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 ПІДТРИМКИ ЗДОРОВОГО РЕЖИМУ РОБОТИ ЗА КОМП'ЮТЕРОМ, ВРАХОВУЮЧИ АДАПТИВНЕ ПЛАНУВАННЯ ПЕРЕРВ ТА ПЕРСОНАЛІЗОВАНІ РЕКОМЕНДАЦІЇ**
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-3
Худолій А. Ю.
(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. Руденко Д. О.
(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 Худолію Антону Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розроблення застосунку для підтримки здорового режиму роботи за комп’ютером, враховуючи адаптивне планування перерв та персоналізовані рекомендації

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література з питань ергономіки праці, охорони здоров'я при роботі за комп'ютером та проєктування програмного забезпечення, матеріали наукових конференцій, дані інтернет-мережі, документація платформи .NET 8 та фреймворку WPF, специфікація архітектурного патерну MVVM, документація бібліотеки Newtonsoft.Json, технічна документація YouTube oEmbed API, документація компонента Microsoft WebView2.

4. Перелік питань, що потрібно опрацювати в роботі
1. Гнучка система налаштування робочого графіка.
2. Своєчасне інформування користувача про перерви у зручній та ненав’язливій формі.
3. Наявність персоналізованих рекомендацій щодо здоров’я.
4. Інтеграція з зовнішніми сервісами для відтворення фонової музики.
5. Наявність унікального маскота-персонажа.
6. Простота у використанні.
7. Стабільна робота у фоновому режимі без перевантаження системних ресурсів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми здорового режиму роботи за комп'ютером, порівняльний аналіз існуючих програмних рішень, архітектура застосунку та структура проєкту, алгоритм відстеження робочого часу та перерв, інтерфейс застосунку «Chill out...», приклади спливаючих сповіщень з маскотом-лінивцем, результати тестування та аналіз споживання системних ресурсів.

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 | 15.04.26-17.04.26 | |
| 3 | Аналіз літератури з проблеми, що вирішується | 20.04.26-21.04.26 | |
| 4 | Аналіз існуючих методів розпізнавання | 22.04.26-23.04.26 | |
| 5 | Формування кроків вирішення проблеми | 24.04.26-26.04.26 | |
| 6 | Програмна реалізація | 27.04.26-05.05.26 | |
| 7 | Аналіз отриманих результатів | 25.05.26-03.06.26 | |
| 8 | Оформлення пояснювальної записки | 04.06.26-15.06.26 | |
| 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. Руденко Д. О._____
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 80 с., 7 табл., 26 рис., 33 джерел.

ЗАСТОСУНОК ДЕСКТОПНИЙ, ЗДОРОВ'Я, ПЕРЕРВА, ПЕРСОНАЛІЗАЦІЯ, ПЛАНУВАННЯ, РЕЖИМ РОБОТИ, СПОВІЩЕННЯ.

Об'єктом роботи є процес дослідження навантаження на користувача для підтримки здорового режиму роботи за комп'ютером з адаптивним плануванням перерв та персоналізованими рекомендаціями.

Метою роботи є розроблення застосунку, що дозволяє користувачам налаштовувати індивідуальний робочий графік, автоматично отримувати нагадування про перерви та кінець робочого часу, а також ознайомлюватись із персоналізованими рекомендаціями щодо збереження здоров'я під час роботи за комп'ютером.

Під час роботи використано: методи проєктування інтерфейсів користувача (UI/UX-дизайн, мінімалістична естетика, адаптивна верстка), методи розробки десктопних застосунків (планування подій, системні сповіщення), а також підходи до персоналізації користувацького досвіду.

Область застосування: офісні працівники та фрілансери, студенти, що тривало працюють за комп'ютером, люди, що дбають про здоров'я очей та опорно-рухового апарату.

DESKTOP APPLICATION, HEALTH, BREAK, PERSONALIZATION, PLANNING, WORK MODE, NOTIFICATION.

The object of the work is the development of an application for supporting a healthy computer work routine with adaptive break scheduling and personalized recommendations.

The aim of the work is to develop an application that allows users to configure an individual work schedule, automatically receive notifications about upcoming breaks and the end of working hours, as well as access personalized recommendations for maintaining health during computer work.

During the work, the following were used: user interface design methods (UI/UX design, minimalist aesthetics, adaptive layout), desktop application development methods (event scheduling, system notifications), and approaches to user experience personalization.

Application domain: office workers and freelancers, students engaged in prolonged computer work, individuals concerned about eye health and musculoskeletal well-being.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МБ – мегабайт

МКХ-11 – Міжнародна класифікація хвороб 11-го перегляду

МРТ – магнітно-резонансна томографія

ОЗП – оперативний запам'ятовувальний пристрій

API – Application Programming Interface (інтерфейс програмування застосунків)

BCL – Base Class Library (базова бібліотека класів)

CLR – Common Language Runtime (загальне середовище виконання)

CPU – Central Processing Unit (центральний процесор)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

CVS – Computer Vision Syndrome (синдром комп'ютерного зору)

GUI – Graphical User Interface (графічний інтерфейс користувача)

JSON – JavaScript Object Notation (нотація об'єктів JavaScript)

JVM – Java Virtual Machine (віртуальна машина Java)

MAUI – Multi-platform App UI (багатоплатформний інтерфейс застосунків)

MVVM – Model-View-ViewModel (модель-представлення-модель представлення)

UI – User Interface (інтерфейс користувача)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

WPF – Windows Presentation Foundation (фундамент представлення Windows)

XAML – eXtensible Application Markup Language (розширювана мова розмітки застосунків)

XML – eXtensible Markup Language (розширювана мова розмітки)

# ВСТУП

У сучасному світі інформаційні технології займають центральне місце в усіх сферах людської діяльності. Стрімкий розвиток комп’ютерної техніки, програмного забезпечення та мережевих технологій докорінно змінив підходи до роботи, навчання, спілкування та відпочинку. Цифровізація охопила практично всі галузі − від медицини та освіти до промисловості та фінансів, створивши нову реальність, у якій людина щодня проводить значну частину свого часу у взаємодії з комп’ютерами та цифровими пристроями.

Разом із безперечними перевагами, які несе технологічний прогрес, з’явились і нові виклики. Одним із найбільш актуальних є проблема збереження здоров’я людей, що тривалий час працюють за комп’ютером. Сидячий спосіб роботи, постійне навантаження на зоровий апарат, неправильна постава та відсутність регулярних перерв призводять до розвитку хронічних захворювань опорно-рухового апарату, погіршення зору та загального зниження продуктивності праці.

Сучасне програмне забезпечення відіграє ключову роль у вирішенні подібних проблем. Завдяки можливостям автоматизації, персоналізації та інтелектуального планування, застосунки здатні не лише нагадувати користувачу про необхідність відпочинку, а й адаптуватися до його індивідуального режиму роботи, формуючи здорові звички та підвищуючи якість робочого процесу.

Актуальність даної роботи зумовлена зростаючою кількістю людей, що проводять більшість робочого часу за комп’ютером, а також недостатньою розповсюдженістю зручних і доступних інструментів для контролю режиму праці та відпочинку. Розробка застосунку для підтримки здорового режиму роботи за комп’ютером з адаптивним плануванням перерв та персоналізованими рекомендаціями є практичною відповіддю на цей виклик.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТП ІСНУЮЧИХ РІШЕНЬ

## 1.1 Проблема здоров’я при тривалій роботі за комп’ютером

У сучасному суспільстві комп’ютер став невід’ємною частиною повсякденного життя. Більшість офісних працівників, програмістів, дизайнерів, студентів та представників багатьох інших професій проводять за екраном від восьми до дванадцяти годин на добу. Таке навантаження, якщо його не контролювати, неминуче призводить до погіршення здоров’я та зниження якості роботи.

Тривала робота за комп’ютером без належних перерв є однією з найбільш поширених причин розвитку цілого ряду захворювань та патологічних станів. Серед них особливо виділяються проблеми з опорно-руховим апаратом − болі в спині, шиї та плечах, що виникають через тривале перебування в незручній або неправильній позі. Також широко розповсюдженим явищем стає комп’ютерний зоровий синдром, який характеризується втомленістю очей, сухістю, головними болями та тимчасовим погіршенням зору [22, 29].

Окрему увагу заслуговує проблема психологічного вигорання та хронічної втоми, яка нерідко є наслідком відсутності структурованого режиму праці та відпочинку. Людина, що безперервно працює протягом багатьох годин, поступово втрачає концентрацію, стає більш схильною до помилок та відчуває загальне зниження продуктивності − навіть якщо суб’єктивно їй здається, що вона «щойно сіла» за комп’ютер.

Відповідно до досліджень у галузі ергономіки та медицини праці, регулярні короткі перерви кожні 45-60 хвилин роботи здатні суттєво знизити навантаження на зоровий апарат, зменшити напруження м’язів спини та покращити загальний психоемоційний стан людини. Проте у реальних умовах більшість людей нехтують цим правилом − або через відсутність звички, або через те, що просто не відстежують час. Саме тут на допомогу

приходять спеціалізовані програмні засоби для контролю режиму праці та відпочинку [18, 20, 30].

Варто також зазначити, що проблема організації здорового робочого процесу є особливо актуальною для людей, що працюють вдома у форматі дистанційної роботи або на фрілансі. На відміну від офісного середовища, де обідні перерви та перерви на каву є частиною загальноприйнятого розкладу, домашні умови часто не передбачають чіткого розмежування між роботою та відпочинком. Це призводить до того, що людина може годинами сидіти за комп’ютером, навіть не помічаючи плину часу.

Таким чином, автоматизація нагадувань про перерви, контроль тривалості робочих сесій та надання персоналізованих рекомендацій щодо здоров’я є практично значущою задачею, розв’язання якої здатне реально покращити якість роботи та самопочуття широкого кола користувачів.

## 1.2 Ергономіка робочого місця та рекомендації фахівців

Ергономіка − це науковий напрямок, що вивчає взаємодію людини з робочим середовищем з метою оптимізації умов праці та мінімізації шкідливих впливів. Стосовно роботи за комп’ютером ергономіка охоплює широкий спектр рекомендацій: від правильного положення тіла та налаштування крісла до відстані до монітора та рівня освітленості приміщення (рис. 1.1).

Одним із ключових аспектів є правильна постава. Фахівці рекомендують сидіти рівно, утримуючи природний вигин хребта, зі ступнями, що повністю торкаються підлоги або спеціальної підставки. Спинка крісла повинна підтримувати поперековий відділ хребта, а лікті − знаходитись приблизно на рівні столу, утворюючи прямий кут у 90 градусів. Недотримання цих правил протягом тривалого часу призводить до

формування хронічних захворювань хребта, зокрема остеохондрозу та грижі міжхребцевих дисків [27, 32].

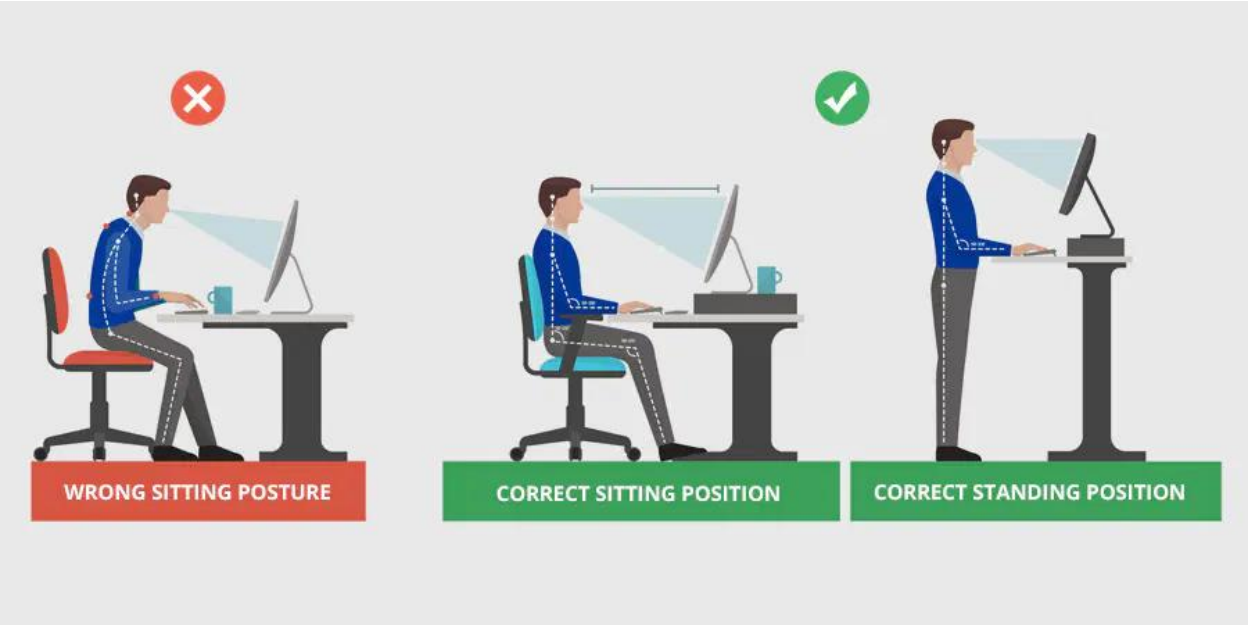


Рисунок 1.1 − Ергономіка робочого місця

Монітор також потребує правильного позиціонування. Верхній край екрану повинен знаходитись на рівні очей або трохи нижче, а відстань до екрану має становити не менше 50-70 сантиметрів залежно від розміру дисплею. Яскравість монітора необхідно налаштовувати відповідно до освітленості приміщення − занадто яскравий або занадто темний екран однаково шкідливо впливає на зір. Рекомендується також використовувати матові екрани або спеціальні антиблікові фільтри для зменшення відблисків від зовнішніх джерел світла [27, 31].

Для зниження навантаження на очі широко використовується правило: кожні 20 хвилин роботи за екраном слід протягом 20 секунд дивитись на об’єкт, розташований на відстані не менше 6 метрів. Цей простий прийом дозволяє розслабити м’язи кришталика ока та значно зменшити відчуття зорової втоми [23, 25].

Крім того, фахівці рекомендують регулярно виконувати гімнастику для очей − набір нескладних вправ, що включають переведення погляду у різних

напрямках, фокусування на близьких та далеких предметах, а також легкий масаж повік. Ці вправи допомагають підтримувати тонус очних м’язів та запобігають прогресуванню короткозорості у людей, схильних до неї.

Важливою складовою ергономічного підходу є і психологічний відпочинок. Зміна виду діяльності під час перерви − наприклад, короткі фізичні вправи, прогулянка або навіть звичайна розтяжка − значно ефективніша для відновлення працездатності, ніж пасивне сидіння на місці. Саме тому в сучасних офісах дедалі популярнішими стають зони відпочинку з можливістю фізичної активності.

## 1.3 Психологічні аспекти та методи управління робочим часом

Ефективна організація робочого часу є не лише питанням продуктивності, але й важливою складовою психічного здоров'я людини. Дослідження у галузі когнітивної психології та нейронауки накопичили значний масив даних про те, як людський мозок функціонує в умовах тривалого інформаційного навантаження та яким чином структуровані перерви впливають на здатність до зосередження, творчого мислення та емоційної стабільності.

Одним із ключових понять у контексті тривалої комп'ютерної роботи є концепція ментальної втоми (mental fatigue). На відміну від фізичної втоми, яка добре відчувається суб'єктивно і сигналізує про необхідність відпочинку, ментальна втома розвивається поступово і часто залишається непомітною для самої людини. Людина продовжує працювати, відчуваючи суб'єктивне відчуття зайнятості та активності, тоді як об'єктивні показники якості роботи − швидкість обробки інформації, точність рішень, здатність до аналізу − вже суттєво знизились.

