.NET, SQLITE, WINDOWS API

Об’єкт розробки – процес виконання робочих завдань працівниками у цифровому середовищі.

Предмет розробки – програмне забезпечення для автоматизованого моніторингу активності та оцінки продуктивності персоналу.

Мета розробки – автоматизація процесу оцінки виконання завдань працівниками для виявлення неефективних витрат часу за рахунок розробки програмного засобу, використовуючи модель машинного навчання класифікації даних активності.

Методи, що використовуються для розробки програмного забезпечення оцінювання виконання робочих задач для раціонального розподілу часу, це аналіз літературних джерел та існуючих систем моніторингу, програмна реалізація та тестування програмного забезпечення. Використання C# та платформи .NET для розробки, SQLite для бази даних.

Практичне значення розробки полягає у створенні програмного прототипу, який дозволяє автоматизовано відстежувати активність користувачів та оцінювати ефективність виконання завдань. Розроблена система має два застосування: індивідуальні користувачі можуть використовувати її для самооцінки, а організації – для підвищення прозорості виконання завдань без шкоди для конфіденційності.

ABSTRACT

Explanatory note: 62 pp., 3 tables, 26 figures, 4 appendices, 23 sources.

AUTOMATION, CLASSIFICATION, DATA ANALYSIS, EFFICIENCY, MACHINE LEARNING, ML.NET, MONITORING, PRODUCTIVITY, SQLITE, TIME MANAGEMENT, USER ACTIVITY, VISUALIZATION, WINDOWS API

The object of development is the process of performing work tasks by employees in a digital environment.

The subject of development is software for automated monitoring of activity and assessment of staff productivity.

The goal of the development is to automate the process of evaluating employee performance on tasks in order to identify inefficiencies in time usage. This will be achieved by developing software that uses a machine learning model to classify activity data.

The research methods used to develop software for evaluating the performance of work tasks for the rational allocation of time are the analysis of literary sources and existing monitoring systems, software implementation, and software testing. C# and the .NET platform are used for development, and SQLite is used for the database.

The practical significance of the development lies in the creation of a software prototype that allows automated tracking of user activity and evaluation of task performance efficiency. The developed system has two applications: individual users can use it for self-assessment, and organizations can use it to increase the transparency of task performance without compromising confidentiality.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних;

API – Application Programing Interface;

CRUD – Create, Read, Update, Delete;

CSV – Comma-Separated Values;

KPI – Key Performance Indicator;

ML – Machine Learning;

MVVM – Model View ViewModel;

SCADA – Supervisory Control and Data Acquisition;

SQL – Structured Query Language;

UI – User Interface;

WinAPI – Windows API;

WPF – Windows Presentation Foundation.

ВСТУП

На сучасному етапі розвитку цифрових технологій питання раціонального використання робочого часу працівників набуває особливої актуальності. Збільшення обсягів інформаційної роботи та високі вимоги до ефективності виробничих процесів зумовлюють необхідність впровадження автоматизованих інструментів для моніторингу та аналізу виконання робочих завдань. У цьому контексті особливу цінність становлять програмні рішення, здатні не лише фіксувати активність користувача, а й надавати керівництву об'єктивні показники продуктивності персоналу.

Застосування таких систем дає змогу підвищити якість управління трудовими ресурсами, виявляти нераціональні витрати часу, оптимізувати виконання завдань і, в кінцевому підсумку, сприяти покращенню організації праці. Автоматизація процесів обліку й оцінки ефективності дозволяє підприємствам більш точно аналізувати ситуацію всередині колективу, знижувати витрати та підвищувати загальну результативність діяльності.

Паралельно із цим зростає потреба у створенні сучасного програмного забезпечення (ПЗ), що здатне автоматично здійснювати оцінку виконання робочих задач. Таке ПЗ повинно забезпечувати надійний і об'єктивний збір даних, мати гнучкий механізм аналізу та забезпечувати наочну візуалізацію результатів для прийняття обґрунтованих управлінських рішень.

Об’єкт розробки – процес виконання робочих завдань працівниками у цифровому середовищі.

Предмет розробки – програмне забезпечення для автоматизованого моніторингу активності та оцінки продуктивності персоналу.

Мета розробки – автоматизація процесу оцінки виконання завдань працівниками для виявлення неефективних витрат часу за рахунок розробки програмного засобу, використовуючи модель машинного навчання класифікації даних активності.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести аналіз аналогічних програмних рішень з оцінки продуктивності, виявити їх переваги та недоліки;

– розробити архітектуру та алгоритм роботи програмного забезпечення;

– розробити модель ML (машинного навчання) класифікації типів активності;

– створити базу даних (БД) для зберігання та аналізу отриманих даних;

– реалізувати механізм збору даних про активність користувача;

– розробити систему візуалізації результатів;

– перевірити систему на стійкість;

– охорона праці;

– оформити кваліфікаційну роботу відповідно до вимог ДСТУ 3008:2015 [1] та методичних вказівок до написання випускних робіт здобувачів вищої освіти [2].

1 АНАЛІЗ ЗАСОБІВ АВТОМАТИЗОВАНОГО ОЦІНЮВАННЯ ВИКОНАННЯ ЗАДАЧ ДЛЯ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ПРАЦІ

1.1 Аналіз аналогічного програмного забезпечення

Метою проведення аналізу програмних засобів, призначених для оцінювання ефективності виконання працівниками своїх обов'язків, є всебічне дослідження їх функціональних характеристик, принципів збору, обробки та подання інформації, а також способів інтерпретації результатів. Особлива увага приділяється виявленню переваг і недоліків таких систем з огляду на їх практичну застосовність в умовах сучасного підприємства.

Результати аналізу використаємо для обґрунтування доцільності створення власного програмного рішення, яке відповідатиме актуальним вимогам виробничих процесів, а також дозволить уникнути повторення вже реалізованих підходів. Докладний аналіз забезпечує підвищення ефективності розробки та сприяє створенню конкурентоспроможного програмного продукту.

Одним із найбільш розповсюджених програмних рішень у сфері оцінювання продуктивності працівників є HUBSTAFF – система моніторингу робочої активності, орієнтована переважно на віддалені та змішані команди. Дане програмне забезпечення дозволяє в автоматичному режимі фіксувати дії користувача за комп'ютером, включаючи відкриті програми, активні вікна, веб-сайти, натискання клавіш та рух миші. Також підтримується ведення обліку часу за проектами й завданнями.

На рисунку 1.1 наведена інформаційна панель продуктивності працівника в програмному забезпеченні HUBSTAFF, а в таблиці 1.1 наведені переваги та недоліки ПЗ.

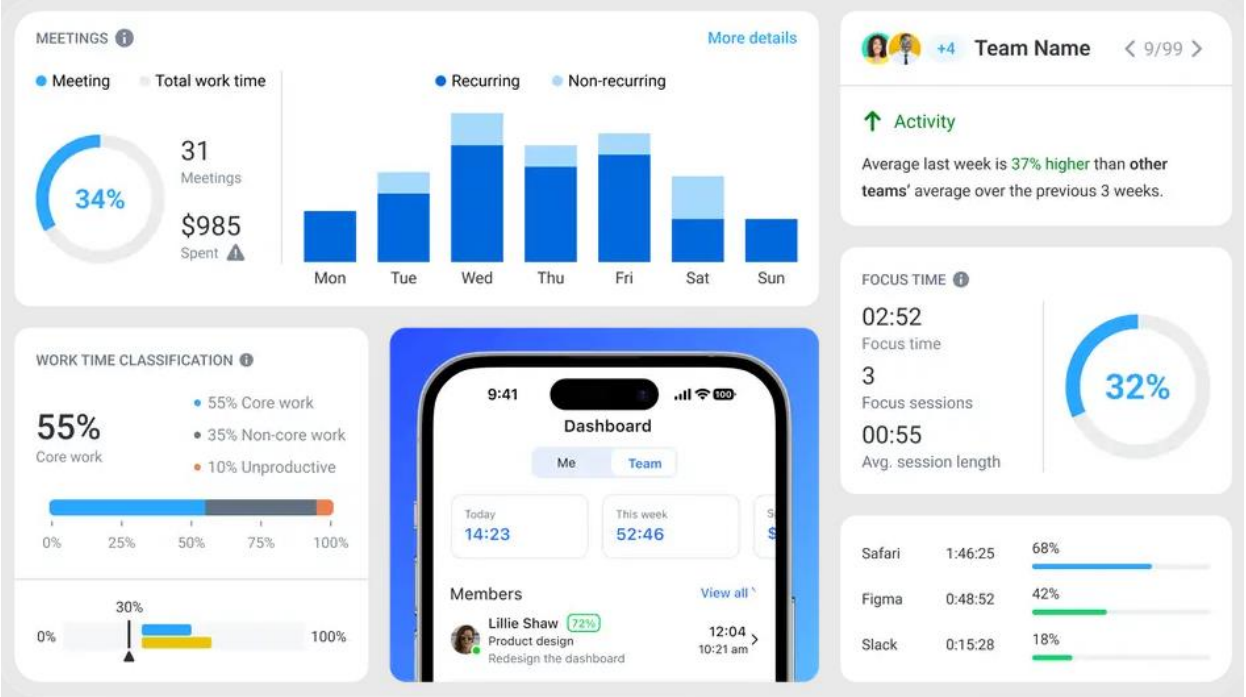


Рисунок 1.1 – Інформаційна панель продуктивності працівника в програмному забезпеченні Hubstaff [3]

Таблиця 1.1 – Переваги та недоліки програмного забезпечення Hubstaff

Характеристика	Опис
Переваги	
Облік часу	Автоматизований трекінг часу: запуск таймера, ручне додавання часу, формування звітів по завданнях (рис. 1.2).
Відстеження активності	Аналіз дій на клавіатурі та миші, фіксація відкритих вікон і сайтів.
Скріншоти	Автоматичне створення скріншотів через задані інтервали часу.
Аналітика і звіти	Формування звітів по часу, продуктивності, активності, витратах (рис. 1.2).
Кросплатформність	Підтримка Windows, macOS, Linux, Android, iOS (рис. 1.2).

Продовження таблиці 2.1.

Характеристика	Опис
Недоліки	
Вартість	Повний функціонал доступний лише в платній версії.
Гнучкість звітів	Обмежена адаптація звітів під потреби компанії без використання API.
Відсутність AI-аналітики	Немає вбудованих інструментів машинного навчання для аналізу чи прогнозування продуктивності.

HUBSTAFF є потужним інструментом для обліку часу та моніторингу активності, який забезпечує прозорість виконання завдань у віддаленій команді. Однак для його використання необхідне зважене ставлення до конфіденційності даних та фінансових витрат.

На рисунку 1.2 зображено реалізацію таймера, звітності та іншого функціоналу ПЗ Hubstaff.

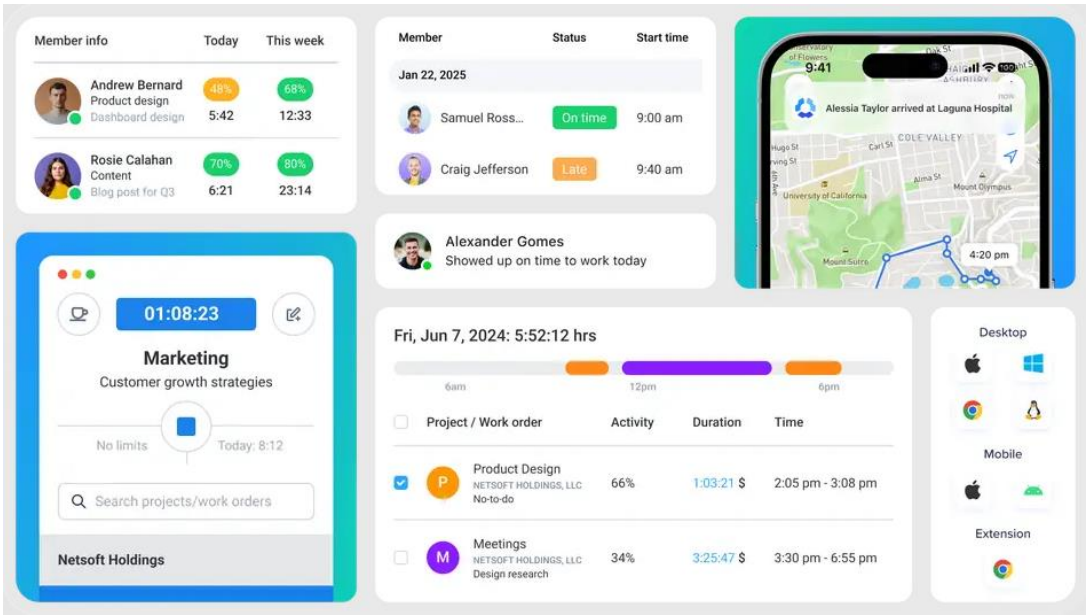


Рисунок 1.2 – Інформаційна панель продуктивності працівника в програмному забезпеченні Hubstaff [3]

Дане рішення не повністю відповідає вимогам системи, яка передбачає глибокий аналіз цілей і результатів виконання завдань, тому може бути використане лише частково або як приклад реалізації окремих модулів, таких як відстеження активності та звітність.

Ще одним поширеним додатком для оцінювання раціонального розподілу часу є RescueTime.

RescueTime – це один з найвідоміших програмних продуктів для відстеження особистої продуктивності. Даний додаток використовується як індивідуальними користувачами, так і командами.

Система надає можливість глибокого аналізу того, як саме розподіляється робочий час користувача за допомогою автоматичного збору статистики про використання додатків, веб-сайтів та часу не активності.

Вся активність класифікується за типами: робоча, розважальна, соціальні мережі тощо; та рівнем продуктивності – від «дуже продуктивної» до «дуже непродуктивної» (рис. 1.3). RescueTime має власну базу даних для автоматичної класифікації, а також дозволяє ручну корекцію.