Нейрофізіологічна основа цього явища пов'язана з виснаженням запасів глюкози та нейромедіаторів у префронтальній корі головного мозку −

ділянці, відповідальній за планування, прийняття рішень та контроль імпульсів. Дослідження за допомогою функціональної МРТ показують, що після 45-50 хвилин безперервної інтелектуальної роботи активність префронтальної кори помітно знижується, тоді як активність лімбічної системи, відповідальної за емоційні реакції, зростає. Це пояснює, чому втомлена людина стає більш дратівливою, імпульсивною та схильною до прокрастинації.

Важливою психологічною концепцією є також теорія виснаження його вольових ресурсів (ego depletion), запропонована психологом Роєм Баумайстером. Згідно з цією теорією, здатність людини до самоконтролю, концентрації та прийняття зважених рішень є обмеженим ресурсом, який витрачається протягом дня. Кожне зусилля волі − чи то опір відволіканням, чи то прийняття складного рішення, чи то підтримання уваги на нецікавому завданні − дещо зменшує цей ресурс. Регулярні паузи дозволяють частково відновити вольовий ресурс і підтримувати ефективність роботи протягом усього дня [18].

Окремого розгляду заслуговує феномен «цифрового стресу» (technostress) − психологічного стану, що виникає внаслідок необхідності постійно адаптуватися до нових технологій, опрацьовувати великі обсяги інформації та перебувати в стані постійної доступності (always-on culture). Дослідження у цій галузі свідчать, що постійна доступність через месенджери та електронну пошту підтримує людину у стані хронічного стресу навіть у позаробочий час. Це призводить до погіршення якості сну, зниження відновлювальних можливостей організму та поступового розвитку синдрому емоційного вигорання.

Синдром емоційного вигорання (burnout) є одним із найбільш серйозних наслідків хронічного перевантаження в умовах інтенсивної комп'ютерної роботи. Всесвітня організація охорони здоров'я у 2019 році офіційно включила вигорання до Міжнародної класифікації хвороб (МКХ-11) як «синдром, що виникає внаслідок хронічного стресу на робочому місці,

з яким не вдається впоратися». Вигорання характеризується трьома компонентами: відчуттям виснаження, зростаючою психічною дистанційованістю від роботи та зниженням професійної ефективності [19].

З точки зору превентивних заходів, структуровані перерви є одним із найбільш доступних і ефективних інструментів профілактики вигорання. Дослідження, проведені Emilyann Spiegel та командою Університету Іллінойсу, показали, що короткі відволікання від завдання не лише не знижують продуктивність, але й запобігають погіршенню результатів при тривалій роботі. Групи досліджуваних, що робили короткі перерви кожні 40-50 хвилин, демонстрували стабільно вищі показники якості роботи протягом чотиригодинного робочого блоку порівняно з групами без перерв [21].

Серед науково обґрунтованих методів управління робочим часом, що спираються на описані психологічні закономірності, особливого поширення набули кілька підходів. Техніка Pomodoro, розроблена Франческо Чірілло у 1992 році, базується на чергуванні 25-хвилинних робочих сесій із 5-хвилинними паузами та більш тривалою перервою кожні чотири цикли. Метод 52/17, виявлений дослідниками сервісу DeskTime на основі аналізу поведінки найпродуктивніших працівників, передбачає 52 хвилини інтенсивної роботи та 17 хвилин повноцінного відпочинку [30]. Принцип ультраніанських ритмів (ultradian rhythms) обґрунтовує природну циклічність активності мозку з піками кожні 90-120 хвилин, що відповідає оптимальній тривалості робочого блоку [22, 24, 26].

Порівняльний аналіз найбільш поширених технік управління робочим часом наведено у таблиці 1.1.

Аналіз таблиці 1.1 свідчить про те, що не існує універсальної техніки, однаково ефективної для всіх користувачів та всіх типів завдань. Оптимальний ритм роботи та відпочинку є індивідуальним і залежить від типу виконуваних завдань, рівня досвіду та особистих особливостей нервової системи. Саме тому програмні засоби для управління режимом роботи мають забезпечувати гнучкість налаштувань, а не нав'язувати єдину методику.

Таблиця 1.1 − Порівняльний аналіз технік управління робочим часом

| **Техніка** | **Інтервал роботи** | **Інтервал відпочинку** | **Для кого підходить** | **Ефективність** |
|---|---|---|---|---|
| Pomodoro | 25 хв | 5 хв / 15–30 хв | Широке коло користувачів | Висока |
| 52/17 | 52 хв | 17 хв | Досвідчені фахівці | Висока |
| DeskTime 90/20 | 90 хв | 20 хв | Творчі спеціальності | Середня |
| Правило 20-20-20 | 20 хв | 20 секунд | Захист зору | Висока |
| Time blocking | Довільно | Заплановано | Менеджери, планувальники | Середня |

Важливим психологічним аспектом є також питання сповіщень та їх сприйняття користувачем. Дослідження у галузі людино-комп'ютерної взаємодії показують, що характер та форма нагадування суттєво впливають на те, наскільки ймовірно людина відреагує на нього. Нав'язливі, агресивні або занадто часті сповіщення швидко стають подразником і викликають тенденцію до їх ігнорування або повного відключення − ефект, відомий як «втома від сповіщень» (notification fatigue). Навпаки, ненав'язливі, позитивно забарвлені нагадування з чітким та зрозумілим повідомленням сприймаються значно краще і з більшою ймовірністю спонукають до бажаної дії [19].

Використання персонажів та маскотів у системах нагадувань є одним із підходів до підвищення їх ефективності. Дружній персонаж, що асоціюється з позитивними емоціями та корпоративним або продуктовим стилем, знижує психологічний опір до сповіщення і формує емоційний зв'язок між користувачем та системою. Це підтверджується дослідженнями в галузі геймдизайну та поведінкової економіки, де встановлено, що антропоморфні

елементи інтерфейсу підвищують залученість користувача та сприяють формуванню стійких звичок [21].

На рисунку 1.2 наведено порівняльну схему впливу різних рівнів ментальної втоми на продуктивність та якість прийнятих рішень. З графіку видно, що без регулярних перерв продуктивність вже через дві години знижується на 20-30%, тоді як при структурованих паузах вона залишається стабільною протягом усього робочого дня.

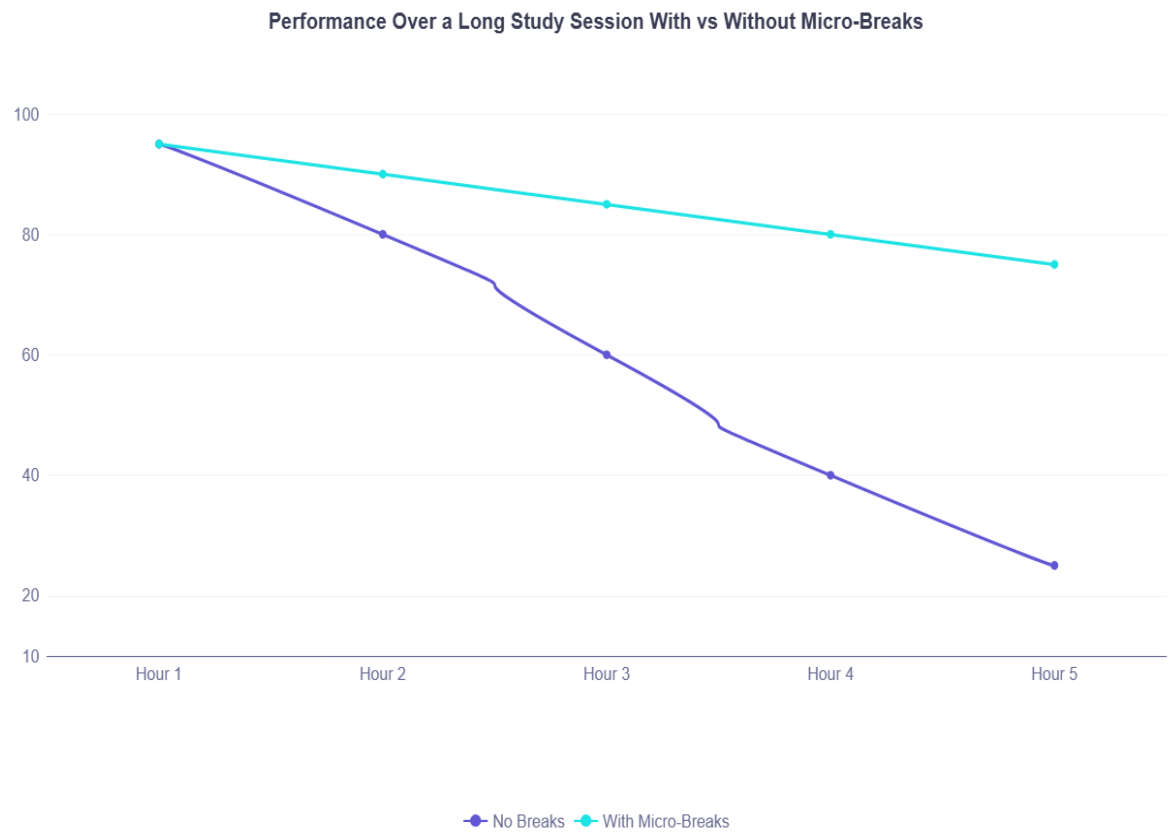


Рисунок 1.2 − Вплив ментальної втоми на продуктивність протягом робочого дня

Таким чином, психологічні дослідження однозначно підтверджують, що регулярні структуровані перерви є не лише засобом збереження фізичного здоров'я, але й необхідною умовою підтримання когнітивної продуктивності, емоційної стабільності та довгострокового психічного благополуччя. Програмні засоби, що автоматизують управління режимом

праці та відпочинку, мають враховувати ці психологічні закономірності та забезпечувати гнучкість налаштувань для адаптації до індивідуальних особливостей кожного користувача. Важливим є також грамотне проєктування системи сповіщень − вони мають бути ненав'язливими, позитивно забарвленими та зрозумілими, щоб не викликати психологічного опору та ефективно формувати здорові звички у довгостроковій перспективі.

## 1.4 Огляд існуючих програмних рішень

На сьогоднішній день на ринку програмного забезпечення існує певна кількість застосунків, спрямованих на підтримку здорового режиму роботи за комп’ютером. Проаналізуємо найбільш відомі з них, щоб зрозуміти, які функції вже реалізовані у наявних рішеннях, а де існують прогалини, що залишаються незаповненими.

Stretchly − це безкоштовний застосунок з відкритим вихідним кодом, доступний для Windows, macOS та Linux. Його основна функція − нагадування про мікропаузи та більш тривалі перерви. Застосунок має мінімалістичний інтерфейс та широкі можливості налаштування інтервалів перерв. Однак він не пропонує жодних персоналізованих рекомендацій щодо здоров’я, не має можливості відтворення музики та не підтримує кастомізацію зовнішнього вигляду сповіщень (рис. 1.3).

Time Out − застосунок, доступний виключно для macOS, що вирізняється приємним зовнішнім виглядом. Він пропонує налаштовувані перерви з плавним затемненням екрану та можливістю пропустити паузу. Платна версія розширює функціонал додатковими налаштуваннями. Проте відсутність версії для Windows та вартість повного функціоналу обмежують аудиторію цього рішення.

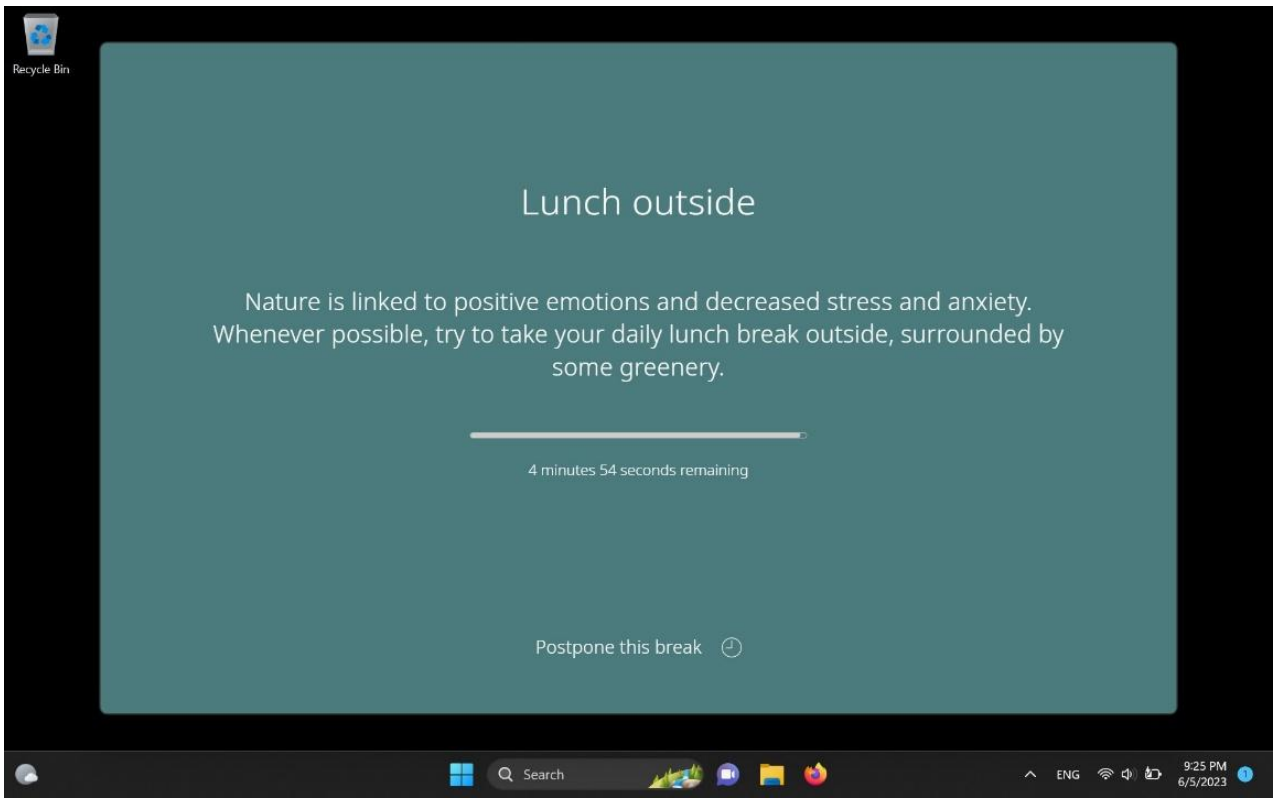


Рисунок 1.3 − Нагадування про обідню перерву з застосунку Stretchly

WorkRave − ще один безкоштовний застосунок, орієнтований насамперед на профілактику − синдрому повторюваних навантажень. Він містить набір вправ для очей та рук, що відображаються у вікні під час перерви, а також базову статистику використання. Проте його інтерфейс виглядає значно застарілим, і він не підтримується на macOS. Відсутність сучасного дизайну та обмежені можливості кастомізації роблять його менш привабливим для сучасних користувачів (рис. 1.4).

Focus Booster побудований навколо техніки Pomodoro − методики управління часом, що передбачає роботу сесіями по 25 хвилин з п’ятихвилинними перервами між ними. Застосунок ведеться облік сесій, формує звіти продуктивності та надає хмарну синхронізацію. Однак він є платним, а жорстка прив’язка до методології Pomodoro не підходить тим, хто хоче налаштувати власний індивідуальний графік роботи та перерв.


Рисунок 1.4 – Розтяжка плечей рук в застосунку WorkRave

EyeLeo – спеціалізований застосунок для захисту зору, що нагадує користувачу про вправи для очей через заданий інтервал. Він має симпатичного леопарда, що демонструє вправи, та підтримує налаштування частоти нагадувань. Однак застосунок доступний лише для Windows, не містить загальних рекомендацій щодо здоров'я та не підтримує музичні функції (рис. 1.5).


Рисунок 1.5 – Нагадування для розминки очей в застосунку EyeLeo

## 1.5 Вплив музики на продуктивність під час роботи

Питання впливу музики на ефективність розумової праці давно привертає увагу дослідників у галузі психології та нейронауки. Незважаючи

на певну суперечливість результатів різних досліджень, загальна думка полягає в тому, що правильно підібрана фонова музика здатна позитивно впливати на концентрацію, настрій та загальну продуктивність при виконанні рутинних або повторюваних завдань.

Зокрема, встановлено, що музика з помірним темпом і без вокалу (наприклад, lo-fi, ембієнт або класична музика) найменше відволікає увагу та сприяє так званому стану «потоку» − психологічного занурення у роботу. Натомість музика з текстом або швидким ритмом може заважати при виконанні складних творчих або аналітичних завдань, що вимагають активної роботи з мовою чи логікою (рис. 1.6).

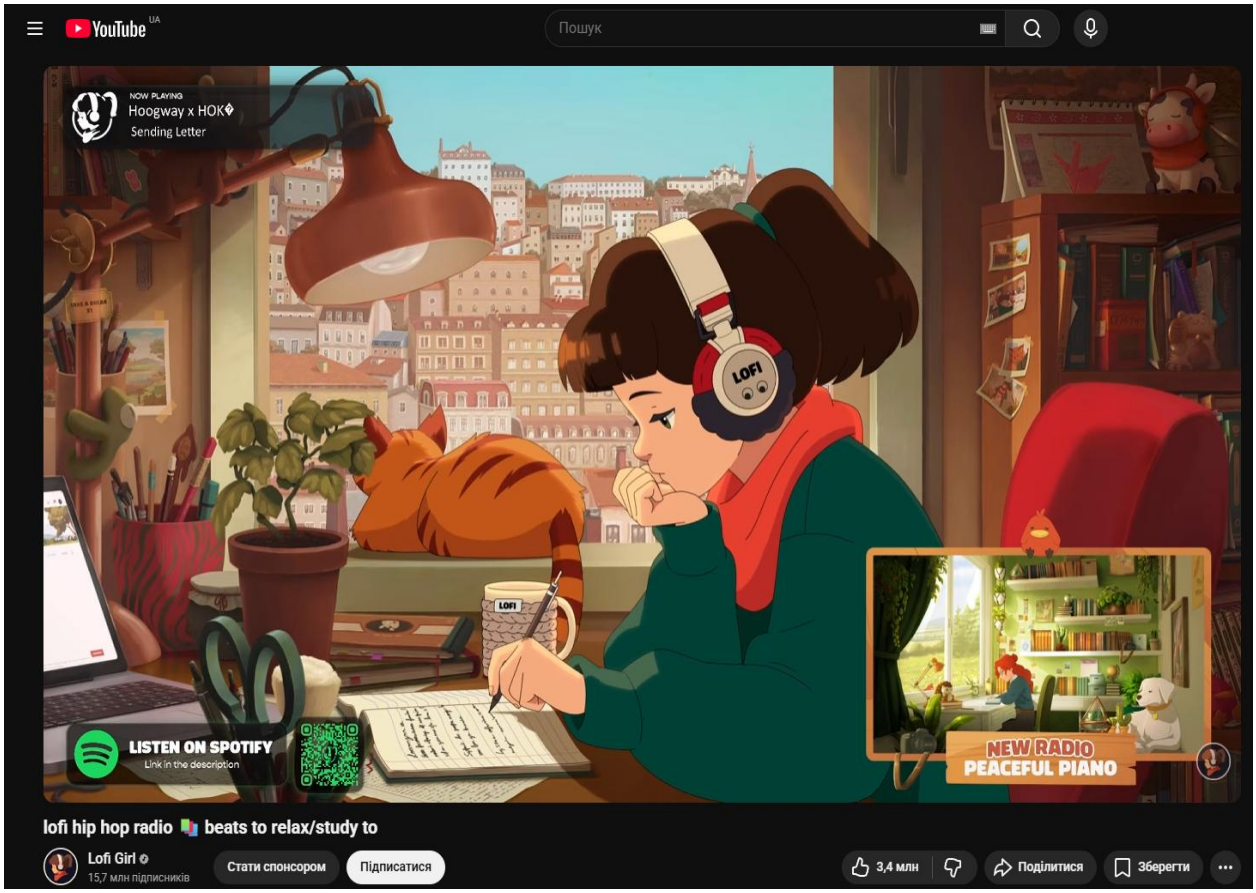


Рисунок 1.6 − Популярна трансляція lo-fi музики з платформи YouTube

Популярний відеохостинг YouTube на сьогоднішній день є однією з найбільших платформ для прослуховування фонової музики під час роботи. Десятки тисяч плейлистів і стрімів у категоріях «lofi hip hop», «study music»,

«ambient work» та подібних збирають мільйони переглядів щодоби. Це свідчить про те, що значна частина людей вже використовує YouTube як джерело фонової музики у своєму робочому процесі. Інтеграція можливості автоматичного відтворення YouTube-посилання безпосередньо в застосунок для контролю режиму роботи є логічним і зручним рішенням, що усуває необхідність переключатися між різними програмами.

## 1.6 Постановка задачі

На основі проведеного аналізу предметної області та огляду існуючих рішень можна сформулювати основну задачу даної кваліфікаційної роботи.

Об’єктом роботи є процес дослідження навантаження на користувача для підтримки здорового режиму роботи за комп’ютером з адаптивним плануванням перерв та персоналізованими рекомендаціями.

Метою роботи є розроблення застосунку, що дозволяє користувачам налаштовувати індивідуальний робочий графік, автоматично отримувати нагадування про перерви та кінець робочого часу, а також ознайомлюватись із персоналізованими рекомендаціями щодо збереження здоров’я під час роботи за комп’ютером.

Проведений аналіз показав, що більшість наявних на ринку рішень вирішують лише окремі аспекти даної проблеми − одні зосереджені виключно на нагадуваннях про перерви, інші − лише на захисті зору чи техніці управління часом. При цьому жоден із розглянутих застосунків не пропонує користувачу повноцінного інструменту, що об’єднував би гнучке планування робочого графіка, персоналізовані рекомендації щодо здоров’я, налаштовуванні сповіщення з власним візуальним зображенням та можливість відтворення фонової музики. Важливою вимогою є також простота у використанні − застосунок повинен бути зрозумілим навіть для

людей без технічного досвіду, а його налаштування не повинні займати багато часу.

Для досягнення мети необхідно вирішити такі завдання:

– гнучка система налаштування робочого графіка;

– своєчасне інформування користувача про перерви у зручній та ненав’язливій формі;

– наявність персоналізованих рекомендацій щодо здоров’я;

– інтеграція з зовнішніми сервісами для відтворення фонової музики;

– наявність унікального маскота-персонажа;

– простота у використанні;

– стабільна робота у фоновому режимі без перевантаження системних ресурсів.

# 2 ЗАСОБИ РОЗРОБКИ ТА ТЕХНІЧНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

## 2.1 Вибір мови програмування

Вибір мови програмування є одним із найважливіших рішень на початковому етапі розробки будь-якого програмного продукту [7, 8]. Від цього вибору залежить не лише зручність розробки, але й продуктивність застосунку, зручність його підтримки та можливості масштабування у майбутньому. Для реалізації десктопного застосунку «Chill out...», що функціонує під операційною системою Windows, було розглянуто кілька потенційних кандидатів: C#, Java, Python та C++.

C# − це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft у рамках платформи .NET. Вона поєднує в собі строгу статичну типізацію, розвинену систему збирання сміття та потужну стандартну бібліотеку. Однією з ключових переваг C# є нативна підтримка технологій Windows Desktop, зокрема WPF (Windows Presentation Foundation) − фреймворку для створення графічних інтерфейсів, що підтримує векторну графіку, анімації та декларативну розмітку через XAML. Мова активно підтримується компанією Microsoft, регулярно оновлюється та має надзвичайно велику спільноту розробників [9].

Java є популярною кросплатформною мовою програмування, що працює на JVM (Java Virtual Machine). Незважаючи на широке застосування у веброзробці та корпоративних системах, Java має суттєві обмеження у контексті розробки десктопних застосунків для Windows. Стандартний фреймворк Swing є морально застарілим, а JavaFX, що прийшов йому на зміну, не забезпечує такого рівня інтеграції з операційною системою Windows, як WPF. Крім того, запуск Java-застосунків вимагає встановленої JRE на комп’ютері користувача, що ускладнює дистрибуцію готового продукту.

Python − динамічна мова з високим рівнем читабельності коду та великою кількістю готових бібліотек. Вона часто використовується для швидкого прототипування та автоматизації задач. Проте для розробки повноцінних десктопних застосунків Python має ряд недоліків: відсутність нативної підтримки MVVM, обмежені можливості декларативного UI, повільніша швидкість виконання порівняно з компільованими мовами та значно більший розмір фінального дистрибутиву через необхідність включення інтерпретатора [10].

C++ забезпечує максимальну продуктивність та мінімальне споживання ресурсів. Однак розробка GUI-застосунків на C++ є значно складнішою та трудомісткою − потребує використання сторонніх бібліотек (Qt, wxWidgets), а сам процес розробки вимагає ручного управління пам’яттю та суттєво більших часових витрат. Для завдань даного застосунку такий рівень низькорівневого контролю є надлишковим [11].

Порівняльний аналіз розглянутих мов програмування наведено у таблиці 2.1.

Таблиця 2.1 − Порівняльний аналіз мов програмування для розробки десктопного застосунку

| **Критерій** | **C#** | **Java** | **Python** | **C++** |
|---|---|---|---|---|
| Підтримка Windows Desktop | Відмінна | Обмежена | Часткова | Хороша |
| Швидкість розробки | Висока | Середня | Висока | Низька |
| WPF / XAML підтримка | Так | Ні | Ні | Ні |
| MVVM підтримка | Нативна | Стороння | Стороння | Відсутня |

Продовження таблиці 2.1

| **Критерій** | **C#** | **Java** | **Python** | **C++** |
|---|---|---|---|---|
| Типізація | Статична, сувора | Статична, сувора | Динамічна | Статична, ручна |
| Споживання пам’яті | Низьке | Середнє | Середнє | Дуже низьке |
| Документація та спільнота | Відмінна | Відмінна | Відмінна | Хороша |

Як видно з таблиці 2.1, мова C# має найбільш комплексну підтримку для розробки Windows Desktop застосунків [12]. Саме тому для реалізації застосунку «Chill out...» було обрано мову C# версії 12, що входить до складу платформи .NET 8. Ця комбінація забезпечує баланс між зручністю розробки, продуктивністю та можливостями інтеграції з операційною системою Windows.

## 2.2 Платформа .NET 8 та її можливості

.NET − це відкрита кросплатформна платформа розробки від компанії Microsoft, що включає середовище виконання (CLR − Common Language Runtime), стандартну бібліотеку класів (BCL − Base Class Library) та набір інструментів для компіляції та публікації застосунків [3]. Починаючи з .NET 5, платформа об’єднала в собі всі попередні версії (.NET Framework, .NET Core та Xamarin), перетворившись на єдине уніфіковане рішення для розробки застосунків різних типів.

.NET 8, випущений у листопаді 2023 року, є версією з довгостроковою підтримкою (LTS − Long-Term Support), що гарантує отримання оновлень безпеки та виправлень протягом трьох років. Це робить його надійним

вибором для розробки нових застосунків, які мають перспективу довготривалої підтримки та розвитку.

Серед ключових можливостей .NET 8, що безпосередньо використовуються у розробці застосунку «Chill out...», варто виділити наступні. По-перше, платформа забезпечує підтримку Windows Desktop Workload, що включає WPF, WinForms та .NET MAUI для Windows. По-друге, .NET 8 надає розвинену систему збирання сміття з підтримкою різних режимів роботи, що мінімізує затримки та забезпечує стабільну роботу застосунку у фоновому режимі. По-третє, платформа містить вбудовану підтримку асинхронного програмування через механізм async/await, що використовується при отриманні назв відеороликів з YouTube через API.

Важливою особливістю .NET 8 є можливість публікації застосунку у форматі Single File − єдиного самодостатнього виконуваного файлу, що не вимагає попередньої встановлення середовища виконання на комп'ютері користувача. Це суттєво спрощує дистрибуцію готового продукту та відповідає вимозі, сформульованій у постановці задачі, − застосунок повинен бути готовий до використання без встановлення додаткового програмного забезпечення.

На рисунку 2.1 зображено структуру файлу проєкту (.csproj), де визначено цільову платформу net8.0-windows та підключено робочі навантаження WPF і Windows Forms.

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net8.0-windows</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <UseWPF>true</UseWPF>
    <UseWindowsForms>true</UseWindowsForms>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.Web.WebView2" Version="1.0.3912.50" />
    <PackageReference Include="Newtonsoft.Json" Version="13.0.4" />
  </ItemGroup>