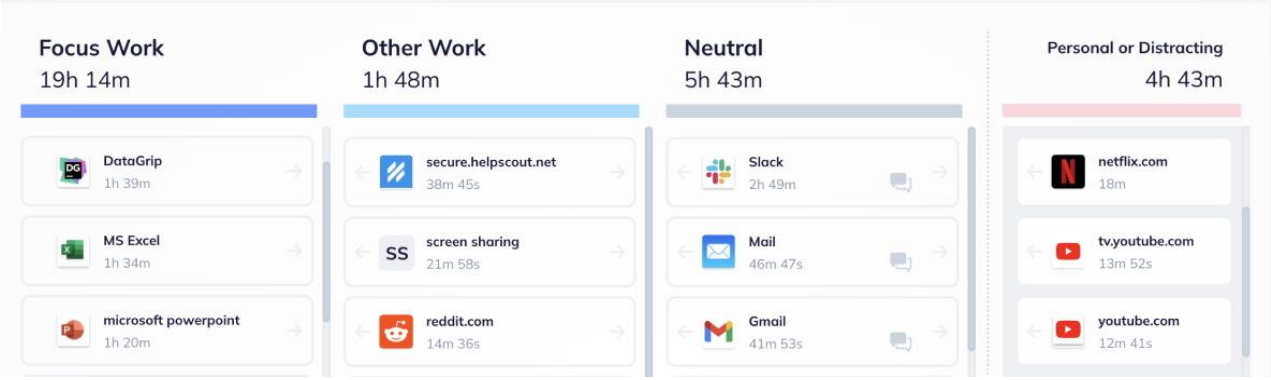


Рисунок 1.3 – Класифікація активності користувача за типами роботи у програмі RescueTime [4]

RescueTime надає користувачеві детальні звіти у вигляді графіків активності, які поділяються на щоденні, тижневі та місячні (рис. 1.4). Як зображено на рисунку 1.3, у додатку час розподіляється по категоріях, а також визначається найбільш продуктивні години.

Рисунок 1.4 – Тижнева візуалізація активності користувача в RescueTime [4]

У таблиці 1.2 наведено переваги та недоліки ПЗ RescueTime.

Таблиця 1.2 – Переваги та недоліки програмного забезпечення RescueTime

Характеристика	Опис
Переваги	
Автоматизація	Додаток не потребує ручного запуску або введення даних – усі дії користувача відслідковуються автоматично.
Деталізовані звіти	RescueTime генерує аналітичні звіти з візуалізацією, що дозволяє краще розуміти розподіл робочого часу.
Персоналізація	Користувач може налаштовувати рівні продуктивності, категорії додатків та сайтів відповідно до власних потреб.
Інтуїтивний інтерфейс	Простий, зручний у використанні, не потребує технічної підготовки.

Продовження таблиці 1.2.

Характеристика	Опис
Приватність	Користувач самостійно обирає, які програми й активності мають бути відслідковані.
Недоліки	
Відсутність контролю завдань	Додаток не оцінює рівень виконання поставлених задач, лише фіксує активність користувача.
Обмежена функціональність у безкоштовній версії	Командні функції та розширені звіти доступні лише при платній підписці.
Можлива некоректна класифікація	Автоматичне визначення продуктивності не завжди враховує контекст виконуваних завдань.
Закритий код	Вихідний код RescueTime недоступний, що унеможливлює його адаптацію або розширення для специфічних потреб.

Таким чином, RescueTime є ефективним інструментом для самоаналізу, проте у завданнях комплексного оцінювання ефективності працівників на підприємстві доцільно доповнювати його іншими рішеннями, або ж створювати власне програмне забезпечення, що поєднує моніторинг активності із контролем досягнення цілей. Такий підхід дозволяє сформувати більш об'єктивну і багатовимірну картину продуктивності.

У результаті аналізу аналогів програмних рішень, таких як Hubstaff та RescueTime, виявлено переваги та недоліки, що не повною мірою задовольняють цілі створюваного ПЗ. Проаналізовано елементи та функції, які доцільно використовувати в рішенні та чим можемо доповнити для вирішення проблеми оцінювання виконання робочих задач.

1.2 Аналіз методів оцінювання ефективності працівників

В умовах активної цифровізації та автоматизації управлінських процесів питання об’єктивного оцінювання ефективності діяльності працівників набуває все більшого значення. Керівники підприємств прагнуть не лише бачити кінцевий результат роботи, але й розуміти, як саме працівники використовують свій робочий час, з якою продуктивністю виконуються задачі, і де саме можуть виникати втрати ефективності.

У контексті об’єктивного оцінювання актуальним стає аналіз існуючих методів оцінювання ефективності персоналу – як класичних, так і сучасних. Особливу увагу варто приділити тим із них, які можуть бути реалізовані в рамках розробки програмного забезпечення для автоматизованого моніторингу активності.

До класичних методів оцінювання відносяться натуральний, трудовий та вартісний методи.

Сутність натурального методу полягає в тому, що обсяг виробничої продукції і продуктивність праці розраховуються в натуральних одиницях (штуках, кілограмах, метрах тощо) [5]. Такий підхід традиційно використовується в галузях, де продукція має чітко вимірюваний обсяг.

Метод є неактуальним для оцінювання інтелектуальної праці, зокрема в офісному середовищі. У рамках даної кваліфікаційної роботи, де йдеться про аналіз цифрової активності працівників, застосування натурального методу є недоцільним через відсутність фізично вимірюваних результатів.

За трудовим методом оцінка базується на обліку витрат робочого часу на виконання конкретних завдань. Це дозволяє співвідносити витрачений час із обсягом виконаної роботи.

Цей метод частково відповідає концепції створюваного програмного забезпечення, оскільки система передбачає фіксацію часу активності працівника в різних додатках чи веб-ресурсах. Трудовий метод може слугувати базовою платформою для збору первинних даних.

У сучасних умовах найпоширенішим методом вимірювання продуктивності праці є вартісний (грошовий), який ґрунтується на використанні вартісних показників обсягу продукції (валова, товарна продукція, валовий оборот, нормативна вартість обробки, чиста, нормативно-чиста й умовно-чиста продукція, валовий дохід) [5].

Попри свою універсальність, метод не може бути реалізованим у межах розроблюваного ПЗ, оскільки його функціонал не включає доступ до фінансових показників компанії. Таким чином, цей метод технічно непридатний для впровадження в межах цієї роботи.

До сучасних методів оцінювання ефективності працівників належить метод ключових показників ефективності (KPI).

Цей метод ґрунтується на встановленні чітких, вимірюваних цілей, досягнення яких дозволяє об’єктивно оцінити результативність роботи конкретного співробітника або команди.

Метод KPI добре поєднується з функціональністю розроблюваного програмного забезпечення, оскільки дозволяє оцінити не лише факт активності, але й якість виконання завдань. Саме тому він може стати одним із ключових компонентів у системі моніторингу.

Наступним сучасним методом оцінки ефективності працівника є метод «360 градусів». Метод передбачає зворотний зв’язок від різних учасників робочого процесу – колег, керівництва, підлеглих і самого працівника. Оцінка формується на основі комплексного, хоча й суб’єктивного, сприйняття діяльності співробітника.

Попри популярність у корпоративному середовищі, цей підхід є малопридатним для реалізації у вигляді автоматизованого інструменту. Його впровадження потребує додаткових ресурсів, анкетування, обробки соціально-психологічних даних, що виходить за межі завдань даної роботи.

Аналіз існуючих підходів показав, що найбільш релевантними для реалізації в межах даного ПЗ є трудовий метод у поєднанні з методом KPI. Така комбінація дозволяє отримати комплексне уявлення про ефективність

працівника. Саме це поєднання стане теоретичною основою для побудови ефективної, об'єктивної системи оцінювання в умовах цифрового робочого середовища.

1.3 Аналіз вибраних інструментів для реалізації ПЗ

Для реалізації програмного забезпечення, яке має здійснювати автоматизоване відстеження робочої активності працівників і на основі зібраних даних формувати аналітику продуктивності, доцільно застосовувати інструменти, що поєднують продуктивність, масштабованість і зручність у використанні.

Для реалізації програмного забезпечення в межах даного проекту обрано середовище розробки Visual Studio, розроблене компанією Microsoft. Таке рішення має технічні переваги і відповідає завданням розробки.

Visual Studio є повноцінним інтегрованим середовищем розробки (IDE), забезпечує високу швидкість створення, тестування та налагодження програм. Середовище підтримує різноманітні інструменти та розширення. IDE дозволяє вільно керувати структурою проекту, легко реалізовувати інтерфейс користувача та інтегрувати сторонні бібліотеки, такі як ML.NET або бібліотеки для візуалізації.

Основою для будь-якої програмної реалізації виступає мова програмування. Для даного проекту вибрано сучасну, об’єктно-орієнтовану мову C#, яка належить компанії Microsoft і є частиною платформи .NET [6]. Мова добре зарекомендувала себе для створення застосунків під операційну систему Windows і забезпечує доступ до широкого набору API (Application Programming Interface) операційної системи, зокрема до бібліотеки WinAPI [7]. Дана бібліотека дозволяє отримувати відомості про активне вікно, фіксувати події введення з клавіатури і миші, а також здійснювати базове відстеження взаємодії користувача з системою.

Для побудови користувацького інтерфейсу обрано Windows Presentation Foundation (WPF). Потужна платформа дозволяє реалізовувати гнучкий інтерфейс з підтримкою прив'язки даних та шаблонів. Використання WPF особливо корисне у контексті роботи з великим обсягом запланованої інформації, яку будемо візуалізувати у вигляді графіків, діаграмам та таблиць активності.

Аналіз отриманих під час моніторингу даних є ключовим етапом в роботі ПЗ. Аналіз здійснюється за допомогою класичних алгоритмів, математичних розрахунків і методів машинного навчання. У межах проекту обрано бібліотеку машинного навчання ML.NET від Microsoft. Бібліотека належить до екосистеми .NET, що сприяє легкій інтеграції машинного навчання з іншими компонентами проекту без потреби у сторонніх мовах програмування.

ML.NET сприяє реалізації широкого спектру задач – від класифікації до побудови систем рекомендацій [8]. Бібліотек надає можливість використовувати вже готові навчені моделі та створювати власні під конкретні задачі. ML.NET дозволяє будувати адаптивні моделі, що поступово вдосконалюються на основі накопичених даних конкретного користувача.

Інтеграція машинного навчання в систему оцінювання не лише збільшує гнучкість аналізу, але й робить результати більш обґрунтованими. Процес аналізу переходить від простого моніторингу до глибинного розуміння особливостей робочого процесу.

У ПЗ оцінювання виконання задач для зберігання даних про активність користувача вибрано локальну базу даних SQlite. SQLite – це легка вбудована реляційна система управління базами даних (СУБД) [9]. БД є автономною, компактною і не потребує особистого серверу.

SQLite виступає як локальне сховище, яке зберігає часові мітки, назви вікон активності, прив'язку до задач, а також результати класифікації. Всі дані зберігаються у вигляді одного локального файлу, доступні миттєво, без додаткових мережевих затримок або потреби в адмініструванні сервера, що робить SQlite ідеальним вибором для настільних застосунків.

Оптимальним вибором для візуалізації результатів аналізу є графічна бібліотеку LiveCharts. Бібліотека є зручним інструментом для створення лінійних, стовпчикових та кругових діаграм, інтегрується в середовищі WPF. Графіки мають привабливий зовнішній вигляд та можливість анімації [10]. LiveCharts дає змогу оновлювати дані на графіках у режимі реального часу.

Функціоналу бібліотеки LiveCharts достатньо для потреб даного проекту, щоб забезпечити просту і наочну подачу звітів та результатів аналізу.

Проаналізували та обґрунтували вибір технологій і інструментів розробки ПЗ. Обрані інструменти відповідають функціональним вимогам та утворюють комплексну стабільну екосистему з можливістю майбутньої масштабованості. Усі компоненти добре поєднуються між собою, що дозволяє реалізувати повноцінну систему оцінювання ефективності працівника.

1.4 Висновки до першого розділу

У першому розділі кваліфікаційної роботи проаналізовано аналогічні програмні засоби, які вирішують задачу відстеження та оцінки активності користувача. Виявлено переваги та недоліки таких програмних рішень, як Hubstaff та RescueTime, що не повною мірою задовольняють цілі створюваного ПЗ. Проаналізували елементи та функції, які доцільно використовувати в рішенні та чим можемо доповнити для вирішення проблеми оцінювання виконання робочих задач.

У результаті аналізу існуючих методів оцінки продуктивності співробітників виявлено, що найбільш релевантними для реалізації в межах даного ПЗ є трудовий метод у поєднанні з методом KPI. Таке поєднання дозволяє отримати комплексне уявлення про ефективність працівника, що стане теоретичною основою для побудови ефективної, об'єктивної системи оцінювання в умовах цифрового робочого середовища.

Проаналізовано та обґрунтовано вибір технологій і інструментів розробки ПЗ. Обрані інструменти відповідають функціональним вимогам та утворюють комплексну стабільну екосистему з можливістю майбутньої масштабованості. Усі компоненти добре поєднуються між собою, що дозволяє реалізувати повноцінну систему оцінювання ефективності працівника.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ВИКОНАННЯ РОБОЧИХ ЗАДАЧ НА ПІДПРИЄМСТВІ

2.1 Розробка алгоритму роботи

Розробка алгоритму роботи програмного забезпечення є критично важливим етапом, який визначає логіку функціонування всієї системи. Алгоритм – це не просто послідовність дій, а структурована модель, що дозволяє забезпечити узгодженість між компонентами, стабільність обробки даних і точність у формуванні звітів.

Чіткий алгоритм сприяє не тільки зручності для розробника, а й забезпечує передбачуваністю й якістю всього продукту. Алгоритм надає можливість на ранніх етапах виявити логічні помилки, оптимізувати процеси, забезпечити масштабованість і простоту.

На початку відбувається ініціалізація компонентів та бібліотек ПЗ, а саме перевіряється стан бази даних, підключаються необхідні бібліотеки та активуються механізми збору інформації. Даний процес створює базову інфраструктуру, на якій надалі працюватиме система.

На етапі визначення задач, користувач самостійно вводить перелік задач, над якими планує працювати. В програмному забезпеченні передбачене окреме вікно введення, що дозволяє чітко сформулювати кожне завдання, задати його назву, тривалість, пріоритет або інші параметри. Такий підхід забезпечує повну контрольованість процесу з боку користувача та виключає будь-які неоднозначності, пов’язані з автоматичним визначенням задач.

У перспективі можливе розширення функціональності, яке передбачатиме централізоване призначення задач керівником або адміністратором.

Система постійно відстежує, чи є користувач активним. Активність визначається за подіями вводу та виводу (клавіатура, миша). Якщо активність

відсутня, фіксується неактивний. У разі активності збирається інформація про тип і характер дій, а саме які програми використовуються, як довго триває взаємодія, чи призначена вона для виконання задач. Усі ці дані зберігаються в локальній базі даних SQLite.

Оцінювання виконання задач є ключовим етапом, який дозволяє визначити, наскільки заплановані дії були реалізовані користувачем протягом робочої сесії. Оскільки програмне забезпечення не має прямого доступу до змісту виконаних задач і не може повністю автоматично перевірити результат, вирішальну роль у цьому процесі відіграє сам користувач.