  <ItemGroup>
    <Folder Include="Assets\Images\" />
  </ItemGroup>

</Project>
```

Рисунок 2.1 − Файл конфігурації проєкту SlothWork.csproj

## 2.3 WPF як фреймворк для графічного інтерфейсу

WPF (Windows Presentation Foundation) − це фреймворк для розробки графічних інтерфейсів десктопних застосунків під Windows, що входить до складу .NET. Він був представлений компанією Microsoft у 2006 році разом з .NET Framework 3.0 і з того часу залишається основним інструментом для розробки багатих GUI-застосунків на платформі Windows [28].

Принципова відмінність WPF від попереднього фреймворку WinForms полягає у використанні DirectX для рендерингу графіки [5]. Це означає, що весь інтерфейс WPF-застосунку відмальовується засобами графічного прискорювача, що забезпечує плавні анімації, підтримку векторної графіки та можливість масштабування без втрати якості. WinForms, навпаки, використовує GDI+ − старіший растровий рендерер, що не підтримує апаратне прискорення та погано масштабується при різних значеннях DPI екрану.

Ключовою концепцією WPF є декларативна розмітка інтерфейсу через XAML (Extensible Application Markup Language) − діалект XML, що описує структуру та зовнішній вигляд UI-елементів. XAML дозволяє чітко розділити логіку відображення (View) від логіки застосунку (ViewModel та Model), що є основою архітектурного патерну MVVM. У застосунку «Chill out...» всі сторінки та вікна описані через XAML-файли: MainWindow.xaml, SchedulePage.xaml, TipsPage.xaml, MusicPage.xaml та ToastWindow.xaml.

Система стилів та шаблонів WPF є надзвичайно гнучкою. Через об’єкти Style можна централізовано визначити зовнішній вигляд будь-якого елемента керування у файлі App.xaml, і ці стилі автоматично застосовуються до всіх елементів відповідного типу в застосунку. У розробленому застосунку таким чином визначено стилі NavButton (кнопки навігаційної панелі), DayButton (кнопки вибору днів тижня), TimeBox (поля введення часу) та PosButton (кнопки вибору позиції сповіщення).

Для порівняння основних фреймворків для розробки Windows Desktop застосунків на платформі .NET складено таблицю 2.2.

Таблиця 2.2 − Порівняльний аналіз UI-фреймворків для .NET

| **Критерій** | **WPF** | **WinForms** | **.NET MAUI** | **Avalonia** |
|---|---|---|---|---|
| Платформа | Windows | Windows | Кросплатформна | Кросплатформна |
| MVVM підтримка | Нативна | Обмежена | Хороша | Хороша |
| Векторна графіка | Так | Ні | Частково | Так |
| Анімації | Розвинені | Обмежені | Середні | Середні |
| Стилізація через XAML | Повна | Відсутня | Повна | Повна |
| Зрілість технології | Висока (2006) | Висока (1998) | Середня (2022) | Середня (2018) |
| WebView2 підтримка | Так | Так | Обмежена | Обмежена |

Аналіз таблиці 2.2 підтверджує, що WPF є оптимальним вибором для застосунку «Chill out...». Незважаючи на те, що .NET MAUI та Avalonia надають кросплатформну підтримку, для даного застосунку, орієнтованого виключно на Windows, кросплатформність не є пріоритетом. Натомість зрілість технології WPF, розвинена екосистема та нативна підтримка MVVM роблять її найбільш доцільним вибором [28].

На рисунку 2.2 зображено загальний вигляд головного вікна застосунку «Chill out...», де відображається бокова навігаційна панель та основна область вмісту, реалізовані засобами WPF.

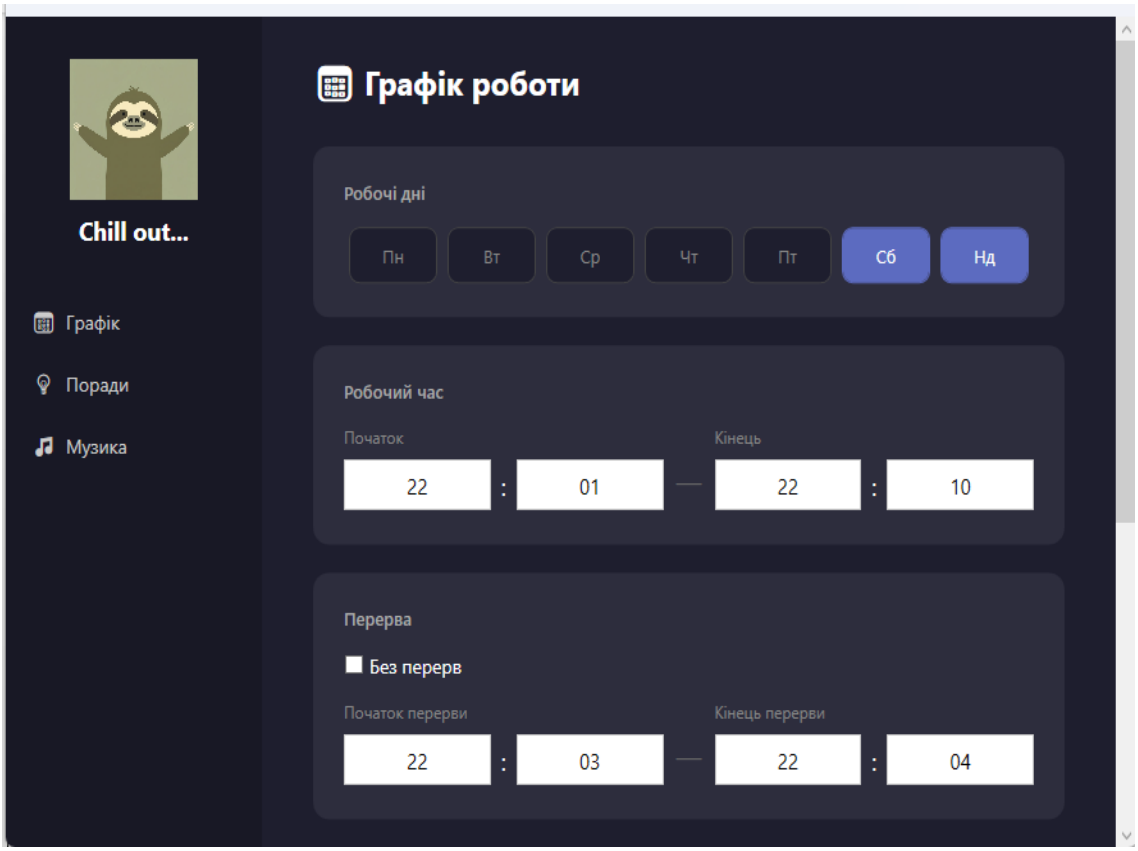


Рисунок 2.2 – Головне вікно застосунку «Chill out...»

## 2.4 Використання маскотів у програмних продуктах

Маскот – це персонаж або образ, що є символом продукту, бренду або організації. У сфері програмного забезпечення маскоти використовуються для створення більш дружнього та людяного відчуття від взаємодії з продуктом. Вони допомагають зробити інтерфейс менш холодним і технічним, що особливо важливо для застосунків, розрахованих на широку аудиторію.

Яскравим прикладом вдалого використання маскота у застосунку є EyeLeo, згаданий раніше, – де леопард демонструє вправи для очей та «спілкується» з користувачем у ненав’язливій формі. Подібний підхід значно підвищує лояльність користувачів та сприяє формуванню емоційного зв’язку з продуктом. Людина, що бачить симпатичного персонажа замість сухого

системного сповіщення, з більшою ймовірністю виконає рекомендацію − в даному випадку зробить перерву.

У рамках даної кваліфікаційної роботи як маскот застосунку обрано лінивця − тварину, що в культурному сприйнятті асоціюється з відпочинком, повільністю та спокоєм. Цей образ ідеально відповідає основній ідеї застосунку − нагадувати користувачу про необхідність сповільнитись, відпочити та подбати про своє здоров’я. Лінивець зображений у мінімалістичному стилі з чіткими лініями та витонченими текстурами, що забезпечує його органічне вписування в сучасний інтерфейс застосунку.

Маскот буде присутній у спливаючих сповіщеннях, що з’являються при настанні часу перерви або завершенні робочого дня. Нижче наведено приклад того, як маскот-персонаж може виглядати у застосунку (рис. 2.3).

Рисунок 2.3 − Приклад маскота-лінивця у мінімалістичному стилі

## 2.5 Архітектурний патерн MVVM

MVVM (Model-View-ViewModel) − це архітектурний патерн проєктування програмного забезпечення, що забезпечує чіткий розподіл відповідальності між компонентами застосунку [13]. Він був розроблений компанією Microsoft спеціально для роботи з технологіями XAML-базованих фреймворків − WPF, Silverlight та пізніше UWP і WinUI. MVVM є природнім розвитком патерну MVP (Model-View-Presenter) і базується на концепції прив'язки даних (Data Binding).

Патерн складається з трьох компонентів. Model (Модель) − це шар даних та бізнес-логіки, що не залежить від UI. У застосунку «Chill out...» до рівня Model відносяться класи AppSettings (налаштування застосунку), MusicTrack (трек музичного списку), TipItem та TipSection (моделі порад), а також WorkSession (сесія роботи). View (Представлення) − це шар відображення, реалізований через XAML-файли. View не містить логіки, лише описує структуру та зовнішній вигляд інтерфейсу. ViewModel (Модель представлення) − це проміжний шар, що містить логіку презентації, трансформує дані з Model у формат, зручний для View, та обробляє команди від користувача через механізм ICommand [14].

Ключовою перевагою MVVM є те, що View і ViewModel спілкуються виключно через механізм прив'язки даних (Binding) та інтерфейс INotifyPropertyChanged [15]. Це означає, що View не має прямих посилань на ViewModel і не знає про деталі реалізації логіки. Така архітектура суттєво спрощує тестування, оскільки ViewModel можна тестувати у повній ізоляції від UI.

У застосунку «Chill out...» базова інфраструктура MVVM реалізована через клас BaseViewModel, що реалізує інтерфейс INotifyPropertyChanged та надає метод SetProperty для зручного оновлення властивостей з автоматичним сповіщенням про зміни. Команди реалізовані через клас RelayCommand, що реалізує інтерфейс ICommand та дозволяє пов'язувати

будь-який делегат Action з кнопками та іншими елементами керування через XAML-прив’язку.

Сервісний шар застосунку відокремлений від ViewModel та включає: SettingsService − читання та запис налаштувань у JSON-файл, TimerService − логіка відстеження робочого часу та перерв, NotificationService − відображення спливаючих сповіщень. Кожен сервіс має чітко визначену відповідальність та не залежить від інших сервісів, що відповідає принципу єдиної відповідальності (SRP) [16].

На рисунку 2.4 показано структуру папок проєкту у Solution Explorer Visual Studio 2022, де відображається розподіл файлів між шарами Models, ViewModels, Views та Services відповідно до архітектурного патерну MVVM.

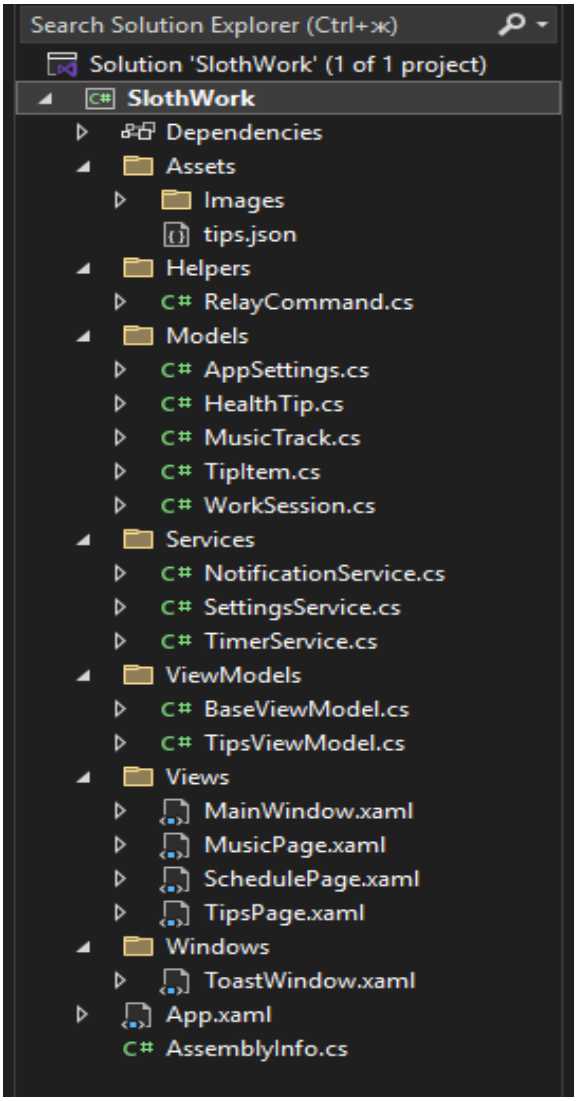


Рисунок 2.4 − Структура проєкту SlothWork у Solution Explorer

## 2.6 Збереження налаштувань у форматі JSON

Для збереження користувацьких налаштувань застосунку «Chill out...» між сесіями роботи було обрано формат JSON (JavaScript Object Notation) − легковаговий текстовий формат обміну даними, заснований на підмножині синтаксису JavaScript [4]. JSON є людиночитабельним, добре підтримується у .NET через бібліотеку Newtonsoft.Json та є де-факто стандартом для зберігання конфігураційних файлів у сучасних застосунках.

Альтернативними варіантами зберігання налаштувань могли бути XML-файли, реєстр Windows, SQLite база даних або вбудований механізм ApplicationSettings у .NET. XML є більш громіздким форматом з надлишковим синтаксисом, що ускладнює ручне редагування файлу налаштувань. Реєстр Windows є системним сховищем, що не рекомендується для зберігання налаштувань сторонніх застосунків через ризики несумісності та складність резервного копіювання. SQLite є реляційною базою даних, що значно потужніша за потреби простого збереження налаштувань − її використання для цієї мети було б надлишком. ApplicationSettings у .NET зберігає дані у форматі XML у системній папці користувача та має обмежену гнучкість при роботі зі складними об’єктними структурами.

Порівняльний аналіз методів збереження даних наведено у таблиці 2.3.

Таблиця 2.3 − Порівняльний аналіз методів збереження налаштувань

| **Критерій** | **JSON** | **SQLite** | **XML** |
|---|---|---|---|
| Читабельність | Висока | Низька | Середня |
| Простота реалізації | Висока | Середня | Середня |
| Підтримка складних запитів | Ні | Так (SQL) | Ні |
| Розмір файлу | Малий | Середній | Великий |
| Залежності | Newtonsoft.Json | Microsoft.Data.Sqlite | Вбудований |

Продовження таблиці 2.3

| Критерій | JSON | SQLite | XML |
|---|---|---|---|
| Налаштування | Ідеальні | Надлишкові | Прийнятні |

Клас AppSettings містить всі налаштування застосунку у вигляді властивостей: список робочих днів (WorkDays), час початку та кінця роботи (WorkStartHour, WorkStartMinute, WorkEndHour, WorkEndMinute), параметри перерви, налаштування нагадувань про очі та шию, список музичних треків (MusicTracks), налаштування розташування сповіщень (NotificationPosition) та кастомні тексти і зображення для кожного типу сповіщень.

На рисунку 2.5 показано вміст файлу settings.json, що автоматично створюється застосунком після першого збереження налаштувань користувачем.


```
{
  "WorkDays": [
    6,
    0
  ],
  "WorkStartHour": 22,
  "WorkStartMinute": 1,
  "WorkEndHour": 22,
  "WorkEndMinute": 10,
  "BreakStartHour": 22,
  "BreakStartMinute": 3,
  "BreakEndHour": 22,
  "BreakEndMinute": 4,
  "EyesReminderEnabled": true,
  "EyesReminderValue": 1,
  "EyesReminderUnit": "Хвилин",
  "NeckReminderEnabled": true,
  "NeckReminderValue": 1,
  "NeckReminderUnit": "Хвилин",
  "NotificationPosition": "MiddleCenter",
  "CustomMascotImagePath": "",
  "YouTubeUrl": "",
  "AutoPlayMusic": false,
  "IsMusicEnabled": false,
  "NoBreak": false,
  "MusicTracks": [
    {
      "Title": "asian lofi radio ⛩ beats to relax/study to",
      "Url": "https://www.youtube.com/watch?v=Na0w3Mz46GA"
    },
    {
      "Title": "Study With Me 🛍 Pomodoro",
      "Url": "https://www.youtube.com/watch?v=1oDrJba2PSs"
    }
  ],
  "SelectedTrackIndex": -1,
```


Рисунок 2.5 − Вміст файлу settings.json зі збереженими налаштуваннями

Серіалізація та десеріалізація об’єкта AppSettings виконується через бібліотеку Newtonsoft.Json за допомогою методів JsonConvert.SerializeObject та JsonConvert.DeserializeObject. При десеріалізації використовується параметр ObjectCreationHandling.Replace, що забезпечує повну заміну колекцій (зокрема списку WorkDays) замість їх доповнення, що вирішує проблему дублювання збережених даних при повторних запусках застосунку.

## 2.7 Компонент WebView2 для інтеграції з YouTube

WebView2 − це компонент Microsoft, що вбудовує повноцінний браузер на основі рушія Chromium (того самого, що використовується в Google Chrome та Microsoft Edge) безпосередньо у вікно WPF-застосунку. Він доступний як NuGet-пакет Microsoft.Web.WebView2 та підтримує всі сучасні вебстандарти: HTML5, CSS3, JavaScript ES2022 та інші [6].

У застосунку «Chill out...» WebView2 використовується для відкриття YouTube-посилань на музику безпосередньо через браузер операційної системи. При натисканні кнопки «Відкрити» поряд із треком у списку музики, застосунок запускає процес браузера Chrome із заданим URL через клас Process з простору імен System.Diagnostics. Це дозволяє уникнути необхідності завантаження та обробки відеопотоку безпосередньо у застосунку, делегуючи цю задачу встановленому браузеру.

Для отримання назви відеоролика з YouTube при додаванні нового треку до списку використовується YouTube oEmbed API − безкоштовний публічний ендпоінт, що не потребує використання API-ключа та повертає метадані відео у форматі JSON за посиланням виду https://www.youtube.com/oembed?url={url}&format=json. HTTP-запит до цього API виконується асинхронно через клас HttpClient з використанням механізму async/await, що забезпечує неблокуючу роботу UI під час очікування відповіді від сервера.

## 2.8 Система спливаючих сповіщень

Спливаючі сповіщення є ключовим функціональним елементом застосунку «Chill out...», оскільки саме вони є основним засобом взаємодії з користувачем під час роботи. Сповіщення реалізовані як окреме вікно WPF (ToastWindow), що відображається поверх усіх інших вікон завдяки властивості Topmost = true, та не відображається на панелі задач (ShowInTaskbar = false).

Вікно сповіщення має прозорий фон (AllowsTransparency = true, Background = Transparent) та кастомну форму з округленими кутами, реалізовану через елемент Border з властивістю CornerRadius. Така реалізація забезпечує сучасний естетичний вигляд сповіщення без стандартних елементів вікна Windows (заголовок, рамка, кнопки мінімізації та закриття).

Кожне сповіщення містить зображення маскота-лінивця та текстовий напис. Зображення завантажується з папки Assets/Images застосунку та відображається через елемент Image з властивістю Stretch = Uniform, що забезпечує повне відображення зображення без обрізання незалежно від його пропорцій. Закриття сповіщення відбувається виключно при кліку мишею в будь-якій точці вікна через обробник події MouseLeftButtonDown.

Позиціонування сповіщення на екрані визначається налаштуванням NotificationPosition з дев’яти можливих варіантів, розташованих у вигляді сітки 3×3 у вкладці «Графік роботи». Координати вікна сповіщення розраховуються на основі розмірів робочої області екрану (SystemParameters.WorkArea) при завантаженні вікна.

Логіка визначення часу показу сповіщень реалізована у класі TimerService, що використовує об’єкт DispatcherTimer для регулярної перевірки поточного часу з інтервалом 1 секунди. Таймер порівнює поточний час із налаштованими часовими мітками (початок роботи, початок перерви, кінець перерви, кінець роботи) та генерує відповідні події, на які підписується MainWindow та делегує їх NotificationService.

Важливою особливістю реалізації є інтелектуальна ініціалізація прапорців стану при запуску застосунку. Метод InitializeFlagsBasedOnCurrentTime аналізує поточний час та визначає, яке сповіщення показати при запуску: якщо застосунок запущено під час робочого часу − показується лише сповіщення про початок робочого дня; якщо під час перерви − сповіщення про початок перерви; якщо після закінчення робочого дня − сповіщення про кінець роботи. Це запобігає ситуації, коли кілька застарілих сповіщень з’являються одночасно після запуску застосунку.

## 2.9 Вкладка налаштування графіку роботи

Вкладка «Графік роботи» є центральним елементом налаштування застосунку та містить усі параметри, що визначають поведінку системи сповіщень. Вона складається з кількох функціональних блоків, кожен з яких відповідає за певний аспект роботи застосунку.

Перший блок − вибір робочих днів тижня. Він реалізований у вигляді семи кнопок з перемиканням (ToggleButton), розташованих в одному рядку через елемент UniformGrid. Активні дні тижня виділяються кольором (фіолетовий фон #5C6BC0), неактивні залишаються темними. При збереженні налаштувань, обрані дні зберігаються у вигляді списку об’єктів DayOfWeek у файлі settings.json.

Другий блок − налаштування робочого часу та часу перерви. Для введення годин та хвилин використовуються поля TextBox зі стилем TimeBox (білий фон, чорний текст, центрування), що дозволяє користувачу вводити час вручну у форматі «ГГ:ХХ». При збереженні виконується валідація введених значень − перевіряється, що години знаходяться в діапазоні 0−23, хвилини в діапазоні 0−59, кінець робочого дня пізніше початку, а час перерви знаходиться в межах робочого часу. Блок перерви

може бути деактивований чекбоксом «Без перерв», що блокує поля введення часу перерви і відповідні сповіщення.

Третій блок − додаткові нагадування для очей та шиї. Кожне нагадування має власний чекбокс для активації та поля для налаштування інтервалу (числове значення та вибір одиниці: хвилини або години). При деактивованому чекбоксі поля налаштування блокуються та відображаються з зменшеною прозорістю (Opacity = 0.3). Інтелектуальний алгоритм чергування нагадувань у TimerService гарантує, що при однаковому інтервалі для обох типів нагадувань вони будуть чергуватися, а не з’являтися одночасно.

Додаткові налаштування (вибір позиції сповіщення та кастомізація текстів і зображень) приховані за замовчуванням та розгортаються при натисканні на текст «Додаткові налаштування ∨». Це дозволяє не перевантажувати інтерфейс другорядними опціями та забезпечує зручність для користувачів, яким не потрібна детальна кастомізація.

## 2.10 Вкладка порад щодо здоров’я

Вкладка «Поради» реалізована у вигляді двопанельного макету: ліва панель містить список тем, права − розгорнутий вміст обраної теми. Такий підхід є стандартним патерном проєктування «Master-Detail» та забезпечує зручну навігацію по великому обсягу текстового контенту.

Особливістю реалізації вкладки є винесення текстового вмісту порад у зовнішній файл Assets/tips.json. Цей підхід відокремлює вміст від коду застосунку, що суттєво спрощує редагування та додавання нових порад у майбутньому − для цього не потрібно перекомпілювати застосунок, достатньо відредагувати JSON-файл. Структура файлу включає для кожної поради заголовок, підзаголовок, список секцій (кожна секція має заголовок та текст) та ім’я зображення.

Застосунок містить чотири розділи порад, що охоплюють найважливіші аспекти здорової роботи за комп’ютером: напруження очей та профілактика зорової втоми, важливість розминки шиї та покращення мозкового кровообігу, ергономіка робочого місця, а також вплив музики на продуктивність розумової праці. Кожен розділ супроводжується ілюстрацією з маскотом застосунку, що підтримує єдиний візуальний стиль продукту.

## 2.11 Visual Studio 2022 як середовище розробки

Visual Studio 2022 Community Edition − це безкоштовна, повнофункціональна інтегрована середа розробки (IDE) від компанії Microsoft, що є стандартним інструментом для розробки застосунків на платформі .NET [17]. Вона надає повний набір засобів для написання, налагодження, тестування та публікації застосунків.

Для розробки застосунку «Chill out...» використовувався робочий набір «.NET desktop development», що включає підтримку WPF, WinForms та MAUI для Windows. Цей робочий набір встановлюється при першому запуску Visual Studio через Visual Studio Installer та надає всі необхідні шаблони проєктів, компілятори та бібліотеки.

Серед ключових інструментів Visual Studio 2022, що активно використовувалися при розробці застосунку, варто відзначити наступні. XAML Designer − візуальний редактор інтерфейсу, що дозволяє переглядати XAML-розмітку в режимі реального часу та вносити зміни через графічний інтерфейс. IntelliSense − система автодоповнення коду, що суттєво прискорює написання коду та зменшує кількість помилок. Debugger − потужний відладчик з підтримкою точок зупинки, перегляду значень змінних та покрокового виконання коду. NuGet Package Manager − менеджер пакетів, що дозволяє встановлювати та оновлювати сторонні бібліотеки. Solution

Explorer − провідник структури проєкту, що відображає ієрархію файлів та папок.

Менеджер NuGet-пакетів використовувався для встановлення наступних залежностей проєкту: Newtonsoft.Json (версія 13.x) − для серіалізації та десеріалізації JSON; Microsoft.Web.WebView2 − для вбудовування браузерного компонента у WPF-застосунок. Всі залежності автоматично записуються у файл SlothWork.csproj у секцію ItemGroup з тегами PackageReference.

На рисунку 2.6 показано інтерфейс Visual Studio 2022 під час розробки застосунку «Chill out...» − відкритий файл SchedulePage.xaml у режимі розбитого перегляду (XAML-розмітка та XAML Designer одночасно).

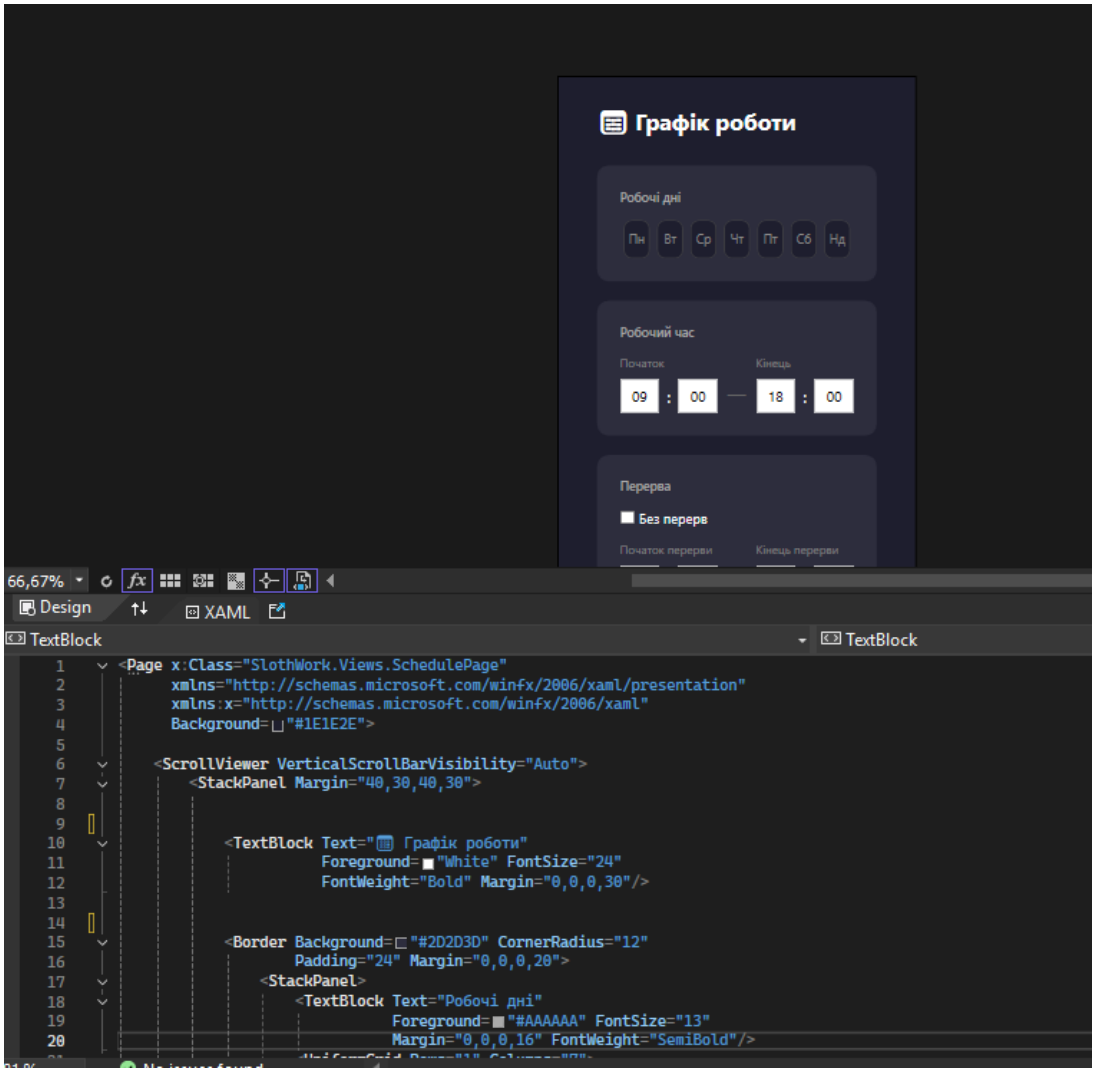


Рисунок 2.6 − Середовище розробки Visual Studio 2022 з відкритим файлом SchedulePage.xaml

## 2.12 Публікація та дистрибуція застосунку

Після завершення розробки застосунок «Chill out...» публікується у форматі Release збірки для забезпечення максимальної продуктивності виконуваного коду. Компілятор .NET у режимі Release застосовує оптимізації коду, видаляє налагоджувальну інформацію та генерує більш ефективний машинний код порівняно з режимом Debug.

Для публікації застосунку у форматі єдиного самодостатнього файлу (.exe) використовується команда dotnet publish з параметрами: конфігурація Release (−c Release), цільова платформа Windows x64 (−r win-x64), режим самодостатності (−−self-contained true), публікація в один файл (−p:PublishSingleFile=true) та включення нативних бібліотек (−p:IncludeNativeLibrariesForSelfExtract=true). Результатом є єдиний виконуваний файл SlothWork.exe, що не потребує встановленого .NET Runtime на комп’ютері користувача.

На рисунку 2.7 зображено структуру папки з готовим застосунком після публікації, де видно виконуваний файл SlothWork.exe та супутню папку Assets.

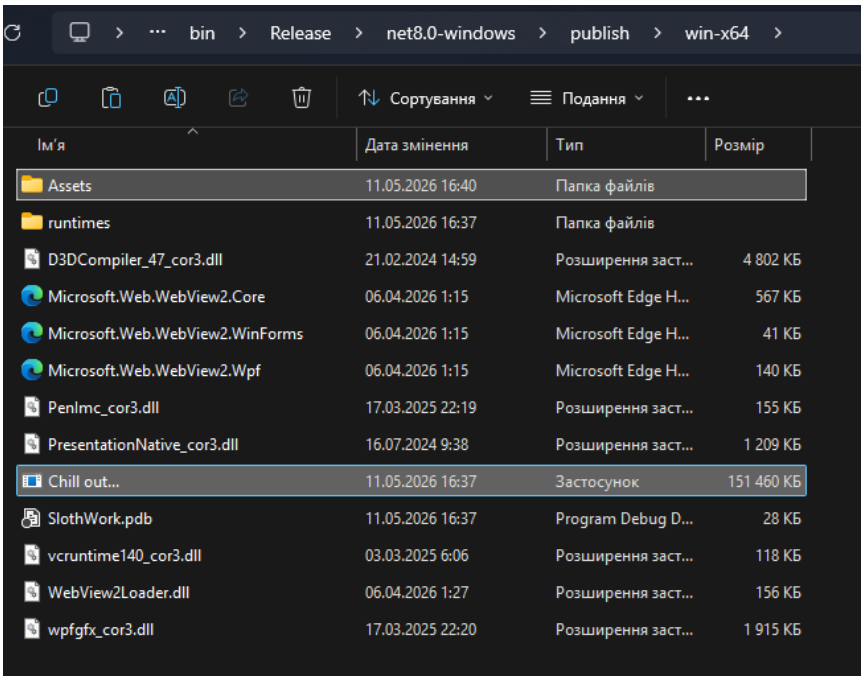


Рисунок 2.7 − Структура папки з готовим застосунком після публікації