Під час сесії або після її завершення користувачеві пропонується надати оцінку виконання кожного із завдань. Прикладом такої оцінки може бути вибір одного із трьох варіантів: не виконано, виконано частково або виконано повністю.

У майбутньому система може бути доповнена можливістю перевірки результатів в автоматичному режимі, проте наразі ручна самооцінка залишається найбільш доцільним і практичним рішенням.

Для візуалізації алгоритму роботи програмного забезпечення використано онлайн-редактор блок-схем draw.io, який дозволяє створювати структуровані діаграми з логічними зв’язками між елементами та розроблено алгоритм роботи, який зображено на рисунку 2.1.

Після завершення збору даних та оцінки виконання задач, система переходить до глибшого аналізу. Тут можуть застосовуватись класичні статистичні методи, таких як розрахунок середніх значень, варіацій, частот, а також алгоритми машинного навчання. Метою даного аналізу є не тільки фіксація активності, а класифікація її за родом діяльності (робоча чи розважальна), виявити закономірності поведінки та слабкі місця в розподілі часу, а також сформувати обґрунтовані рекомендації щодо підвищення продуктивності.

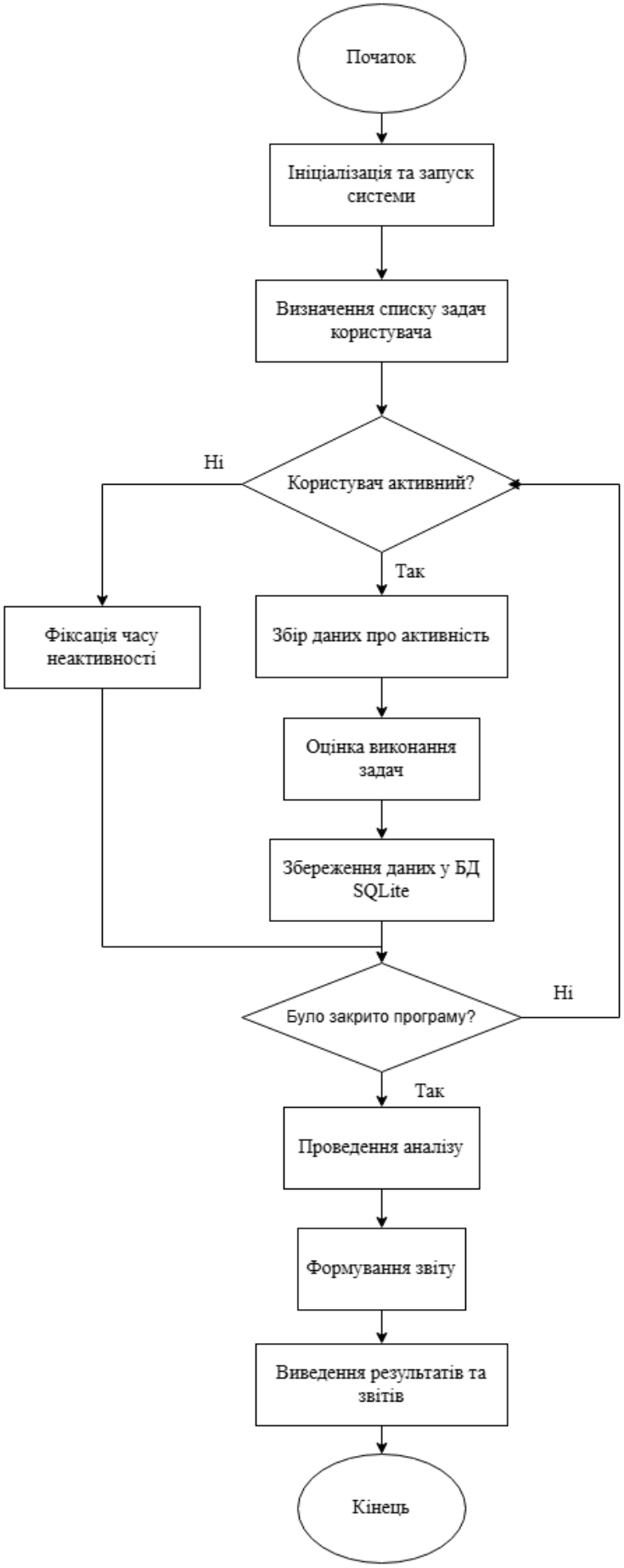


Рисунок 2.1 – Алгоритм роботи програмного забезпечення

Останнім етапом є формування звіту, який містить графіки активності, співвідношення ефективного та неефективного часу, а також рекомендації, щодо покращення продуктивності. Виведення результатів та звітів буде здійснюватися за допомогою WPF, а також передбачимо можливість експорту до файлу.

Розроблений алгоритм забезпечує чітко структурований процес взаємодії з користувачем – від введення задач до отримання звітів. Алгоритм узгоджує логіку між компонентами та послідовність етапів для уникнення суперечностей. Кожен етап має своє місце й функцію, що дозволяє системі працювати стабільно, передбачувано й точно відображати реальний процес виконання робочих задач. Також передбачили можливість масштабованості для майбутнього розширення ПЗ та додання нових функцій.

2.2 Розробка структури БД

Розробка структури бази даних – це важливий етап на шляху до написання коду програмного забезпечення. Задача даного етапу є розібратись, які дані будуть зберігатись в БД. ПЗ оцінювання виконання робочих задач для раціонального розподілу часу передбачає збір певної інформації. Далі роздивимось, які саме дані будуть збиратись, зберігатись та аналізуватись.

На етапі визначення задач, користувач вручну буде вводити задачі та їх пріоритетність на день або на інший проміжок часу. Автоматично буде додано дату створення задачі. Для збереження даних створена таблиця Tasks.

Система постійно буде фіксувати періоди активності та не активності користувача. Для цього в таблицю ActivitySessions буде автоматично занесена інформація початку, кінця та тривалості сесії. Розмежування між двома станами сесії відбувається у полі «IsActive», де при значені 0 – користувач не взаємодіяв з комп’ютером, а при значенні 1 – була зафіксована активність.

Продовженням таблиці ActivitySessions є детальніша таблиця ActivityLogs, які пов’язані полем «SessionId». В таблицю буде записано назви активного вікна та програми, час події, тривалість, а також класифікація продуктивності за

допомогою попередньо навченої моделі машинного навчання, записаної у поле «ActivityType». За результатом класифікації буде визначено продуктивна, нейтральна або непродуктивна активність (значення «2», «1» або «0» відповідно).

На етапі оцінки виконання задач користувач зазначає статус, який має три варіанти: не виконано, виконано частково або виконано повністю. Можемо передбачити бажання користувача написати коментар до виконаної задачі. Автоматично записуємо час оцінки задачі. Ці всі дані заносимо у таблицю TaskEvaluations.

Останньою таблицею денної аналітики є таблиця AggregatedStats, яка необхідна для побудування графіків. Доцільно роздивитись кожне поле цієї таблиці. Першим полем є дата запису. Наступними двома полями є сума тривалості всіх активних та не активних подій, яка розраховуються з таблиці ActivitySessions. Зі всіх активних подій розраховуємо у поле «EffectiveWork» тривалість подій, де «ActivityType» дорівнює 2. Також система підрахує та запише кількість виконаних цього дня задач у поле «TaskCount». Співвідношення ефективного часу до активного відображає ступінь продуктивності пройденого дня. Дане співвідношення запишемо у поле таблиці «ProductivityScore». Дані в цю таблицю вносяться автоматично в результаті аналізу інших таблиць.

За допомогою онлайн-редактора реалізовано ER-діаграми для візуалізації структури бази даних (рис. 2.2).

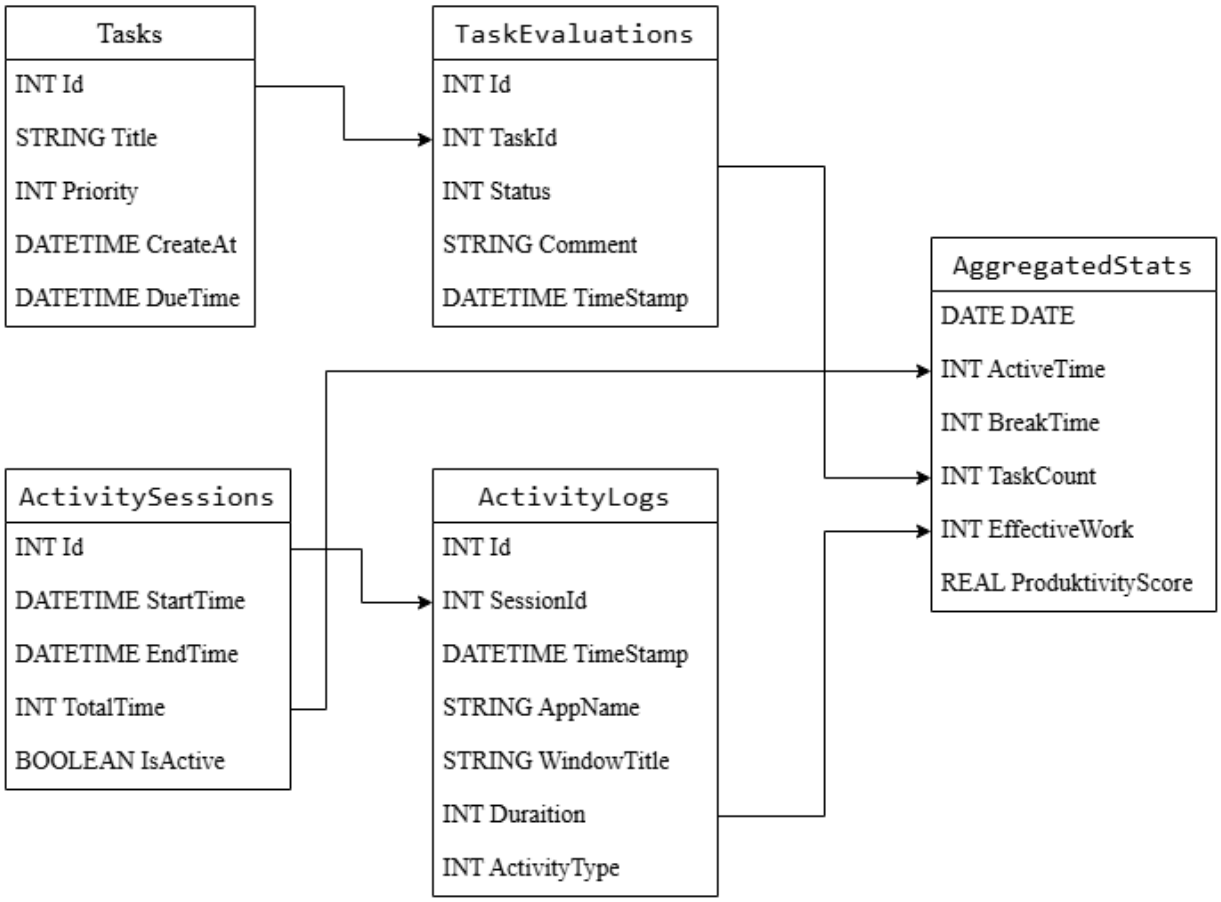


Рисунок 2.2 – Структура БД для ПЗ

Розроблено структуру бази даних, яка відповідає архітектурі та вимогам ПЗ для зберігання інформації, подальшій класифікації та обробки даних. Запропонована структура БД, яка містить 5 таблиць та відповідні поля для збереження необхідного обсягу даних. Визначили ключові залежності між таблицями. Визначили ключові залежності між таблицями. Спроектована БД дозволяє формувати точну аналітику за певний проміжок часу.

2.3 Розробка моделі класифікації цифрової активності

Реалізація програмного забезпечення передбачає класифікацію активності користувача з метою визначення ступеня продуктивності. Аналогічні розглянуті додатки використовують готовий набір вже відфільтрованих даних. Даний метод має найбільшу вірогідність, що дані будуть правильно класифікованими. Не зважаючи на це ручна фільтрація потребує людського втручання, не забезпечує гнучкості та масштабованості, особливо в умовах динамічного цифрового середовища. Виходячи з цього вибрано інший підхід, а саме машинне навчання.

На основі назви процесу та заголовка вікна модель машинного навчання відносить активність до однієї з трьох категорій: продуктивна (середа розробки, офісні програми), нейтральні (системні вікна, навігація по папкам) та непродуктивна (соціальні мережі, розважальні ресурси).

У межах проекту обрано бібліотеку машинного навчання ML.NET від Microsoft. ML.NET дає змогу навчати, створювати та постачати користувацькі моделі машинного навчання з використанням C# або F# для різних сценаріїв машинного навчання. ML.NET включає в себе такі функції, як автоматичне машинне навчання (AutoML) [8].

Перед тим як будувати модель класифікації сформовано навчальну вибірку з реальних записів активності, які занесемо в тимчасову таблицю БД TrainingData. Кожен запис включає текстові поля: ім’я запущеного процесу, заголовок активного вікна та час запису (рис. 2.3).

Id	TimeStamp	ProcessName	WindowsTitle
1	29.04.2025 17:22	TrainingTrackingActiv.exe	MainWindow
2	29.04.2025 17:22	explorer.exe	Program Manager
5	29.04.2025 17:22	DB Browser for SQLite.exe	DB Browser for SQLite
6	29.04.2025 17:23	DB Browser for SQLite.exe	Вибрати файл бази даних
7	29.04.2025 17:23	DB Browser for SQLite.exe	DB Browser for SQLite – D:\Бакалавр ...
8	29.04.2025 17:23	chrome.exe	диплом саша – Робота над дипломом – Google Chrome
9	29.04.2025 17:23	chrome.exe	ADO.NET і C#
10	29.04.2025 17:23	chrome.exe	Клас StackOverflowException (система)
18	29.04.2025 17:23	chrome.exe	Нова вкладка – Google Chrome
23	29.04.2025 17:24	SearchHost.exe	Пошук
24	29.04.2025 17:24	StartMenuExperienceHost.exe	Пуск
26	29.04.2025 17:24	explorer.exe	Вікно переповнення області завдань
27	29.04.2025 17:24	Spotify.exe	Spotify
28	29.04.2025 17:24	Spotify.exe	Spotify Premium
29	29.04.2025 17:24	steamwebhelper.exe	Steam
30	29.04.2025 17:24	steamwebhelper.exe	Спеціальні пропозиції

Рисунок 2.3 – Таблиця БД TrainingData

Експортуємо отримані дані до .cvs файлу та вручну додаємо мітку – ActivityType (рис. 2.4), де 2 – продуктивна, 1 – нейтральна та 0 – непродуктивна активність.

Id	TimeStamp	ProcessName	WindowsTitle	Label
1	29.04.2025	trainingtrackingactiv.exe	MainWindow	2
2	29.04.2025	explorer.exe	Program Manager	2
3	29.04.2025	db browser for sqlite.exe	DB Browser for SQLite	2
4	29.04.2025	db browser for sqlite.exe	Вибрати файл бази даних	2
5	29.04.2025	db browser for sqlite.exe	DB Browser for SQLite – D:\Бакалавр	2
6	29.04.2025	chrome.exe	диплом саша – Робота над диплом	2
7	29.04.2025	chrome.exe	ADO.NET і C# | Підключення до баз	2
8	29.04.2025	chrome.exe	Клас StackOverflowException (систе	2
9	29.04.2025	chrome.exe	Нова вкладка – Google Chrome	1
10	29.04.2025	searchhost.exe	Пошук	1
11	29.04.2025	startmenuexperiencehost.e	Пуск	1
12	29.04.2025	explorer.exe	Вікно переповнення області завдан	1
13	29.04.2025	spotify.exe	Spotify	0
14	29.04.2025	spotify.exe	Spotify Premium	0
15	29.04.2025	steamwebhelper.exe	Steam	0
16	29.04.2025	steamwebhelper.exe	Спеціальні пропозиції	0

Рисунок 2.4 – Навчальні дані с міткою ActivityType

Для зручності використано Model Builder – графічний інтерфейс для створення, навчання та тестування моделей.

На основі навчальної таблиці побудовано модель класифікації тексту. Найбільш успішною є модель LbfgsLogisticRegression, в якої вірогідність правильної відповіді 96,6%, що є ідеальним показником для реалізації ПЗ. Оскільки модель має аналізувати вхідні дані і класифікувати, а не запам’ятовувати правильні відповіді, то дана вірогідність є найбільш оптимальною.

Model Builder надає можливість одразу ж перевірити модель (рис. 2.5). Вводячи вхідні дані отримаємо результат класифікації.

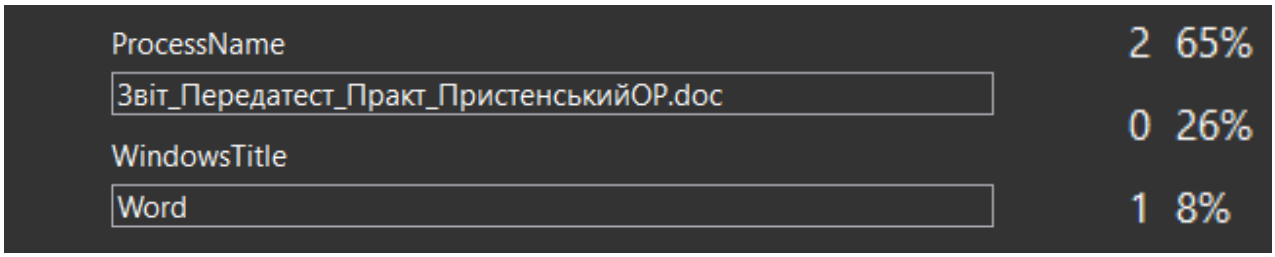


Рисунок 2.5 – Перевірка побудованої моделі класифікації

Далі інтегруємо модель у ПЗ. Кожен раз при зміні вікна чи вкладки активного процесу ПЗ фіксує назви та заголовки та передає у модель для класифікації (рис. 2.6).

```
//класифікація МН
var sampleData = new ClasificationOfProductivity.ModelInput()
{
    ProcessName = lastProcessName,
    WindowsTitle = lastWindowTitle,
};
var result = ClasificationOfProductivity.Predict(sampleData);

int activityType = (int)result.PredictedLabel;
```

Рисунок 2.6 – Інтеграція у ПЗ

Отриманий PredictedLabel використовується для статистичного аналізу активності, побудови графіків і розрахунку продуктивності за день.

Перевагами даного методу класифікації є відсутність потреби в ручному налаштуванні списків програм. Також є можливість натренувати модель під конкретну професію або вид діяльності для більш точного результату класифікації.

Класифікація активності на основі машинного навчання є гнучким і достатньо точним інструментом аналізу продуктивності.

2.4 Висновки до другого розділу

В даному розділі розроблено алгоритм роботи програмного забезпечення, що чітко визначає взаємодію між компонентами системи та послідовність виконання кожної функції. Алгоритм допомагає уникнути суперечностей між компонентами у майбутньому.

Розроблено структуру бази даних для збереження подальшій класифікації та обробки отриманого обсягу інформації. Спроектована БД відповідає архітектурі та вимогам ПЗ оцінювання виконання задач.

Розроблено модель машинного навчання для класифікації активності за критерієм продуктивності. Запропонована модель є досить гнучкою та класифікує типи активності працівників з точністю 97%, що є достатнім для розроблення програмного забезпечення оцінювання виконання робочих задач для раціонального розподілу часу.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація бази даних

На етапі реалізації ПЗ прийнято рішення об’єднати таблиці задач в одну. Об’єднання таблиць спрощує реалізацію функціональної частини розроблюваного додатку та забезпечує централізоване управління даними, зменшує складність запитів, а також полегшує подальшу підтримку й розширення системи.

Оновлена таблиця Tasks містить назву, опис та пріоритет задач, також зазначається час та дата створення задачі. При створені статус, коментар та час оцінки задачі мають пусті рядки, які заповнюються по мірі оцінки.

Для реалізації моделі машинного навчання розроблено додаткову та тимчасову таблицю TrainingData. Ця таблиця не пов’язана з іншими та виконує функцію збору інформації, на якій навчено модель ML. Цю таблицю вирішено не вносити в структуру БД, оскільки це лише тимчасовий інструмент.

Всі інші таблиці були реалізовані згідно розробленій структурі БД. Оновлену структуру можемо побачити на рисунку 3.1.

Створення та налаштування бази даних SQLite відбувається в програмному коді. Для цього створено окремий клас «Database.cs», де будуть записані скрипти ініціалізації в правильному порядку, які формують коректну структуру таблиць, індексів, зовнішніх ключів та налаштувань PRAGMA.

В класі відбувається не тільки ініціалізація, а також всі звернення до БД. Це може бути записи отриманих даних у конкретні таблиці або звернення отримання даних.

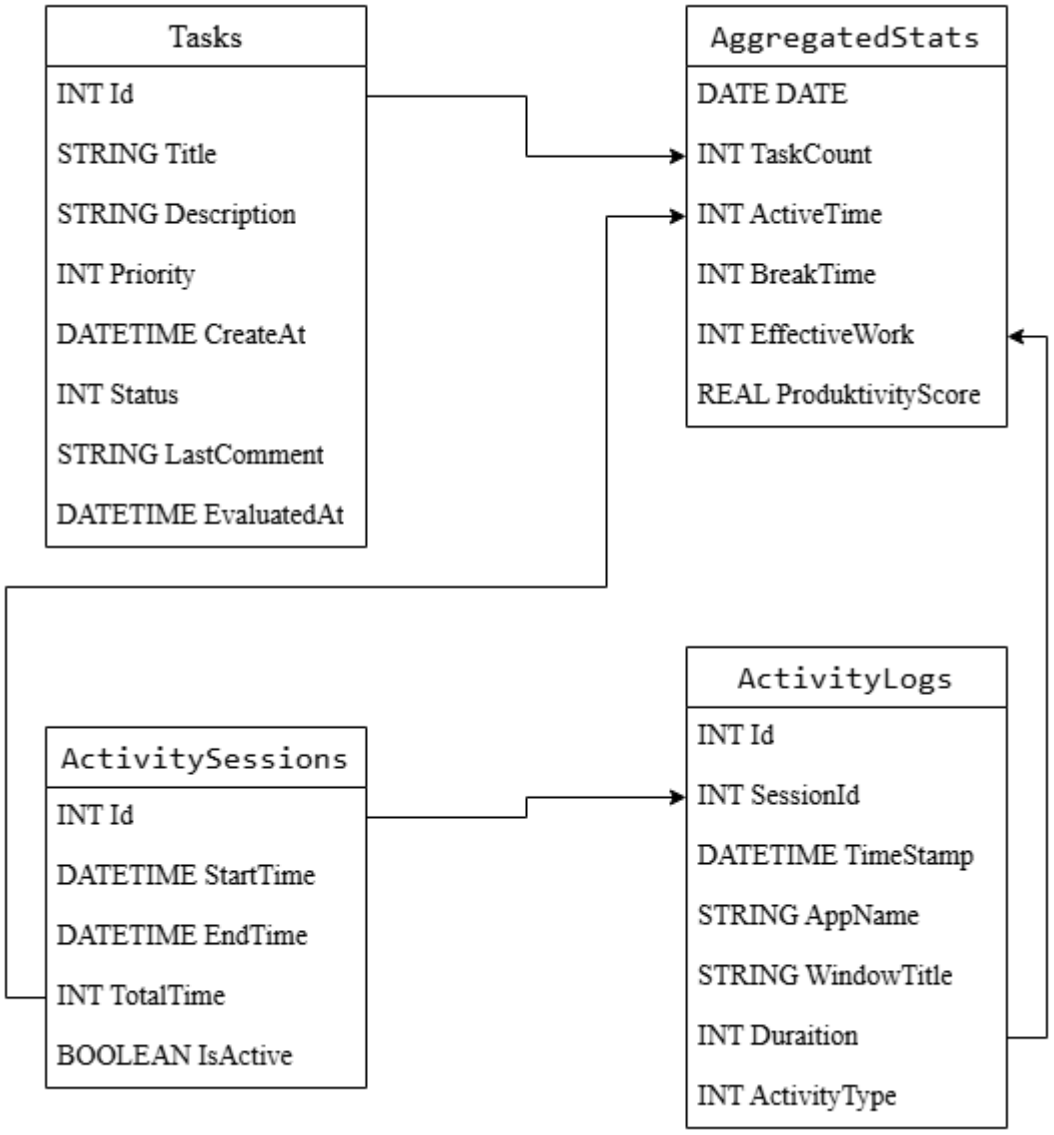


Рисунок 3.1 – Структура БД для ПЗ (створено в draw.io)

Ініціалізація і звернення відбувається за допомогою команди CreateCommand(). Для прикладу наведено скрипт ініціалізації однієї з таблиць на рисунку 3.2.

Перевагою цього класу є те, що навіть якщо видалити вже заповнену БД, то при новому запуску програми з’явиться новий файл, але вже з пустою базою даних. Так само відбувається і при першому запуску ПЗ.

```
var command = connection.CreateCommand();
//Таблиця для тренування МН
command.CommandText = @"
    CREATE TABLE IF NOT EXISTS TrainingData (
        Id INTEGER PRIMARY KEY AUTOINCREMENT,
        TimeStamp TEXT NOT NULL,
        ProcessName TEXT NOT NULL,
        WindowsTitle TEXTNOT NULL,
        UNIQUE(ProcessName, WindowsTitle)
    )";
command.ExecuteNonQuery();
```

Рисунок 3.2 – Приклад команди звернення до БД

У результаті реалізовано та налаштовано чотири основні таблиці у БД SQLite: ActivitySessions, ActivityLogs, Tasks та AggregatedStats, а також одна додаткова тимчасова таблиця TrainingData. Реалізували скрипти запису у БД та отримання даних. Повний лістинг класу Database.cs знаходиться у додатку А.

3.2 Реалізація модулю збору та обробки даних

Збір даних відбувається під час використання користувачем його комп'ютера. Для реалізації модулю збору створено клас «Tracking.cs». В цьому класі записані основні функції та методи отримання даних активності, деякі з цих методів записані в основному файлі програми.

Для отримання назви активного процесу та заголовку відкритого вікна використано бібліотеку WinAPI. Дана бібліотека має два методи необхідних для ПЗ, одним з них є GetForegroundWindow(), який повертає дескриптор (handle) поточного активного вікна, що дозволяє зрозуміти, у якій програмі або документі знаходиться користувач у момент фіксації. Після цього викликається метод GetWindowText(), який копіює текстовий заголовок активного вікна у буфер типу StringBuilder, де формується рядок.

Використовуючи ці методи написано функції отримання та відображення отриманих заголовків та назв активного процесу GetActiveWindowTitle() та GetActiveWindowProcessName().

Повний лістинг класу Tracking.cs знаходиться у додатку Б.

Ще одною бібліотекою для відстеження активності користувача є GMA.System.MouseKeyHook. Дана бібліотека надає методи для перехоплення подій клавіатури та миші [11]. На початку роботи відбувається ініціалізація глобального хуку-обробників, після чого реєструються події натискання клавіш, рух та кліки миші. На рисунку 3.3 наведений лістинг реалізації методів даної бібліотеки.

```
private void SetupGlobalHooks()
{
    globalHook = Hook.GlobalEvents();
    globalHook.KeyDown += (s, e) => {
        keyPressCount++;
        _lastInputTime = DateTime.Now;
    };
    globalHook.MouseDown += (s, e) => {
        mouseClickCount++;
        _lastInputTime = DateTime.Now;
    };
    globalHook.MouseMove += (s, e) =>
    {
        _lastInputTime = DateTime.Now;
    };
    globalHook.MouseWheel += (s, e) =>
    {

        _lastInputTime = DateTime.Now;
    };
}
```

Рисунок 3.3 – Лістинг функції реєстрації подій миші та клавіатури

Функція SetupGlobalHooks() реалізує підрахунок кількість натискань клавіш та миші, які заносяться в таблицю ActivityLog, а також кожною дією оновлює таймер _lastIntputTime. Цей таймер необхідний для відстеження чи знаходиться користувач перед комп’ютером або ж відлучився на паузу.

В ПЗ закладена можливість налаштувати поріг неактивності, тобто час бездіяльності, за який користувач вважається неактивним. Для цього маємо змінну TimeSpan _inactivityThreshold, який дорівнює TimeSpan.FromMinutes(5).

Якщо система не реєструє події з клавіатури та миші протягом 5 хвилин, то активна сесія закривається і відкривається неактивна.

Введення та оцінка задач є одним з головних пунктів реалізованої системи. Основною метою цієї частини ПЗ є об'єктивність і прозорість процесу, щоб виключити суб'єктивні упередження. Прийнято рішення, що користувач сам буде вводити задачі, після виконання завдання сам оцінювати. ПЗ буде оцінювати витрачений час користувача на виконання задач. Цей факт робить систему автоматизованою.

Для кожної задачі передбачено чіткий набір параметрів, таких як заголовок, опис, дата створення та рівень пріоритету. Такий підхід змушує користувача одразу визначити атрибути задачі.