Важливою частиною дистрибуції є структура папки Assets, що повинна знаходитись поряд з виконуваним файлом. Ця папка містить підпапку Images з зображеннями маскота (img1.jpg − img8.jpg), що використовуються у сповіщеннях та в інтерфейсі застосунку, а також файл tips.json з текстовим вмістом вкладки «Поради». Для автоматичного копіювання цих файлів у папку збірки при кожній компіляції у Visual Studio встановлено властивість «Copy to Output Directory: Copy if newer» для кожного з цих файлів через Solution Explorer.

Альтернативним способом публікації застосунку є використання графічного інтерфейсу Visual Studio 2022 через пункт меню Publish, що дозволяє виконати ту саму процедуру без необхідності вручну вводити команди у термінал. При цьому розробник обирає цільову папку (Folder) як спосіб публікації, після чого у вікні налаштувань профілю публікації встановлюються параметри Configuration (Release), Target Framework (net8.0-windows), Deployment mode (Self-contained) та Target runtime (win-x64), а також активується опція Produce single file. Такий підхід є більш зручним для розробників, що віддають перевагу візуальним інструментам над командним рядком, та дозволяє наочно переглянути всі доступні параметри публікації перед запуском процесу збирання фінальної версії застосунку.

# 3 РЕАЛІЗАЦІЯ, ТЕМТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

## 3.1 Реалізація постановки задачі

У першому розділі даної роботи була сформульована постановка задачі, що визначила основні вимоги до розроблюваного застосунку. Метою розробки є створення десктопного застосунку для операційної системи Windows, що допомагає користувачу підтримувати здоровий режим роботи за комп'ютером. У цьому підрозділі розглядається відповідність реалізованого застосунку «Chill out...» кожній із сформульованих вимог.

Першою вимогою була гнучка система налаштування робочого графіка. У застосунку вона реалізована через вкладку «Графік роботи», що дозволяє обирати будь-яку комбінацію робочих днів тижня, встановлювати час початку та кінця робочого дня з точністю до хвилини, а також налаштовувати проміжок перерви або повністю відключати перерви через чекбокс «Без перерв». Всі ці налаштування зберігаються у файлі settings.json та відновлюються при кожному запуску застосунку.

Другою вимогою було своєчасне інформування користувача про перерви у зручній та ненав'язливій формі. Реалізована система сповіщень використовує власне вікно WPF без стандартних елементів заголовка Windows, розташоване поверх усіх інших вікон. Сповіщення не блокує роботу користувача − воно лише інформує та закривається при одиночному кліку миші в будь-якій його точці. Позиція сповіщення налаштовується через сітку 3×3 з дев'яти можливих варіантів розміщення на екрані.

Третьою вимогою була наявність персоналізованих рекомендацій щодо здоров'я. Вкладка «Поради» містить чотири тематичні розділи: напруження очей та профілактика зорової втоми, важливість розминки шиї та покращення мозкового кровообігу, ергономіка робочого місця, та вплив музики на продуктивність. Кожен розділ супроводжується ілюстрацією маскота

застосунку та структурованим текстом, що підготовлений на основі рекомендацій фахівців у галузі ергономіки та медицини праці.

Четвертою вимогою була інтеграція з зовнішніми сервісами для відтворення фонової музики. Вкладка «Музика» дозволяє зберігати список YouTube-посилань з автоматичним отриманням назви відеоролика через публічний oEmbed API, відкривати обраний трек у браузері одним кліком та налаштовувати автоматичний запуск обраного треку на початку робочого дня. Застосунок не намагається самостійно програвати відео, а делегує цю задачу встановленому браузеру, що є більш надійним та сумісним підходом.

П’ятою вимогою була наявність унікального маскота-персонажа. Лінивець у мінімалістичному стилі присутній у всіх спливаючих сповіщеннях застосунку, а також відображається у верхній частині бічної навігаційної панелі головного вікна. Для кожного з шести типів сповіщень передбачено окреме зображення маскота, що відображає тематику конкретного нагадування. При цьому користувач має можливість замінити стандартне зображення власним для кожного типу сповіщення окремо.

Шостою вимогою була простота у використанні − застосунок повинен бути зрозумілим навіть для людей без технічного досвіду, а налаштування не повинні займати багато часу. Інтерфейс реалізований у темному мінімалістичному стилі з чіткою структурою та зрозумілими підписами. Всі складні налаштування (позиція сповіщення, кастомізація текстів та зображень) приховані за кнопкою «Додаткові налаштування» і не завантажують початковий вигляд вкладки.

Сьомою вимогою була стабільна робота у фоновому режимі без перевантаження системних ресурсів. TimerService використовує легковаговий DispatcherTimer з інтервалом перевірки 5 секунд − такий підхід практично не навантажує процесор, оскільки перевірка зводиться до порівняння кількох числових значень. Застосунок не виконує жодних важких фонових операцій, мережевих запитів у циклі чи інтенсивного введення-виведення під час фонової роботи.

## 3.2 Опис інтерфейсу користувача

Головне вікно застосунку «Chill out...» побудоване за двоколонковим макетом. Ліва колонка шириною 200 пікселів є постійною навігаційною панеллю, що присутня незалежно від обраної вкладки. Вона містить зображення маскота-лінивця у верхній частині, назву застосунку «Chill out...» та три кнопки навігації: «Графік», «Поради» та «Музика». Права колонка займає всю решту простору вікна та відображає вміст активної вкладки через елемент Frame з вимкненою навігаційною панеллю (NavigationUIVisibility = Hidden).

Кольорова схема застосунку побудована на темній палітрі: основний фон вікна − #1E1E2E (темно-синій), фон навігаційної панелі − #181825 (ще темніший відтінок), фон карток-блоків на сторінках − #2D2D3D (темно-сірий). Акцентний колір − #5C6BC0 (м’який фіолетовий), що використовується для активних елементів, кнопок збереження та виділених днів тижня. Такий вибір кольорової схеми мінімізує навантаження на очі при тривалому використанні застосунку − що є особливо доречним для продукту, орієнтованого на покращення здоров’я при роботі за комп’ютером.

Вкладка «Графік роботи» організована як вертикальний список карток, кожна з яких відповідає певному блоку налаштувань. Перша картка − вибір робочих днів через кнопки з перемиканням. Друга − налаштування робочого часу з полями для введення годин та хвилин. Третя − налаштування перерви з чекбоксом «Без перерв». Четверта − додаткові нагадування для очей та шиї з налаштуванням інтервалів. Далі розташована кнопка «Додаткові налаштування», що розгортає блоки вибору позиції сповіщення та кастомізації текстів і зображень. Внизу − кнопка «Зберегти графік» та рядок підтвердження збереження.

Вкладка «Поради» реалізована у вигляді двопанельного макету з патерном «Master-Detail». Ліва панель шириною 220 пікселів містить список із чотирьох тем, оформлений у вигляді навігаційного списку з виділенням

активного пункту фіолетовим кольором. Права панель відображає повний текст обраної поради, структурований за підрозділами з синіми заголовками (#90CAF9) та сірим текстом (#DDDDDD), а внизу − зображення маскота для відповідної теми.

Вкладка «Музика» містить три функціональні блоки. Перший − поле для введення YouTube-посилання з кнопкою «Зберегти» та рядком статусу операції. Другий − блок автозапуску з чекбоксом та випадаючим списком для вибору треку. Третій − список збережених треків з кнопками «Відкрити» та «🗑» для кожного треку.

На рисунку 3.1 наведено загальний вигляд застосунку з відкритою вкладкою «Графік роботи» та заповненими налаштуваннями.

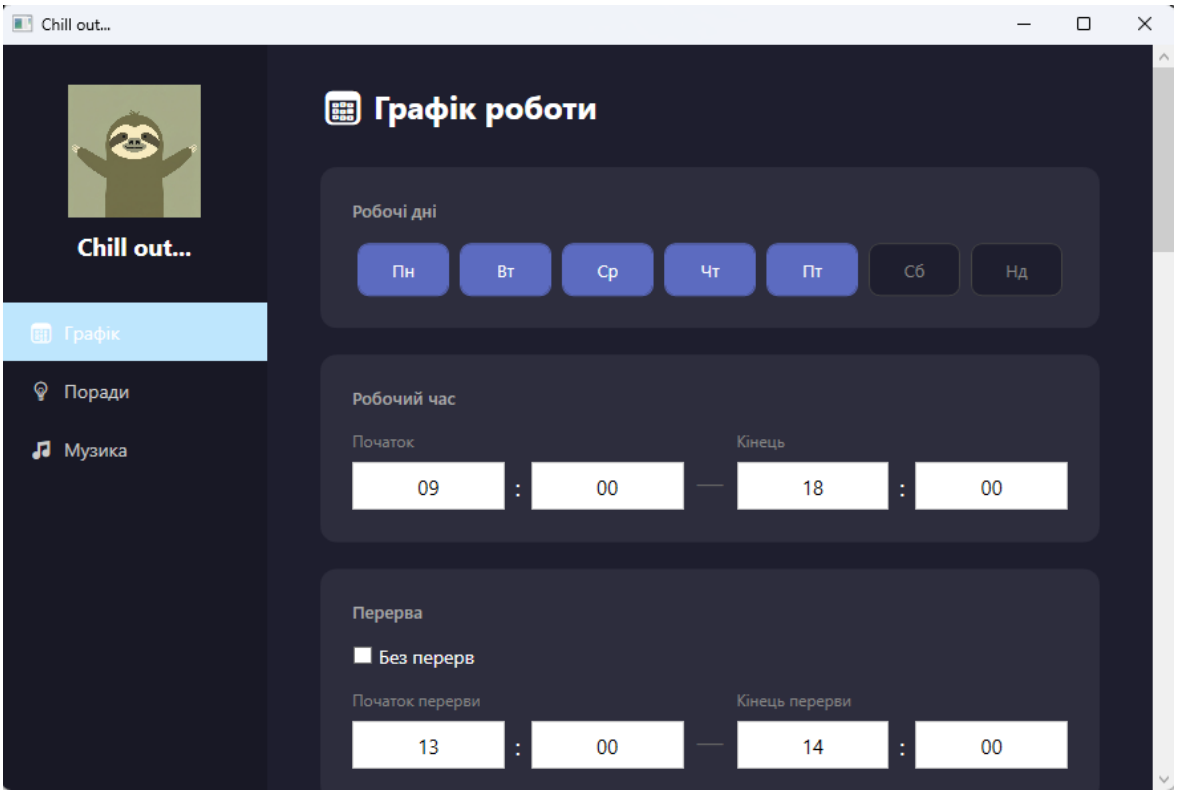


Рисунок 3.1 − Загальний вигляд застосунку «Chill out...»

На рисунку 3.2 зображено розгорнуті додаткові налаштування вкладки «Графік роботи» − сітка вибору позиції сповіщення та блок кастомізації текстів і зображень.

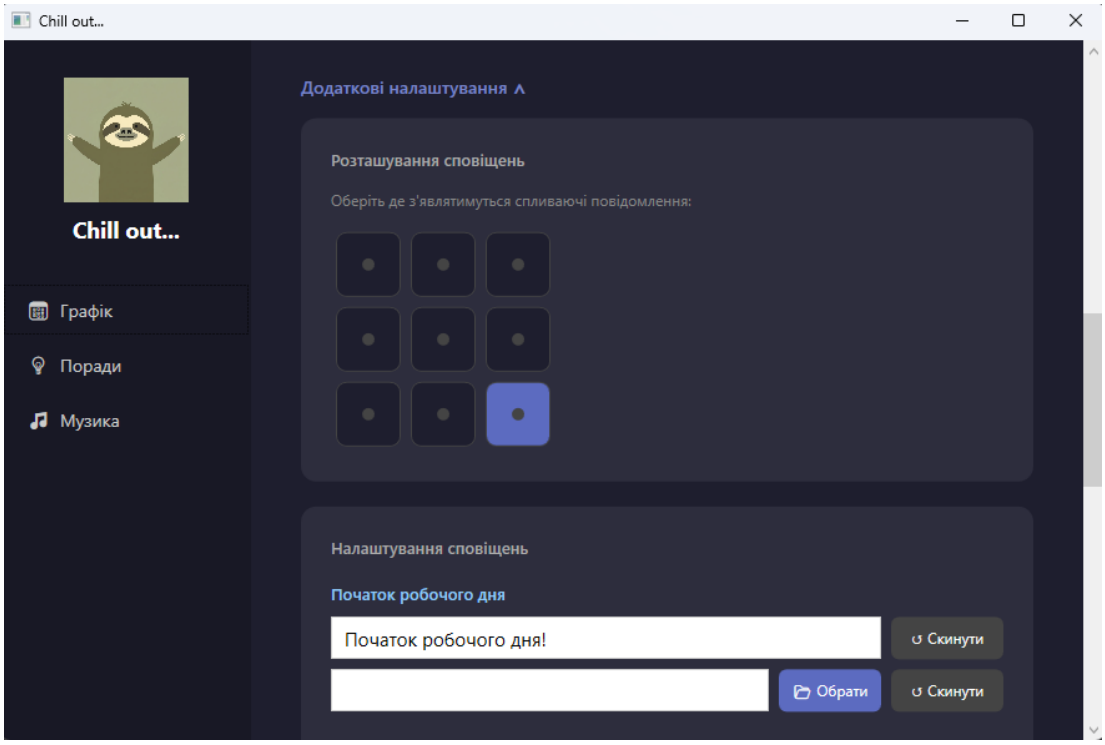


Рисунок 3.2 – Додаткові налаштування сповіщень у вкладці «Графік роботи»

На рисунку 3.3 зображено вкладку «Поради» з відкритим розділом про ергономіку робочого місця.

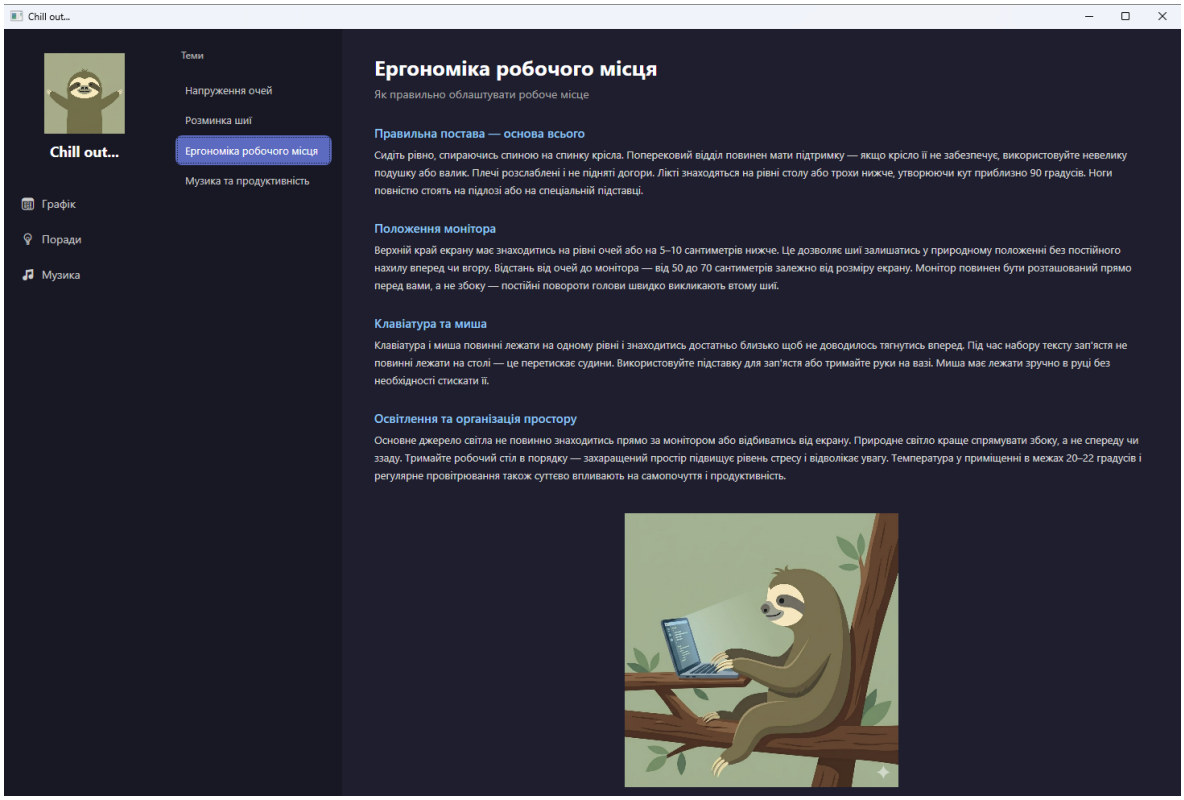


Рисунок 3.3 – Вкладка «Поради» з розділом про ергономіку

На рисунку 3.4 зображено вкладку «Музика» зі збереженими треками у списку.

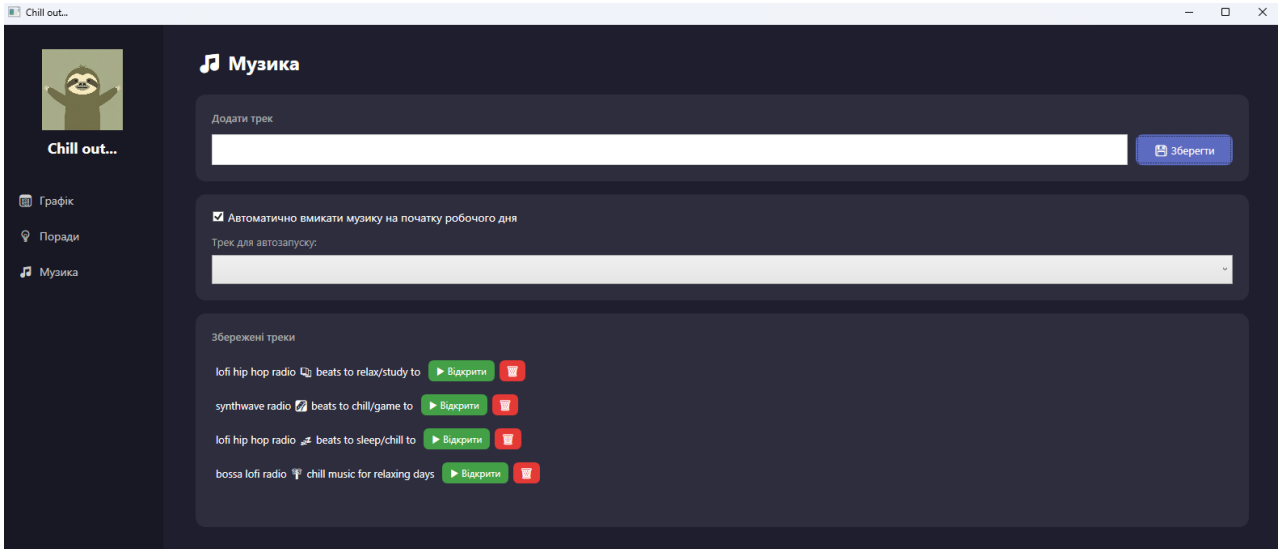


Рисунок 3.4 – Вкладка «Музика» зі збереженим списком треків

## 3.3 Реалізація алгоритму відстеження часу

Алгоритм відстеження часу є серцевиною застосунку «Chill out...» і реалізований у класі TimerService. Розглянемо детально логіку його роботи та особливості реалізації, що забезпечують коректну поведінку у всіх можливих сценаріях використання.

Таймер перевіряє поточний час кожну секунду через об’єкт DispatcherTimer. При кожному спрацюванні алгоритм виконує перевірки у суворо визначеному порядку. Спочатку перевіряється, чи є поточний день тижня робочим згідно з налаштуваннями – якщо ні, решта перевірок пропускається. Далі визначаються часові мітки: час початку та кінця роботи, початку та кінця перерви. Після цього розраховуються два булеві стани: isBreakTime (чи зараз перерва) та isWorkTime (чи зараз робочий час без перерви).

Для запобігання повторному спрацюванню кожного типу сповіщення використовується система булевих прапорців: _workStartNotified, _breakStartNotified, _breakEndNotified та _workEndNotified. Кожен прапорець встановлюється в true після першого спрацювання відповідного сповіщення та залишається таким до скидання налаштувань або перезапуску застосунку. Це гарантує, що кожне сповіщення з’явиться рівно один раз за робочий день.

Нагадування про очі та шию відстежуються окремо через поля _lastEyesReminder та _lastNeckReminder типу DateTime? (nullable). Значення null означає, що відлік ще не розпочато − такий стан встановлюється при ініціалізації та під час перерви. При початку робочого дня або після закінчення перерви обом полям присвоюється поточний час DateTime.Now, що запускає відлік. Нагадування спрацьовує, коли різниця між поточним часом та останнім нагадуванням перевищує налаштований інтервал.

Окремої уваги заслуговує алгоритм чергування нагадувань при однаковому інтервалі для очей та шиї. Коли обидва типи нагадувань увімкнені та мають однаковий інтервал, застосунок використовує булевий прапорець _nextIsNeck для визначення черги. Після кожного спрацювання нагадування обидва лічильники (_lastEyesReminder та _lastNeckReminder) скидаються до поточного часу, а прапорець перемикається. Це забезпечує суворе чергування: перший раз − нагадування про очі, другий − про шию, третій − знову про очі, і так далі.

При різних інтервалах для очей та шиї алгоритм перевіряє обидва незалежно. Якщо обидва готові до показу одночасно, відображається те, яке давніше не показувалось (порівняння _lastEyesReminder та _lastNeckReminder). Після показу одного нагадування другий тип залишається у черзі та показується при наступному спрацюванні таймера.

Важливою частиною алгоритму є метод InitializeFlagsBasedOnCurrentTime, що виконується при кожному запуску або оновленні налаштувань. Він аналізує поточний час і визначає початковий стан прапорців таким чином, щоб при запуску застосунку з’явилося лише

одне найбільш актуальне сповіщення, а всі пропущені події були позначені як вже оброблені. Це вирішує проблему накопичення застарілих сповіщень при запуску застосунку після закінчення робочого дня або в інший неробочий час.

На рисунку 3.5 зображено код методу InitializeFlagsBasedOnCurrentTime у файлі TimerService.cs, що реалізує інтелектуальну ініціалізацію стану таймера.