При оцінці задач фіксуються інший набір параметрів, наприклад час оцінки, статус виконання задачі, а також є можливість для користувача написати коментар, щодо результату виконаної роботи.

Після накопичення сирих даних про активність користувача, наведених вище, дані потребують обробки. Обробка відбувається двома способами: класичним методами та використання машинного навчання.

Під класичними методами розуміється написання алгоритмів очищення та сортування даних. Основні з функцій обробки даних записані в клас Database.cs, зокрема методи нормалізації форматів полів до єдиного стандарту або видалення дубльованих даних, якщо це потрібно для подальшої роботи. Також відбувається перевірка коректності отриманих даних перед тим як будуть записані до бази даних. Повний лістинг класу Database.cs знаходиться у додатку А.

Машинне навчання використовуємо для класифікації логів.

Реалізовано модулі збору та обробки інформації, які поєднують різні функціональні можливості. Даний інструмент, завдяки інтеграції машинного навчання, досить гнучкий та автоматизований. Розроблені модулі з прозорим введенням та оцінкою задач надають об'єктивну та надійну основу для подальшої роботи ПЗ оцінки виконання задач для раціонального розподілу часу.

3.3 Реалізація користувацького інтерфейсу

Інтерфейс користувача побудовано у рамках архітектури MVVM (Model-View-ViewModel) з використанням WPF (.NET). Програмне забезпечення складається з трьох вікон: головне вікно, менеджер задач та аналітика. Роздивимось архітектуру системи та окремо кожне вікно.

Загальна архітектура інтерфейсу користувача складається з моделей, уявлень, управління представленням та зв'язками.

У Model входять всі класи-сутності та сервіси до бази даних. Розроблене ПЗ містить шість моделей, серед яких TaskRecord, SessionRecord, ActivityLog та AggregatedStatsRecord призначені для зчитування та запису одного рядка відповідної таблиці бази даних. Склад названих моделей містять прості рядки з визначеними типами змінних, що відповідають даним, які будуть зчитуватись та записуватись в БД, а також вказані get та set, які й відповідають за зчитування та запис параметрів.

Також маємо дві моделі: TaskSummary та TopApplicationUsage. TaskSummary є допоміжною моделлю, яка відображає ключову статистику задач у системі. Дана модель містить зведені показники за певний проміжок часу, таких як, загальну кількість задач, повністю, частково виконаних або невиконаних задач, а також виконаних та назначених сьогодні задач. Ці показники розраховуються з таблиці Tasks.

TopApplicationUsage – це модель, яка допомагає відображати рейтинг програм за часом використання. Допоміжна модель містить назву програми, загальний час використання та місце у рейтингу. Джерело даних для цієї моделі – це таблиця ActivityLog з БД.

ViewModel містить класи для кожного вікна, безпосередньо класи містять властивості для двонаправленого зв'язку з уявленнями (View), команди для обробки подій з кнопок та інших елементів керування.

У випадку розроблюваного ПЗ реалізовано MainViewModel, яка містить функції завантаження рейтингу додатків за часом використання, сумарної

кількість задач, зміни продуктивності відносно попереднього дня, а також команд збереження оцінених задач.

View являє собою розмітку, яка описує розміщення та стилі всіх елементів у вікні. Код уявлення містить мінімум логічних елементів, окрім виклику ініціалізації DataContext і обробка специфічних подій.

Головна сторінка є стартовим вікном, де розпочинається взаємодія користувача із застосунком. Головне вікно поєднує статус активності, кнопки швидкого доступу та коротку статистику.

Статус активності містить заголовок активного вікна, назву процесу та результат класифікації, який має три варіанти – продуктивна, нейтральна і непродуктивна активність.

Під статусом активності знаходяться блок навігації, який складається з двох кнопок швидкого доступу. Користувач може відкрити вікно менеджеру задач або ж детальну аналітику.

Статус активності і блок навігації можемо побачити на рисунку 3.4.

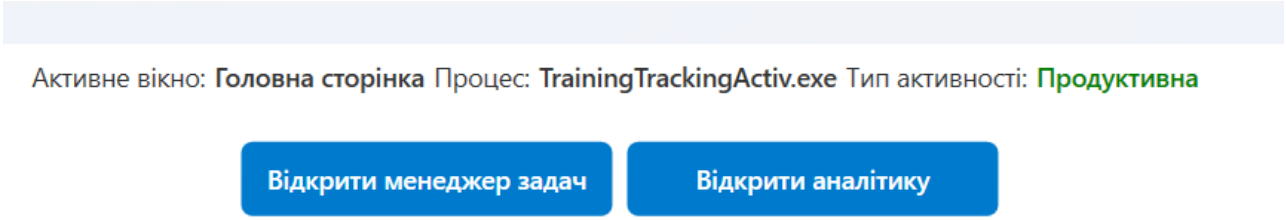


Рисунок 3.4 – Статус активності та навігація на головному вікні

Після кнопок навігації знаходиться блок короткої статистики. Кожен з елементів блоку оформлений у вигляді картки.

Першим елементом блоку є картка «Ваша продуктивність», яка відображає динамічну метрику продуктивності таку, як приріст або спад у відсотковому відношенні щодо попереднього дня. Продуктивність у відсотках беремо з таблиці AggregatedStats з БД.

Наступним елементом є картка «Топ програм за часом». Картка містить горизонтальний бар-чарт, де ліворуч відображається назва програми,

горизонтальна смуга, яка пропорційна витраченому часу, та точний час у форматі годин, хвилин та секунд. Дані беруться з таблиці ActivityLogs.

Останнім елементом статичного блоку на головному вікні є картка «Статистика задач», яка відображає числові показники щодо задач. Картка показує загальну кількість створених задач, кількість задач із статусами «Не виконано», «Виконано частково» та «Виконано», а також кількість задач назначених та виконаних сьогодні. Кожен показник виокремлений своїм кольором, що допомагає швидко зорієнтуватися. Всі показники та дані взято з таблиці Tasks з БД.

Всі елементи з блоку статистики та головне вікно можна побачити на рисунку 3.5.

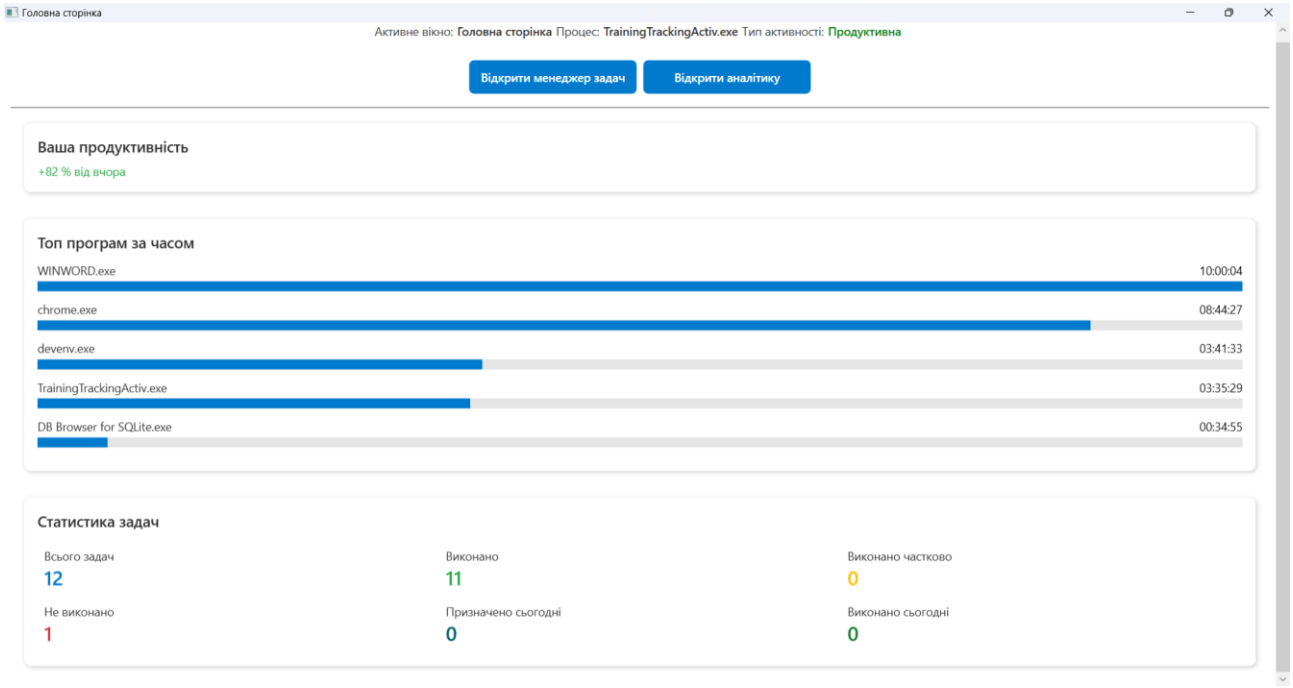


Рисунок 3.5 – Головне вікно

Розроблена структура головного вікна дозволяє користувачу одним поглядом отримати деяке уявлення про свій робочий процес і перейти вже до деталей у відповідних вікнах або розділах.

Одним з важливих складових програмного забезпечення є вікно «Менеджер задач». Вікно призначене для створення нових задач та оцінки або регулювання вже існуючих. Сторінка поділена на дві частини.

Ліворуч знаходиться картка з можливістю додати нову задачу, для чого необхідно заповнити поля назви та опис задачі, а також вибрати пріоритет від 1 до 5, де чим більший показник, тим вище є пріоритет (рис. 3.6). Після заповнення полів необхідно натиснути кнопку «Додати» і нова задача одразу ж з’явиться праворуч. Картка «Додати нову задачу» закріплена на одному місці, навіть при великій кількості задач для виконання.

При додаванні задачі у БД записуються назва, опис, пріоритет та дата створення. У всі інші стовпці таблиці Tasks записується значення «NULL».

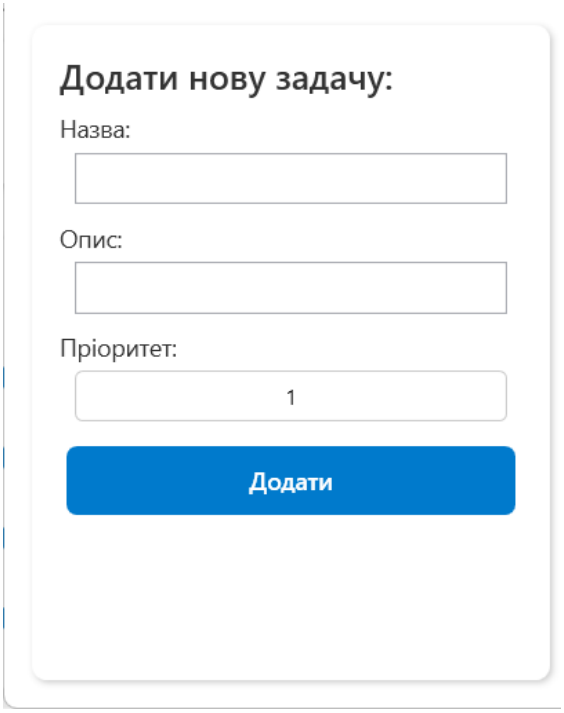


Рисунок 3.6 – Вигляд картки «Додати нову задачу»

Праворуч на сторінці знаходиться список задач для виконання (рис. 3.7). Кожна задача стилізована під окрему картку, де вказано заголовок, опис задачі. Після виконання користувач має можливість оцінити задачу, вибравши статус виконання роботи – «Виконано», «Виконано частково» та «Не виконано», а

також опціонально є можливість написати коментар щодо виконаної задачі. За замовчуванням статус виконання задачі стоїть «Не виконано».

Коли користувач оцінив задачу та можливо написав коментар, має бути натиснута кнопка «Зберегти». При натисканні кнопки оновлюється рядок з таблиці Tasks, куди записуються параметри статус, коментар та час оцінки задачі.

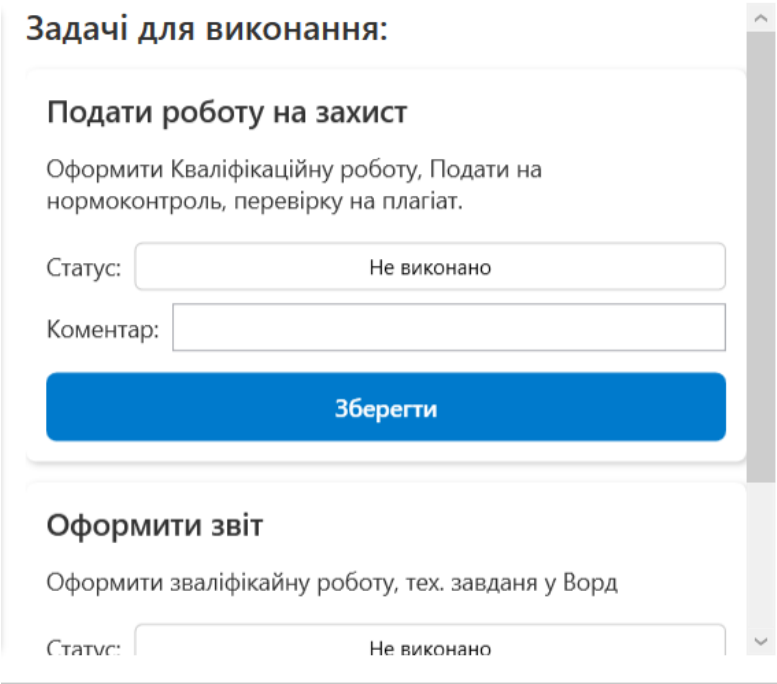


Рисунок 3.7 – Вигляд список задач для виконання у вікні «Менеджер задач»

Вікно «Менеджер задач» забезпечує інтуїтивно зрозумілий для користувача інтерфейс, де є можливість додати нові, переглянути та оцінити вже існуючі задачі. Повний вигляд вікна зображено на рисунку 3.8.

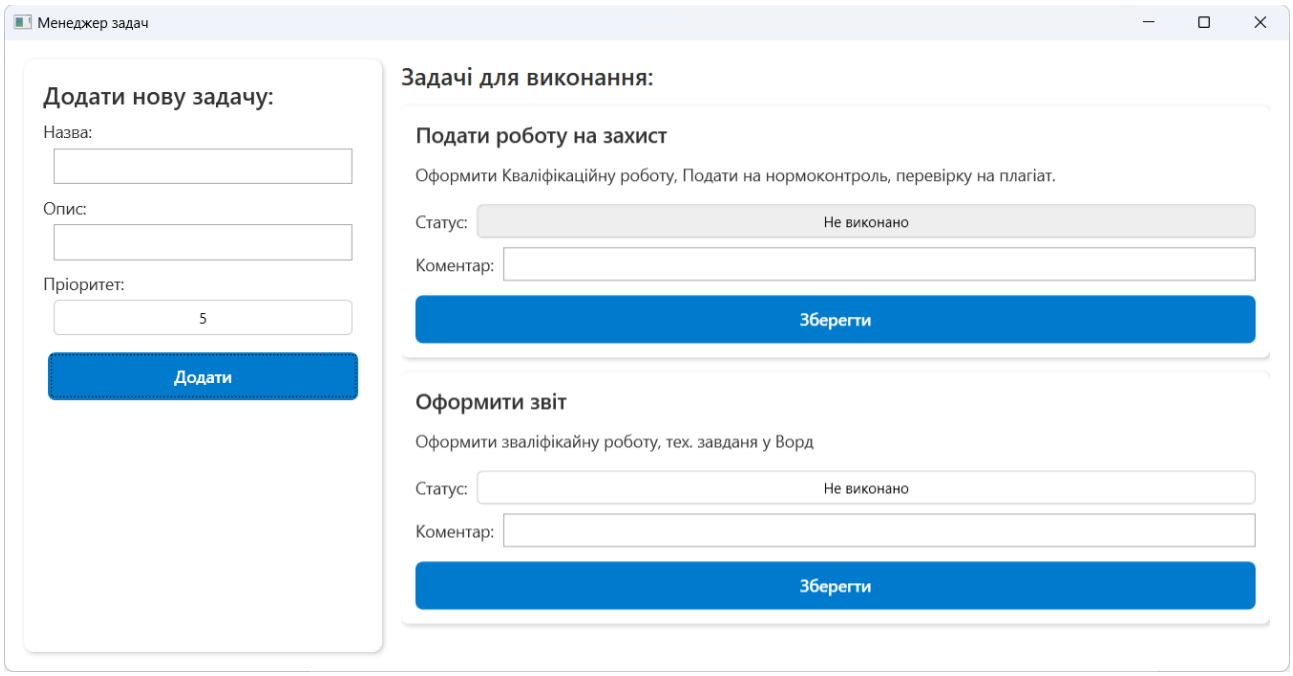


Рисунок 3.8 – Вікно «Менеджер задач»

Третє вікно розроблюваного програмного забезпечення є «Аналітика». Дане вікно призначене для більш детального аналізу продуктивності користувача та раціонального розподілу часу.

Вікно аналітики містить переважно графіки із зображенням та порівнянням зібраних даних і класифікації. Вікно поділене на дві частини, де перша являє собою огляд активності за вибраний день, а друга частина – це аналіз за вибраний період.

Аналіз за день починається з графіку розподілу за типами активності (рис. 3.9). На діаграмі показано, яку частку від активного часу припадає на продуктивну, нейтральну та непродуктивну роботу.

Кругова діаграма реалізована за допомогою бібліотеки LiveCharts.WPF через PieChart. Розміри секцій пропорційні кількості часу у кожній категорії. Реалізовано підказку справа, яка допомагає ідентифікувати який колір до якої категорії відноситься.

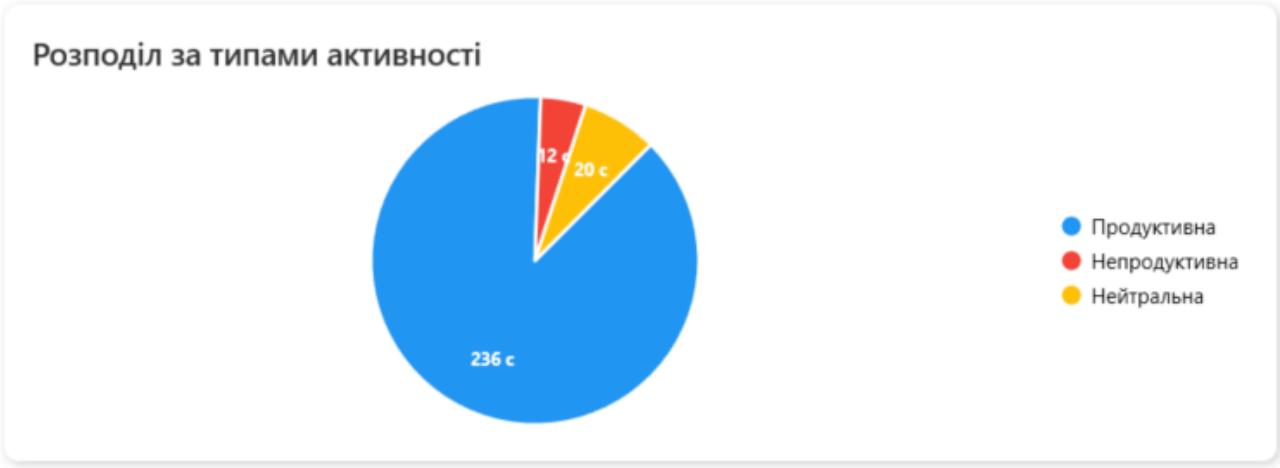


Рисунок 3.9 – Кругова діаграма «Розподіл за типами активності»

Графік «Розподіл за типами активності» дає швидке розуміння на скільки продуктивним був день і на скільки ефективно користувач використовує свій час.

Наступним графіком є стовпчиковою діаграмою, яка відображає, скільки часу протягом відповідної години користувач був активним за комп'ютером. Графік «Час за комп'ютером» зображений на рисунку 3.10.

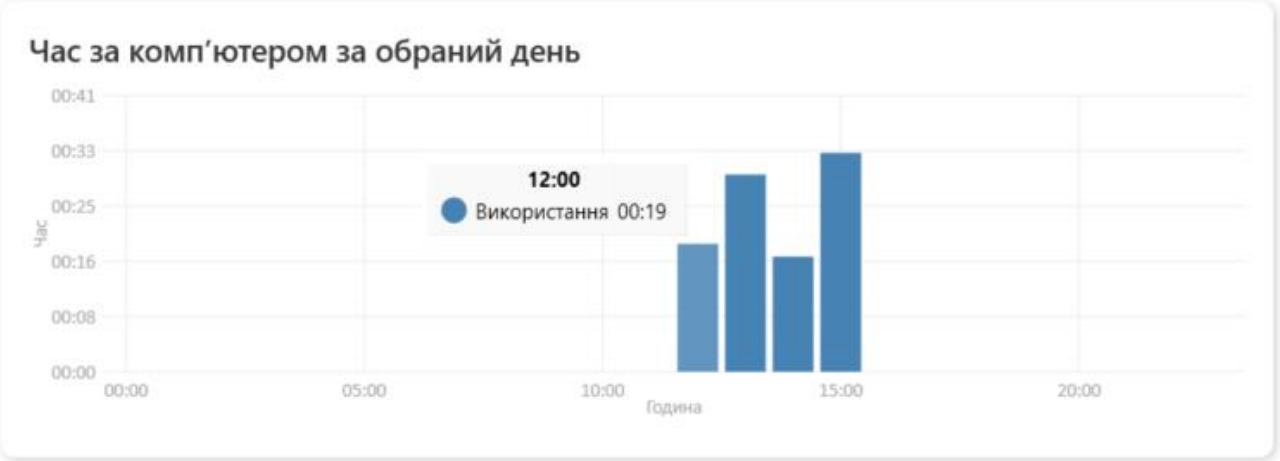


Рисунок 3.10 – Стовпчикова діаграма «Час за комп'ютером»

Візуалізація даного графіку дозволяє швидко оцінити, у які інтервали часу користувач був найбільш активним за комп'ютером.

Останній графік за денну аналітику показує кількість сесій та їх тривалість. На гістограмі по осі X відкладено значення тривалості, які знаходяться в межах 0-1 хв., 1-5 хв., 5-15 хв., 15-30 хв. та більше 30 хв., а на осі У – кількість сесій, що потрапили до відповідного інтервалу. На рисунку 3.11 зображено приклад відображення гістограми «Тривалість сесій».

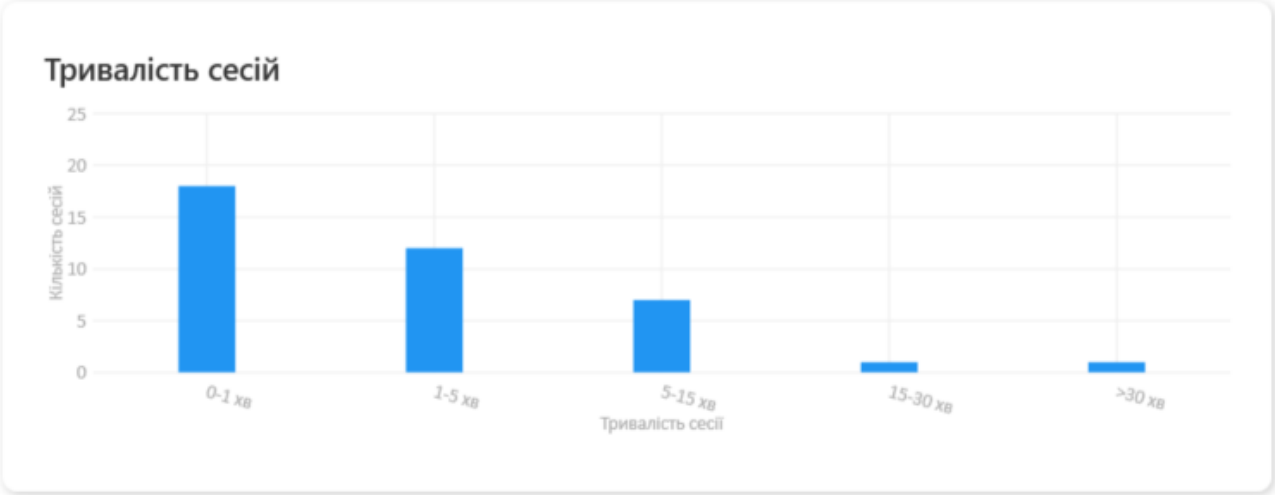


Рисунок 3.11 – Гістограма «Тривалість сесій»

Гістограма дає зрозуміти на скільки зосереджено працює користувач, як часто відволікається або ж навпаки, безперервно працює. Велика кількість сесій в межах 0-1 хв. свідчить про часті відволікання.

Гістограма «Тривалість сесій» є корисним інструментом для оцінки використаного часу.

Наступні графіки відносяться до періодичного аналізу, та можуть показати результати аналізу для порівняння від двох днів.

Графік «Продуктивність за період» призначений для відображення вже розрахованих показників продуктивності з таблиці AggregatedStats у БД. На осі X графіку відкладений вибраний проміжок часу, а по осі У – відсоток продуктивності. Кожна крапка відповідає значенню продуктивності за конкретний день. Лінія з’єднує кожну крапку у хронологічному порядку.

Графік «Продуктивність» за період з першого по одинадцяте червня зображений на рисунку 3.12.

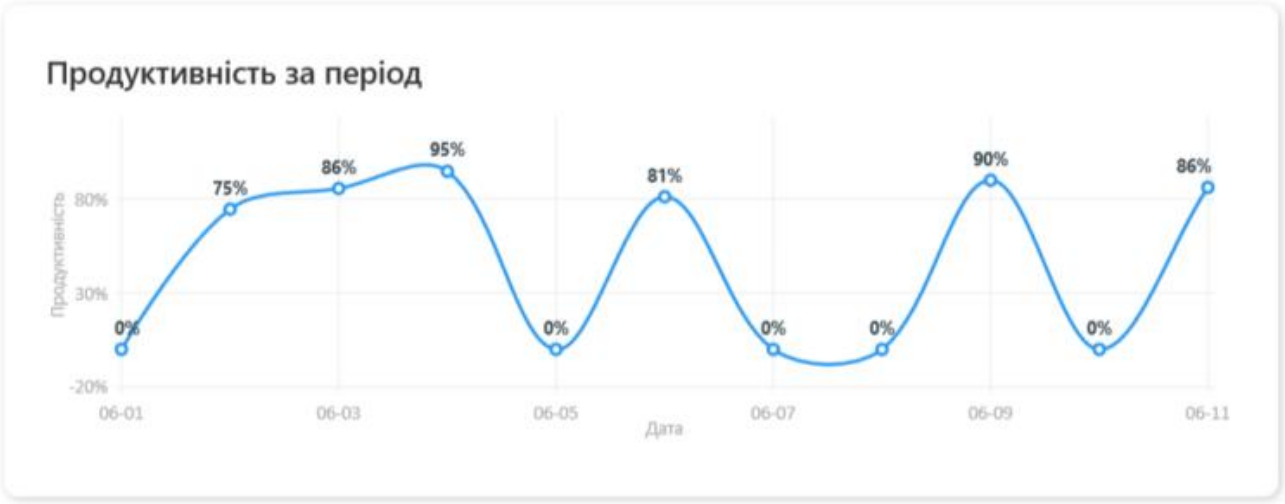


Рисунок 3.12 – Графік «Продуктивність за період»

Даний графік допомагає порівняти ефективність користувача за певний період, а також швидко виявити дні, коли система не фіксувала активність працівника зовсім (вихідні, відпустка) або ж не фіксувала продуктивні активності.

Останній графік являє собою зображення часу активності за вибраний період часу. По осі X графіку «Час активності за період» відкладаються дати, а по осі У відкладається час у секундах. Графік показує дві лінії, де синя лінія відповідає активному часу, коли сесія є активною, а червона відповідає часу перерв, коли користувач не виконував жодної активності. Крапки відповідають значенням кількості часу активності та не активності. Дані для відображення взято з БД таблиці ActivitySessions, де відповідно значенню IsActive (True або False) сесії визначається її активність.

На рисунку 3.13 зображений такий графік за період з першого по одинадцяте червня.