```
private void InitializeFlagsBasedOnCurrentTime()
{
    var now = DateTime.Now;
    var current = TimeOnly.FromDateTime(now);
    var workStart = new TimeOnly(_settings.WorkStartHour, _settings.WorkStartMinute);
    var workEnd = new TimeOnly(_settings.WorkEndHour, _settings.WorkEndMinute);
    var breakStart = new TimeOnly(_settings.BreakStartHour, _settings.BreakStartMinute);
    var breakEnd = new TimeOnly(_settings.BreakEndHour, _settings.BreakEndMinute);
    bool isBeforeWork = current < workStart;
    bool isDuringWork = current >= workStart && current < workEnd;
    bool isDuringBreak = !_settings.NoBreak && current >= breakStart && current < breakEnd;
    bool isAfterWork = current >= workEnd;

    if (isBeforeWork)
    {
        _workStartNotified = false;
        _breakStartNotified = false;
        _breakEndNotified = false;
        _workEndNotified = false;
        _lastEyesReminder = null;
        _lastNeckReminder = null;
        _nextIsNeck = false;
        return;
    }

    if (isDuringBreak)
    {
        _workStartNotified = true;
        _breakStartNotified = false;
        _breakEndNotified = false;
        _workEndNotified = false;
        _lastEyesReminder = null;
        _lastNeckReminder = null;
        _nextIsNeck = false;
        return;
    }

    if (isDuringWork)
    {
        _workStartNotified = false;
        _breakStartNotified = false;
```

Рисунок 3.5 – Метод ініціалізації стану таймера InitializeFlagsBasedOnCurrentTime

## 3.4 Реалізація системи збереження налаштувань

Система збереження налаштувань застосунку «Chill out...» реалізована через клас SettingsService, що відповідає за читання та запис файлу settings.json. Файл розташовується у робочій директорії застосунку − тобто поряд з виконуваним файлом SlothWork.exe. Такий підхід забезпечує прозорість зберігання даних та дозволяє користувачу за необхідності вручну переглянути або відредагувати файл налаштувань у будь-якому текстовому редакторі.

Клас AppSettings містить 28 властивостей, що охоплюють усі аспекти конфігурації застосунку. Серед них: список робочих днів (List<DayOfWeek>), часові параметри роботи та перерви (вісім цілочисельних полів для годин та хвилин), прапорець відсутності перерв (NoBreak), параметри нагадувань для очей та шиї (чотири поля), список музичних треків (List<MusicTrack>), індекс обраного треку для автозапуску, рядок позиції сповіщення (NotificationPosition) та по два рядки для кожного з шести типів сповіщень − кастомний текст та шлях до кастомного зображення.

При десеріалізації JSON-файлу використовується параметр ObjectCreationHandling.Replace бібліотеки Newtonsoft.Json. Без цього параметру колекції (List<DayOfWeek> та List<MusicTrack>) не замінювалися б збереженими значеннями, а доповнювалися значеннями за замовчуванням, що призводило б до дублювання елементів при кожному запуску застосунку. Параметр Replace забезпечує повну заміну колекції під час десеріалізації, що є коректною поведінкою для даного сценарію.

Збереження налаштувань відбувається при двох подіях: натисканні кнопки «Зберегти графік» у вкладці «Графік роботи» та автоматично при зміні будь-яких налаштувань у вкладці «Музика» (активація автозапуску, зміна обраного треку, додавання або видалення треку). Після збереження у вкладці «Графік роботи» метод ReloadSettings головного вікна оновлює

налаштування в TimerService та NotificationService без перезапуску застосунку.

На рисунку 3.6 зображено код класу SettingsService з методами Load та Save у файлі Services/SettingsService.cs.

```
using Newtonsoft.Json;
using SlothWork.Models;
using System.IO;

namespace SlothWork.Services
{
    6 references
    public class SettingsService
    {
        private readonly string _path = "settings.json";

        4 references
        public AppSettings Load()
        {
            if (!File.Exists(_path))
                return new AppSettings();

            var json = File.ReadAllText(_path);
            return JsonConvert.DeserializeObject<AppSettings>(json, new JsonSerializerSettings
            {
                ObjectCreationHandling = ObjectCreationHandling.Replace
            }) ?? new AppSettings();
        }