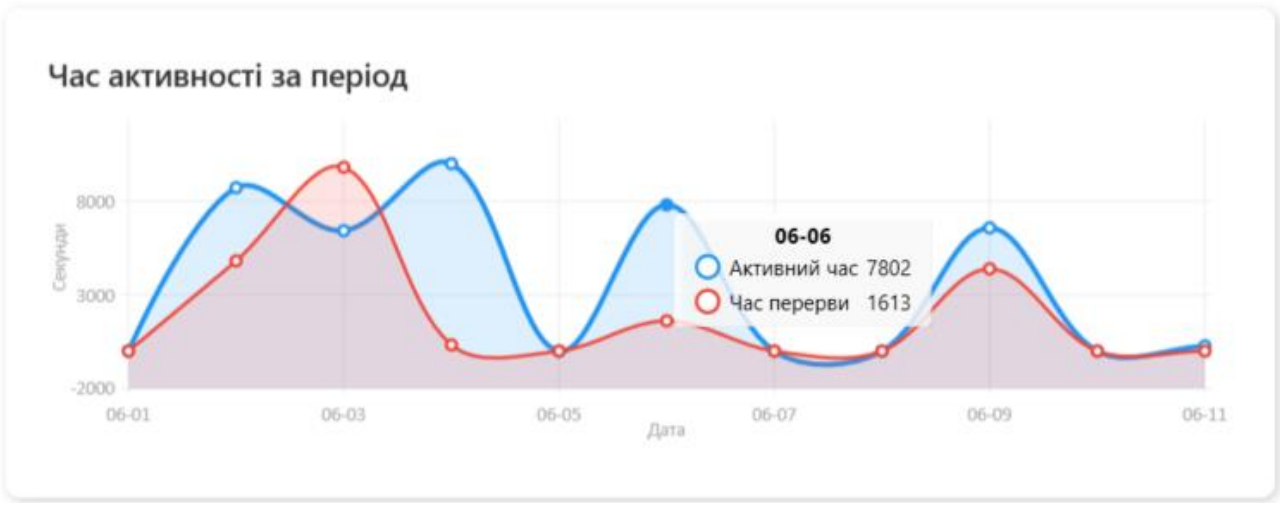
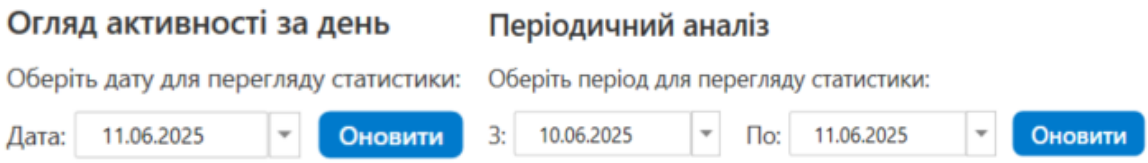


Рисунок 3.13 – Графік «Час активності за період»

Такий графік дозволяє проаналізувати два чинники, а саме порівняти дні за робочим навантаженням і дні відпочинку або низької активності. Другим чинником є оцінка співвідношення активного часу та перерв за кожен день.

Реалізовано прості елементи керування DatePicker для вибору однієї дати (рис. 3.14, а) або потрібного періоду часу (рис. 3.14, б).



а) б)

Рисунок 3.14 – Елементи вибору

Таким чином з цих графіків та елементів вибору дати і періоду складається вікно аналітики. Вікно допомагає роздивитися під різними кутами та оцінити ефективність використання часу користувачем. Всі елементи виконані в одному стилі, текст та графіки максимально зрозумілі для аналізу.

У результаті реалізації користувацького інтерфейсу створено інтуїтивно зрозумілий інструмент для відстежування активності та оцінки виконання задач користувачем. Завдяки використаним технологіям та бібліотекам використаних при реалізації, досягли високої швидкодії, гучності, а також досить красивий

зовнішній вигляд. Інтерфейс відповідає принципам зручності, прозорості та має потужну основу для розширення.

3.4 Перевірка та тестування реалізованого програмного забезпечення

Перевірка та тестування застосунку спрямовані на підтвердження коректності роботи всіх ключових елементів та функцій програмного забезпечення. Тестування включають в себе тестові завдання з сценарієм для перевірки форм та модулів у різних умовах.

Тестові сценарії забезпечують перевірку на цілісність застосунку, наявність дефектів, а також перевірка критично важливих елементів системи. Тестові завдання та результати перевірки занесено у таблицю 3.1.

Таблиця 3.1 – Тестові сценарії для перевірки ПЗ

№	Сценарій	Вхідні дані	Очікуваний результат
1	2	3	4
Сценарії для вікна «Менеджер задач»			
1	Додавання нової задачі з коректними полями	Назва: Test Опис: Test Text Пріоритет: 3	Задача з’являється у списку, всі поля очищуються, у головному вікні зміняться значення «Всього задач», «Не виконано»
2	Спроба додати без заголовка	Назва: - Опис: - Пріоритет: 4	Показ повідомлення про помилку, запис не створюється

Продовження таблиці 3.1

1	2	3	4
3	Оцінка вибраної задачі	Вибір задачі з заголовком «Test», вибрати значення статусу «виконано»	Запис зник із списку, у головному вікні зміниться к-сть задач «Виконано сьогодні» та «Виконано»
Сценарії для головного вікна			
4	Відкриття вікна «Менеджер задач»	Натиск кнопки «Відкрити менеджер задач»	Відкривається вікно з задачами, де праворуч – поля для додання нових задач, ліворуч – список всіх не виконаних або частково виконаних задач
5	Відкриття вікна «Аналітика»	Натиск кнопки «Відкрити аналітику»	Відкривається вікно, де завантажені графіки за аналізом за сьогоднішній день, та за період з попереднього дня до сьогодення
Сценарії для вікна «Аналітика»			
6	Побудова графіків з коректним періодом дат	Період: з 01.06.2025 по 07.06.2025	Графіки лінійного відображення продуктивності та активного часу будуються без помилок
7	Побудова графіків з некоректним періодом дат	Некоректний період дат, коли кінцева дата є раніше за початкову дату: з 10.06.2025 по 05.06.2025	Показ повідомлення про помилку, графік не будується

Процедура додання нової задачі з коректними вхідними даними та операція оцінювання вибраної задачі відповідають очікуваним результатам зазначеним в таблиці 3.1.

При спробі додати нову задачу без заповненого поля «Назва» призвело до повідомлення про помилку (рис. 3.15).

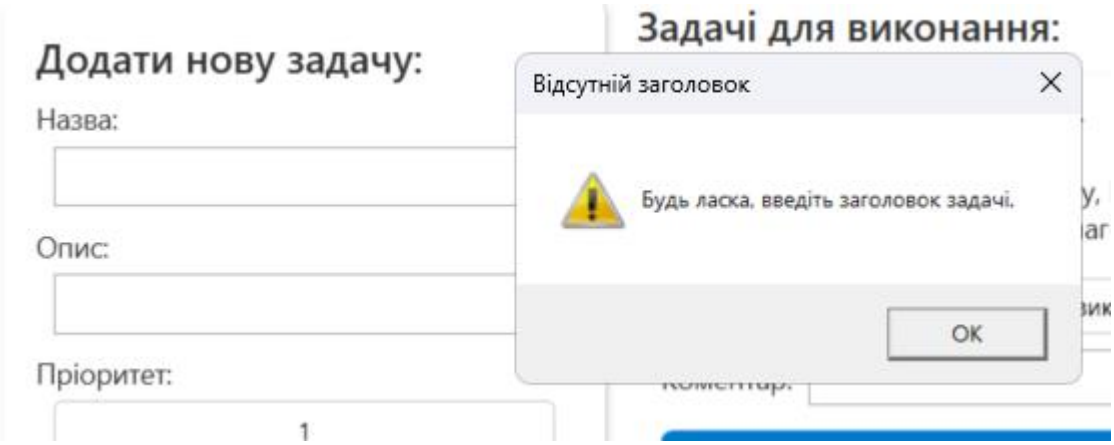


Рисунок 3.15 – Повідомлення про помилку при спробі додати задачу без назви

Ініціалізація вікон «Менеджер задач» та «Аналітика» через навігаційний модуль головного вікна відбувається без виникнення помилок або функціональних дефектів.

У вікні аналітики всі графіки відображаються, при вказанні коректного періоду дат графіки успішно будуються. При вказані невірного періоду виникає повідомлення про помилку (рис. 3.16), оновлення графіку не відбувається. Раніше побудований графік залишається до моменту вказання коректного періоду або перезапуску вікна «Аналітика».

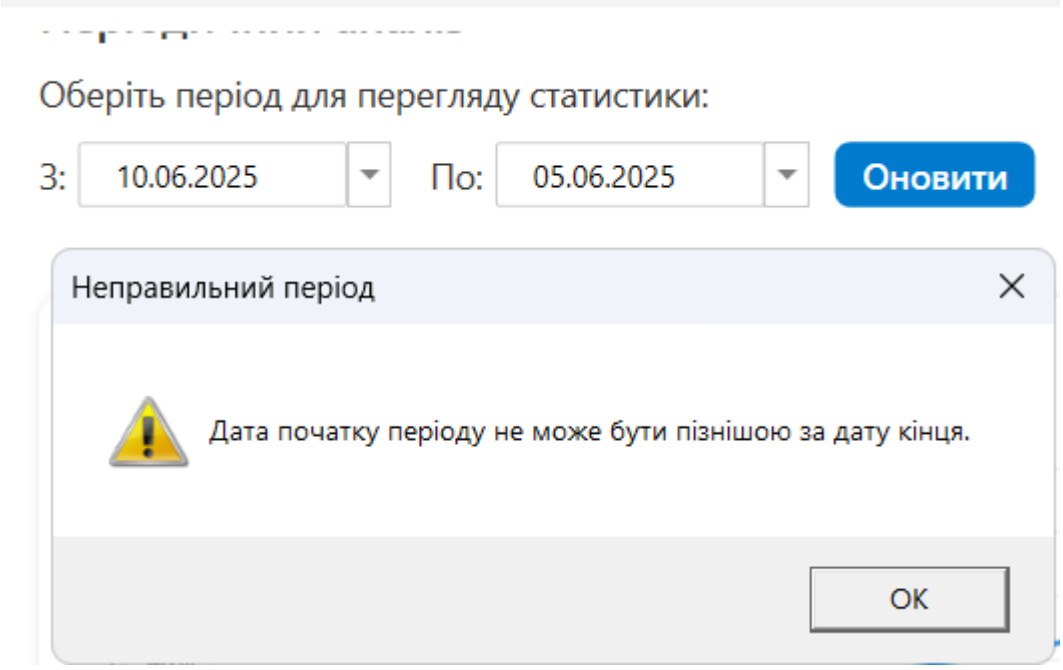


Рисунок 3.16 – Помилка про неправильні дати періоду

Виконання сценаріїв, наведених у таблиці 3.1, продемонструвало коректну поведінку всіх ключових елементів та функцій ПЗ. Усі тестові сценарії пройдені успішно. Відсутні критичні дефекти та помилки. Результати тестування відповідають запланованим критеріям.

3.5 Перевірка стійкості системи за критерієм Михайлова

Розроблення ПЗ для оцінки продуктивності користувача передбачає перевірку стабільності як елемента системи автоматизованого управління. Стабільність системи визначається здатністю системи за будь-яких збурень повертатися до передбачуваного режиму роботи без розбалансування та самозбудження. Збурення у ПЗ можуть виникати при змінні активності користувача або при завершенні сесії.

Програмне забезпечення оцінки продуктивності можна описати математично у вигляді передавальної функції другого порядку з одним нулем [12]. ПЗ приймає вхідні сигнали, такі як завершення сесії, зміна задачі та перехід в активний або неактивний стан, після чого обробляє їх і видає результат

у вигляді графіків та звітів. У спрощеному вигляді реакція на збурення описується формулою 3.1:

$$W(s) = \frac{K(s+z)}{(s+p_1)(s+p_2)}. \quad (3.1)$$

де: K = 1 – підсилення;

z = 0,05 – нуль системи (вибираємо для регулювання пікового відгуку);

p_1 = 0,2, p_2 = 0,1 – полюси системи, обрані відповідно до часу реакції ПЗ (5-10 секунд після завершення активної сесії) (3.2) – (3.3).

$$T_1 \approx 5 \text{ сек } => \ p_1 = \frac{1}{T_1} = 0{,}2; \quad (3.2)$$

$$T_2 \approx 10 \text{ сек } => \ p_2 = \frac{1}{T_2} = 0{,}1. \quad (3.3)$$

Аналізуючи перехідну характеристику бачимо, що відгук монотонно зростає, після чого стабілізується. Полюси знаходяться в лівій півплощині, отже система є асимптотично стійкою.

Перехідна характеристика програмного забезпечення зображена на рисунку 3.17.

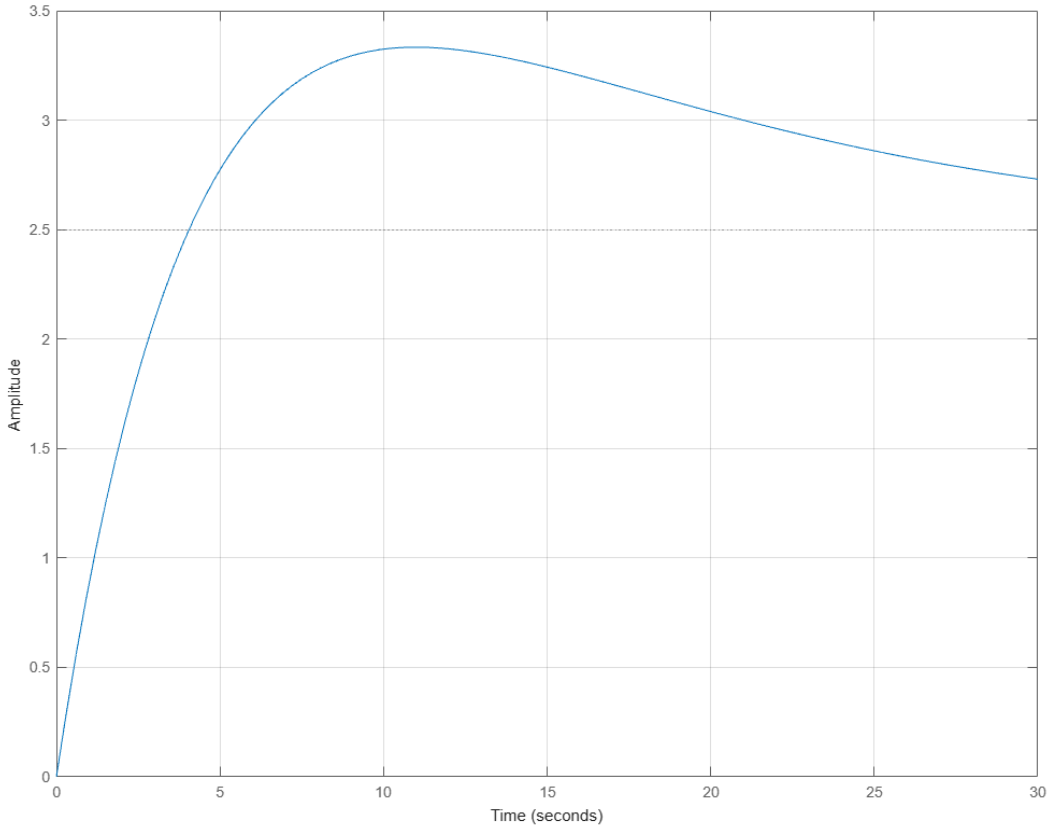


Рисунок 3.17 – Перехідна характеристика програмного забезпечення

Один із способів аналізу системи автоматизованого керування на стійкість є критерій стійкості Михайлова.

У критерії Михайлова нас цікавить характеристичне рівняння замкненої системи, яке описане (3.4):

$$D(s) = 1 + W(S) = 1 + \frac{(s+0{,}05)}{(s+0{,}2)(s+0{,}1)} = \frac{(s+0{,}2)(s+0{,}1)+(s+0{,}05)}{(s+0{,}2)(s+0{,}1)}. \quad (3.4)$$

Проведено розрахунок чисельнику нового виразу (D(s)) (3.5):

$$D(s) = s^2 + 0{,}3s + 0{,}02 + s + 0{,}05 = s^2 + 1{,}3s + 0{,}07. \quad (3.5)$$

На рисунку 3.18 представлено побудову кривої Михайлова для характеристичного рівняння (3.5).

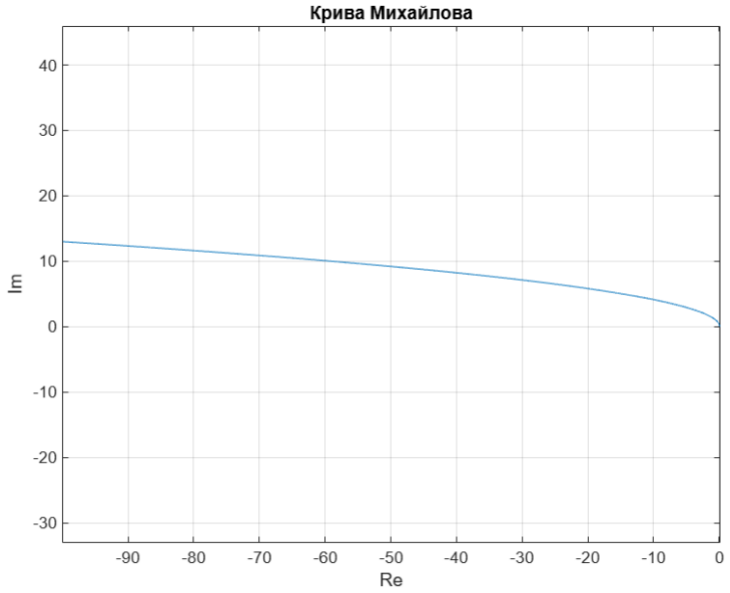


Рисунок 3.18 – Крива Михайлова для моделі програмного забезпечення

Крива прямує проти годинникової стрілки і не охоплює початок координат, що відповідає умовам стійкості за критерієм Михайлова.

3.6 Охорона праці

Основна мета підрозділу «Охорона праці» в кваліфікаційних роботах бакалавра – виявлення чинників, що спричиняють можливість виробничого травматизму, професійних захворювань, отруєнь, пожеж, вибухів, забруднення навколишнього середовища промисловими викидами під час експлуатації, виготовлення і застосування радіоелектронного устаткування на робочих місцях і в робочих приміщеннях, та розробка заходів, щодо техніки безпеки [13].

Перед тим як аналізувати можливі небезпечні і шкідливі виробничі фактори, роздивимося робоче місце. Розробник програмного забезпечення оцінки виконання завдання для раціонального розподілу часу в обладнані має комп’ютер або ноутбук з дисплеєм, клавіатуру, комп’ютерну мишу, а також мережеве підключення. Робоче приміщення обладнане штучним освітленням і

системою опалення. Повітря в кімнаті підтримується за температури в діапазоні від +18 до +24 °С та відносної вологості від 40% до 60 %.

Аналізуючи небезпечні та шкідливі виробничі фактори, маємо роздивитися ергономічні фактори. Тривале сидіння без перерв призводить до перевантажень опорно-рухового апарату, що може проявлятися, як болі в спині або остеохондроз.

Відповідно до Наказу Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65 «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» [14], виділимо декілька правил для робочого місця з монітором, клавіатурою.

Відстань від екрану до очей користувача має бути більшою за 600 мм, має корегуватися залежно від діагоналі екрана. Слід забезпечити можливість регулювання кута нахилу в межах ±30° від лінії зору. Клавіатура має розташовуватися на регульованій підставці або поверхні столу на відстані від 100 мм до 300 мм від його переднього краю та кут нахилу клавіатури має дорівнювати від 5° до 15°.

Роздивимось фактори промислової санітарії. Згідно з ДСанПіН 3.3.2.007-98 «Санітарно-гігієнічні вимоги до мікроклімату приміщень» [15], у робочому просторі слід забезпечити щоденне вологе прибирання робочих поверхонь із використанням миючих та дезінфікуючих засобів. Має бути регулярне провітрювання приміщення два рази на день не менше 15 хвилин. Для зняття статичного навантаження необхідна організація перерв у роботі за комп’ютером не рідше ніж через 45 – 60 хвилин.

Пожежна безпека є важливим фактором охорони праці, а правила до цього вже устаткувались і записані в ДБН В.1.1-7:2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги» [16]. На робочому місці з радіоелектронним устаткуванням має бути передбачено системи пожежної сигналізації, які забезпечують раннє виявлення пожежі та повідомлення про це централізованої диспетчерської. Шляхи для евакуації мають бути промаркірованими та карта з

ними має знаходитись в межах видимості. Має бути реалізований періодичний контроль справності електромереж.

3.7 Висновки до третього розділу

У третьому розділі реалізовано програмне забезпечення оцінки виконання задач для раціонального розподілу часу. Розділ складається з реалізації ключових елементів системи.

Реалізовано БД, всі її сутності у вигляді таблиць та зв’язки між ними. Створено і налаштовано схему SQLite з таблицями Tasks, ActivityLogs, ActivitySession, AggregatedStats, а також з тимчасовою таблицею TrainingData, необхідної для тренування моделі машинного навчання. Розроблено клас Database.cs, де записано код функцій ініціалізацій, а також операцій створення, оновлення, читання та видалення.

У результаті реалізації модулю збору та обробки інформації створено клас Tracking.cs з методами бібліотек WinAPI та GMA.System.MouseKeyHook, призначених для фіксації назв процесу, заголовків вікон та подій зчитаних з клавіатури та миші. До класу Database.cs додано класичні методи для нормалізації і запису даних у БД, а також інтегровано модель машинного навчання для класифікації типу активності.

Користувацький інтерфейс побудовано за MVVM-архітектурою в WPF. Реалізовано три вікна: Головне, Менеджер задач, Аналітика. Кожне вікно дотримується одного стиля та в одній кольоровій гамі. Елементи статистики оформлені у вигляді карток. Графіки та динамічні елементи виконані за допомогою бібліотеки LiveCharts.

Перевірено розроблене ПЗ на наявність дефектів та незапланованих повідомлень про помилки. Проведено тести за сценарієм на працездатність застосунку для перевірки коректності роботи. Підтверджено стабільну роботу всіх ключових елементів.

Дана модель є стійкою за критерієм Михайлова, тобто система швидко, плавно та без коливань реагує на зміну вхідних даних (збурення). Така реакція задовольняє потреби програмного забезпечення оцінки ефективності працівника для раціонального розподілу часу.

Визначено ергономічні, санітарно-гігієнічні вимоги та заходи пожежної безпеки.

ВИСНОВКИ

Під час кваліфікаційної роботи на першому етапі проведено аналіз аналогів ПЗ оцінювання ефективності, таких як Hubstaff і RescueTime. У результаті аналізу виявлено переваги та недоліки аналогічних систем, які було враховано для постановки задач при розробці програмного забезпечення.

Проаналізовано методи оцінювання ефективності працівників. Аналіз існуючих підходів показав, що найбільш релевантними для реалізації в межах даного ПЗ є трудовий метод та метод KPI, який в поєднанні стане теоретичною основою для побудови ефективної, об'єктивної системи оцінювання в умовах цифрового робочого середовища.

Обрано інструменти, які повністю відповідають функціональним вимогам та утворюють комплексну стабільну екосистему з можливістю масштабованості.

Розроблено алгоритм роботи програмного забезпечення, що чітко визначає взаємодію між компонентами системи та послідовність виконання кожної функції.

Розроблено структуру бази даних для збереження подальшої класифікації та обробки отриманого обсягу інформації. Спроектована БД відповідає архітектурі та вимогам ПЗ оцінювання виконання задач.

Розробили модель машинного навчання для класифікації активності за критерієм продуктивності. Розроблена модель класифікує з точністю 97%, що є оптимальним варіантом для розроблення даного програмного забезпечення, а також модель досить гнучка.

Розроблено БД дозволяє записувати, оновлювати, читати та за необхідності видаляти дані з таблиць. Для взаємозв’язку з базою даних реалізовано клас Database.cs, який містить методи ініціалізації, загальні виклики SQL та CRUD-методи.

Реалізований модуль збору та обробки даних успішно виконує фіксацію назв активних процесів та події зчитаних з клавіатури та комп’ютерної миші. Дані передаються та записуються у БД.

Інтуїтивно зрозумілий інтерфейс користувача забезпечує зручною для розуміння інформацією. Розроблений інтерфейс стилізований, має задані стилі для карток, графіків, шрифтів та кнопок. Графіки та динамічні елементи системи реалізовано за допомогою бібліотеки LiveCharts.

Проведено тестування розробленого програмного засобу сценаріями з коректними та некоректними даними. Функціональних дефектів та незапланованих повідомлень про помилки не виявлено, що вказує на стабільність системи.

Запропонована система є стійкою за критерієм Михайлова, тобто система швидко, плавно та без коливань реагує на зміну вхідних даних (збурення). Така реакція задовольняє потреби програмного забезпечення оцінки ефективності працівника для раціонального розподілу часу.

Проаналізовано ергономічні, санітарно-гігієнічні вимоги та заходи пожежної безпеки під час розроблення та використання автоматизованої системи моніторингу для раціонального розподілу часу.

Практична цінність розробленого програмного забезпечення полягає в можливості аналізувати активність та оцінювати ефективність виконання задач користувача, що сприяє раціональному розподілу часу.

Майбутні версії програмного забезпечення можуть бути значно вдосконалені шляхом інтеграції з хмарними сервісами для забезпечення синхронізації даних між пристроями та надання віддаленого доступу. Також перспективним є впровадження вбудованого календаря, системи нагадувань та гнучкого планування завдань в рамках єдиної платформи. Вдосконалення алгоритмів машинного навчання за допомогою використання більших, реалістичних наборів даних та впровадження механізмів самонавчання підвищить точність класифікації активності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2022 – 55 с.

3. Hubstaff. Офіційний сайт: інтелектуальна система обліку робочого часу працівників [Електронний ресурс]. – Режим доступу: https://hubstaff.com. – Дата звернення: 15.04.2025.

4. RescueTime. Система обліку цифрової продуктивності [Електронний ресурс]. – Режим доступу: https://www.rescuetime.com. – Дата звернення: 15.04.2025.

5. Освіта.ua. Аналіз продуктивності праці та методи її оцінювання [Електронний ресурс]. – Режим доступу: https://osvita.ua/vnz/reports/econom_pidpr/19413/ – Назва з екрана. – Дата звернення: 15.04.2025.

6. Microsoft. Introduction – C# Language Specification [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction. – Дата звернення: 15.04.2025.

7. Microsoft. Win32 API Reference [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/api. – Дата звернення: 14.06.2025.

8. ML.NET [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com/en-us/learn/ml-dotnet/what-is-mldotnet – Дата звернення: 07.05.2025.

9. SQLite Home Page [Електронний ресурс]. – Режим доступу: https://sqlite.org. – Дата звернення: 15.04.2025.

10. LiveCharts [Електронний ресурс]. – Режим доступу: https://v0.lvcharts.com. – Дата звернення: 14.06.2025.

11. GMA.System.MouseKeyHook [Електронний ресурс]. – Режим доступу: https://www.nuget.org/packages/MouseKeyHook . – Дата звернення: 14.06.2025.

12. Невлюдов І. Ш., Токарєва О. В. Теорія автоматичного управління (збірник задач): навч. посіб. – Харків : ХНУРЕ, 2020. – 240 с.

13. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» : навч. вказівки / упоряд. І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2023. – 64 с.

14. Державний комітет України з промислової безпеки, охорони праці та гірничого нагляду. Наказ від 26 березня 2010 р. № 65 «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» // Офіційний вісник України. – 2010. – № 12. – С. 17–22.

15. Міністерство охорони здоров'я України. Санітарно-гігієнічні вимоги до мікроклімату приміщень: ДСанПіН 3.3.2.007-98 (із змінами). – Київ, 1998.

16. Державні будівельні норми України. Пожежна безпека об'єктів будівництва. Загальні вимоги: ДБН В.1.1-7:2016. — Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2016.

17. Закон України "Провищу освіту" від 01.07.2014 No 1556-VII. [Електронний ресурс] Режим доступу: https://zakon.rada.gov.ua/laws/show/1556-18#Text. – Дата звернення: 14.06.2025.

18. Положення про організацію освітнього процесу у ХНУРЕ [Електронний ресурс] : Наказ ХНУРЕ від 27.11.2020 р. №400. Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-pro-organizaciju-osvitnogo-procesu-v-hnure.pdf. – Дата звернення: 14.06.2025.

19. Положення про академічну доброчесність [Електронний ресурс]: Наказ ХНУРЕ від 02 лютого 2021р. №50. Режим доступу: https://nure.ua/branch/akademichna-dobrochesnist-ta-zabezpechennja-jakosti-osviti. – Дата звернення: 14.06.2025.

20. Невлюдов І.Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 366 с.

21. Невлюдов І.Ш. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» : Навчальний посібник / І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова.– Харків: ХНУРЕ, 2023 . – 150 с.

22. Невлюдов І. Ш. Комп'ютерно-інтегровані технології виробництва технічних засобів автоматизації. Частина 1: підручник для студентів закладів вищої освіти ; Харків. нац. ун-т радіоелектроніки. – Харків : ФОП Панов А.М., 2021. – 604 с. ISBN 978-617-7947-67-6

23. Комплекс навчально-методичного забезпечення навчальної дисципліни «Безпека праці в індустрії ІТ-технологій» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [http://catalogue.nure.ua/knmz] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 122 с.