        5 references
        public void Save(AppSettings settings)
        {
            var json = JsonConvert.SerializeObject(settings, Newtonsoft.Json.Formatting.Indented);
            File.WriteAllText(_path, json);
        }
    }
}
```

Рисунок 3.6 − Клас SettingsService з методами читання та запису налаштувань

## 3.5 Тестування застосунку

Тестування застосунку «Chill out...» проводилося методом функціонального тестування у ручному режимі. Для кожного функціонального блоку застосунку були визначені тестові сценарії, що перевіряли як стандартні шляхи використання (happy path), так і граничні та нестандартні ситуації (edge cases).

Тестування проводилося на комп’ютері з операційною системою Windows 11 з роздільністю екрану 1920×1080 пікселів. Для перевірки коректності роботи таймера час початку роботи, перерви та кінця робочого

дня встановлювався на значення, близькі до поточного часу, що дозволяло спостерігати спрацювання сповіщень без тривалого очікування. Тестування функції збереження налаштувань проводилося шляхом повного закриття та повторного запуску застосунку після кожного збереження.

Особлива увага при тестуванні приділялася сценаріям запуску застосунку в різний час доби: до початку робочого дня, під час робочого часу, під час перерви та після закінчення робочого дня. Метою цих тестів була перевірка коректності методу InitializeFlagsBasedOnCurrentTime − щоб при кожному сценарії з’являлося рівно одне актуальне сповіщення, а не кілька застарілих одночасно.

Також тестувалася поведінка застосунку при введенні некоректних даних: символів замість цифр у полях часу, значень поза допустимими діапазонами (наприклад, годин більше 23 або хвилин більше 59), логічно некоректних часових комбінацій (кінець роботи раніше початку, перерва поза межами робочого часу). У всіх випадках застосунок коректно відображав зрозуміле повідомлення про помилку та не виконував збереження некоректних даних.

## 3.6 Виявлені проблеми та їх вирішення

У процесі розробки та тестування застосунку «Chill out...» було виявлено ряд технічних проблем, що вимагали окремих рішень. Розглянемо найбільш значущі з них.

Першою проблемою стало дублювання робочих днів у файлі settings.json при повторних запусках застосунку. При завантаженні налаштувань бібліотека Newtonsoft.Json за замовчуванням додавала десеріалізовані елементи до вже ініціалізованого списку WorkDays (що містить п’ять робочих днів за замовчуванням), а не замінювала його. В результаті після кожного перезапуску список ставав удвічі більшим.

Проблема вирішена додаванням параметра ObjectCreationHandling.Replace при десеріалізації, що забезпечує повну заміну колекції.

Другою проблемою було одночасне відображення кількох застарілих сповіщень при запуску застосунку у випадках, коли поточний час вже перевищував кілька налаштованих часових міток. Наприклад, якщо застосунок запускався о 15:00, а початок роботи був встановлений на 09:00, а перерва на 13:00-14:00, то при першому спрацюванні таймера відображалися б три сповіщення одночасно: початок роботи, початок перерви та кінець перерви. Проблема вирішена реалізацією методу InitializeFlagsBasedOnCurrentTime, що визначає початковий стан прапорців на основі поточного часу.

Третьою проблемою було обрізання кастомних зображень у вікні сповіщення при нестандартних пропорціях зображення (наприклад, горизонтальні фотографії). Стандартне значення «Stretch = UniformToFill» масштабує зображення до заповнення контейнера, обрізаючи частини, що виходять за межі. Проблема вирішена зміною значення на «Stretch = Uniform», що масштабує зображення зберігаючи пропорції та при необхідності залишає порожній простір (letterbox) замість обрізання.

Четвертою проблемою стали конфлікти іменування типів через одночасне підключення просторів імен System.Windows та System.Windows.Forms. Зокрема, типи Application, MessageBox, Button та OpenFileDialog існують в обох просторах імен. Проблема вирішена додаванням явних псевдонімів у директивах using для усунення неоднозначності типів: using Application = System.Windows.Application, using MessageBox = System.Windows.MessageBox тощо.

На рисунку 3.7 зображено приклад відображення кастомного горизонтального зображення у сповіщенні після виправлення проблеми з обрізанням (Stretch = Uniform). Як видно з рисунка, зображення масштабується пропорційно до доступного простору контейнера, а вільний простір, заповнюється фоновим кольором вікна сповіщення.

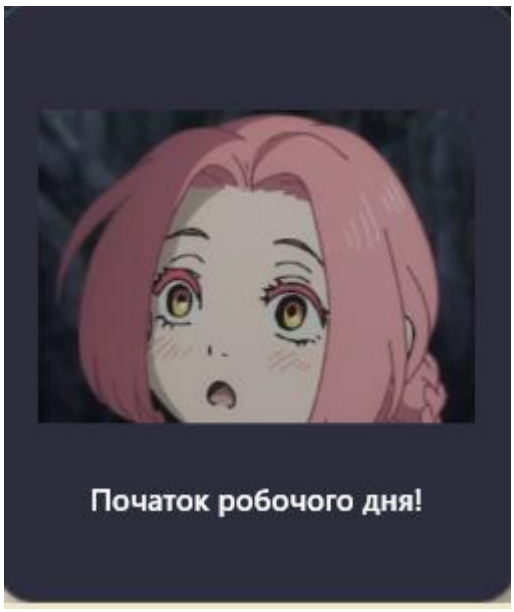


Рисунок 3.7 − Коректне відображення кастомного зображення у сповіщенні

## 3.7 Аналіз отриманих результатів

За результатами розробки та тестування застосунок «Chill out...» є повністю функціональним програмним продуктом, що реалізує всі вимоги, сформульовані у постановці задачі. Проаналізуємо отримані результати з різних точок зору.

З точки зору функціональності застосунок надає комплексне рішення для контролю режиму праці та відпочинку [33]. На відміну від розглянутих у першому розділі аналогів, що закривають лише окремі аспекти проблеми, «Chill out...» об’єднує гнучке планування графіка, шість типів налаштовуваних сповіщень, нагадування про здоров’я очей та шиї, розділ порад щодо ергономіки та інтеграцію з YouTube в єдиному продукті. Порівняно з EyeLeo, що має лише один тип нагадувань, або Stretchly, що не підтримує кастомізацію, застосунок «Chill out...» надає суттєво ширші можливості персоналізації.

З точки зору технічної реалізації обраний стек технологій (C# + .NET 8 + WPF + MVVM) довів свою ефективність. Архітектура MVVM забезпечила

чіткий розподіл відповідальності між компонентами, що спростило розробку та усунення дефектів. Сервісний шар (SettingsService, TimerService, NotificationService) відокремлений від шару відображення та може бути легко розширений у майбутньому без змін у UI.

З точки зору користувацького досвіду застосунок відповідає сучасним стандартам дизайну. Темна кольорова схема знижує навантаження на очі, що особливо доречно для застосунку, спрямованого на покращення здоров'я при роботі за комп'ютером. Чітка структура вкладок, зрозумілі підписи елементів керування та приховані додаткові налаштування забезпечують простоту освоєння для нових користувачів.

З точки зору продуктивності застосунок споживає мінімальну кількість системних ресурсів у фоновому режимі. DispatcherTimer з інтервалом 5 секунд не створює помітного навантаження на процесор. Файл налаштувань settings.json зчитується лише при запуску та після явного збереження, а не при кожному спрацюванні таймера. Зображення маскота завантажуються з диска лише при створенні нового вікна сповіщення.

З точки зору дистрибуції та зручності розгортання застосунок публікується у форматі єдиного самодостатнього файлу SlothWork.exe, що не вимагає встановленого середовища виконання .NET на комп'ютері кінцевого користувача. Єдиною додатковою умовою є наявність папки Assets поряд з виконуваним файлом, що містить зображення маскота та файл tips.json з текстом порад.

## 3.8 Інструкція користувача

Для початку роботи із застосунком «Chill out...» необхідно виконати кілька простих кроків. Перший запуск застосунку здійснюється подвійним кліком на файлі SlothWork.exe. При першому запуску застосунок

відкривається з налаштуваннями за замовчуванням: робочі дні − понеділок−п’ятниця, час роботи − 09:00−18:00, час перерви − 13:00−14:00.

Для налаштування графіку роботи необхідно перейти на вкладку «Графік» та виконати наступні дії. У блоці «Робочі дні» обрати потрібні дні тижня натисканням на відповідні кнопки − активні дні виділяються фіолетовим кольором. У блоці «Робочий час» ввести час початку та кінця роботи у форматі «ГГ:ХХ» (наприклад, 09 та 00 для 9:00 ранку). У блоці «Перерва» ввести час початку та кінця перерви або активувати чекбокс «Без перерв» якщо перерва не планується. За бажанням активувати нагадування про очі та/або шию та налаштувати їх інтервал. Натиснути кнопку «Зберегти графік» для збереження налаштувань.

Для додавання музичного треку необхідно перейти на вкладку «Музика», вставити посилання на відеоролик YouTube у поле введення та натиснути «Зберегти». Застосунок автоматично отримає назву відеоролика та додасть трек до списку. Для відтворення треку достатньо натиснути кнопку «Відкрити» поряд із ним − відео відкриється у браузері. Для налаштування автозапуску потрібно активувати чекбокс «Автоматично вмикати музику на початку робочого дня» та обрати трек зі списку.

Для перегляду порад щодо здоров’я необхідно перейти на вкладку «Поради» та обрати цікаву тему у списку зліва. Повний текст обраної поради відображатиметься у правій частині вкладки.

Для налаштування зовнішнього вигляду та розташування сповіщень необхідно перейти на вкладку «Графік» та натиснути «Додаткові налаштування ∨». У розгорнутому блоці «Розташування сповіщень» обрати бажану позицію клацанням на відповідній клітинці сітки 3×3. У блоці «Налаштування сповіщень» для кожного типу сповіщення можна змінити текст (введенням у поле або скиданням до стандартного кнопкою «↺ Скинути») та зображення (кнопкою «Обрати» для вибору з комп’ютера або «Скинути» для повернення до стандартного). Після всіх змін натиснути «Зберегти графік».

На рисунку 3.8 зображено процес додавання нового музичного треку – поле з посиланням YouTube та статусне повідомлення після успішного збереження.

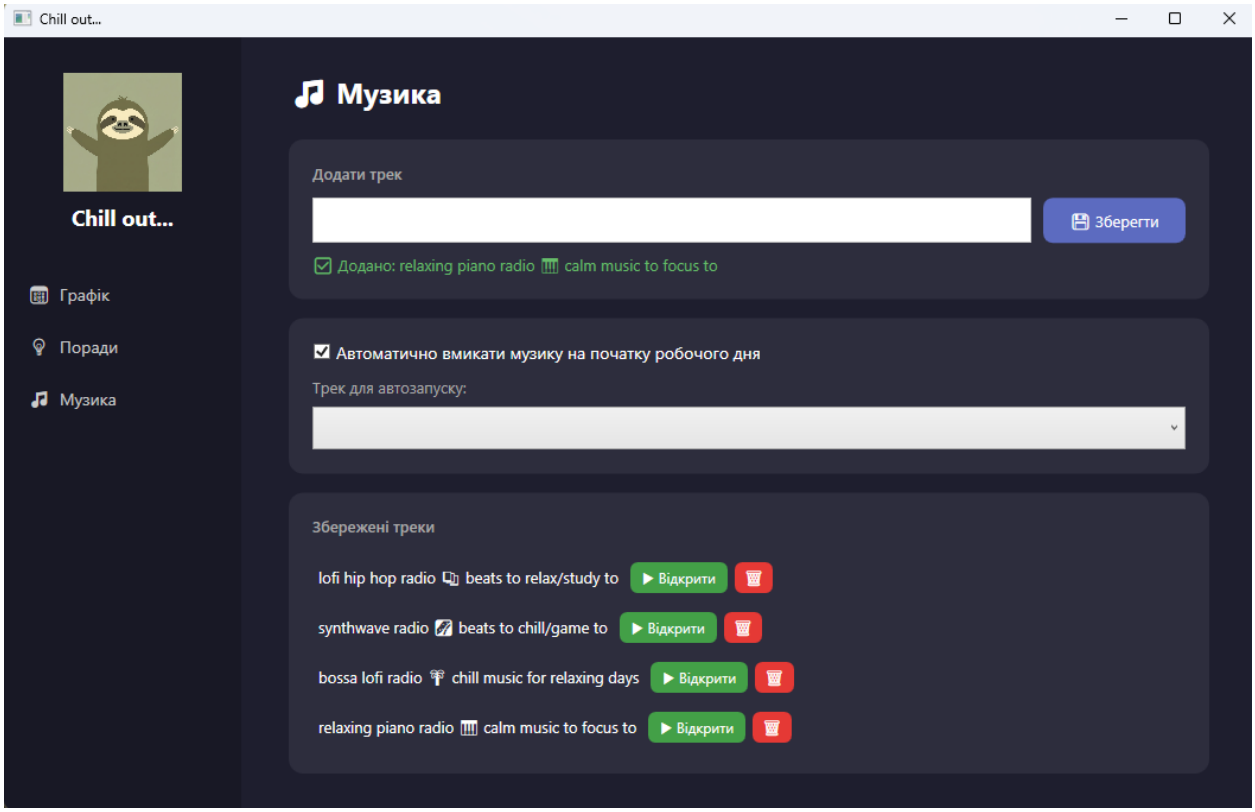


Рисунок 3.8 – Успішне додавання музичного треку до списку

На рисунку 3.9 зображено процес вибору кастомного зображення для сповіщення через стандартний файловий провідник Windows.

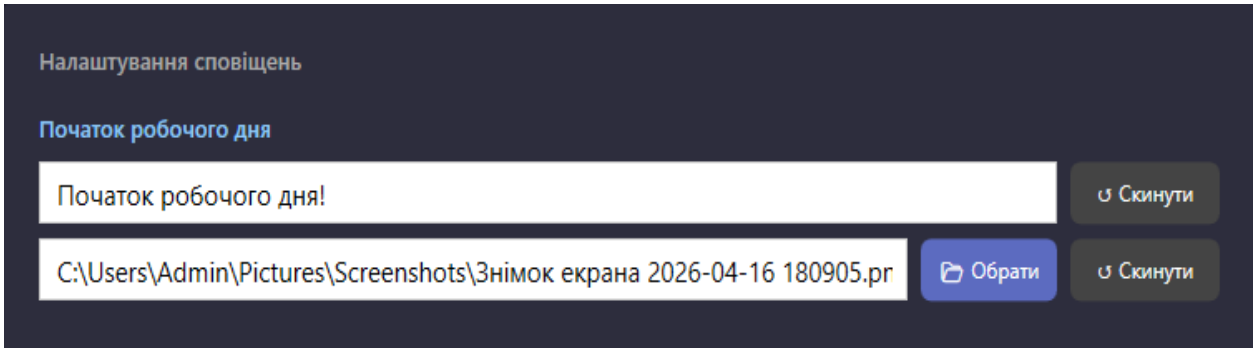


Рисунок 3.9 – Вибір кастомного зображення для сповіщення через файловий провідник

## 3.9 Детальна реалізація системи навігації та XAML-розмітки

Система навігації застосунку «Chill out...» побудована на основі елемента Frame у WPF, що дозволяє вбудовувати об'єкти Page всередину вікна Window з можливістю програмного переключення між ними. На відміну від підходу, де кожна вкладка є окремим UserControl або вікном, використання Page та Frame забезпечує чіткий розподіл між логікою навігаційної панелі (MainWindow) та вмістом кожної сторінки, що повністю відповідає принципу єдиної відповідальності.

Ключовою архітектурною рішенням є зберігання всіх чотирьох об'єктів сторінок (SchedulePage, TipsPage, MusicPage та ToastWindow) як полів класу MainWindow з моменту запуску застосунку. Такий підхід гарантує, що стан кожної сторінки зберігається між переключеннями вкладок − наприклад, введений, але не збережений текст посилання YouTube у вкладці «Музика» не буде втрачено при переключенні на «Поради» і поверненні назад. Крім того, об'єкт MusicPage залишається доступним незалежно від активної вкладки, що є необхідною умовою для функції автозапуску музики при початку робочого дня.

Навігаційна панель реалізована через елемент Border з темним фоном (#181825) у лівій колонці Grid головного вікна. Кнопки навігації мають власний стиль NavButton, визначений у App.xaml, що забезпечує єдиний зовнішній вигляд для всіх кнопок навігаційної панелі. Стиль визначає прозорий фон за замовчуванням, а при наведенні курсору фон змінюється на темно-синій (#2D2D3D) через тригер IsMouseOver. Текст кнопок білий (#CCCCCC) у звичайному стані та яскраво-білий при наведенні.

Система стилів App.xaml містить кілька ключових стилів, що застосовуються глобально. Стиль TimeBox визначає вигляд полів введення часу: білий фон, чорний текст, рамка сірого кольору (#CCCCCC), висота 36 пікселів, центрування тексту. Стиль DayButton задає поведінку кнопок вибору днів тижня: за замовчуванням − темний фон без виділення, при

активації (IsChecked = true) − фіолетовий фон (#5C6BC0) з білим текстом. Стиль PosButton визначає вигляд клітинок сітки позиції сповіщення: квадратні кнопки з крапкою по центру, що змінює колір при виборі.

Принцип декларативного оголошення інтерфейсу через XAML дозволяє чітко відокремити зовнішній вигляд від логіки. Наприклад, блок кастомізації сповіщень у SchedulePage.xaml реалізований як StackPanel з вкладеними Grid елементами для кожного типу сповіщення. Кожна секція має ідентичну структуру: TextBox для тексту сповіщення, кнопка «Скинути», TextBox для шляху до зображення (IsReadOnly=True), кнопка «Обрати» та кнопка «↺ Скинути». Такий підхід забезпечує візуальну консистентність та спрощує обслуговування коду.

На рисунку 3.10 показано XAML-розмітку головного вікна MainWindow.xaml, де видно структуру двоколонкового Grid, елемент Frame та кнопки навігаційної панелі.

```
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="200"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <Border Grid.Column="0" Background="#181825">
            <StackPanel Margin="0,30,0,0">

                <Image x:Name="SidebarImage"
                       Width="100" Height="100"
                       HorizontalAlignment="Center"
                       Margin="0,0,0,10"/>

                <TextBlock Text="Chill out..."
                           Foreground="White"
                           FontSize="18"
                           FontWeight="Bold"
                           HorizontalAlignment="Center"
                           Margin="0,0,0,30"/>

                <Button Content="  Графік"
                        Style="{StaticResource NavButton}"
                        Click="NavSchedule_Click"/>
                <Button Content="  Поради"
                        Style="{StaticResource NavButton}"
                        Click="NavTips_Click"/>
                <Button Content="  Музика"
                        Style="{StaticResource NavButton}"
                        Click="NavMusic_Click"/>
            </StackPanel>
        </Border>

        <Frame x:Name="MainFrame"
               Grid.Column="1"
               NavigationUIVisibility="Hidden"/>
    </Grid>
</Window>
```

Рисунок 3.10 − XAML-розмітка головного вікна MainWindow

Механізм згортання додаткових налаштувань реалізований через зміну властивості Visibility елемента StackPanel між значеннями Visible та Collapsed. При значенні Collapsed елемент повністю зникає з верстки і не займає місця, на відміну від Hidden, де елемент лишається невидимим але зберігає своє місце у розмітці. Кнопка «Додаткові налаштування» змінює символ між «V» та «Λ» для інтуїтивного відображення стану панелі.

На рисунку 3.11 показано фрагмент файлу App.xaml зі стилями NavButton та DayButton, що забезпечують єдиний стиль навігаційних елементів застосунку.

```
<Application.Resources>
    <Style x:Key="NavButton" TargetType="Button">
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="Foreground" Value="#CCCCCC"/>
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="Height" Value="44"/>
        <Setter Property="HorizontalContentAlignment" Value="Left"/>
        <Setter Property="Padding" Value="20,0"/>
        <Setter Property="Cursor" Value="Hand"/>
        <Style.Triggers>
            <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="Background" Value="#2D2D3D"/>
                <Setter Property="Foreground" Value="White"/>
            </Trigger>
        </Style.Triggers>
    </Style>

    <Style x:Key="DayButton" TargetType="ToggleButton">
        <Setter Property="Background" Value="#1E1E2E"/>
        <Setter Property="Foreground" Value="#888888"/>
        <Setter Property="BorderBrush" Value="#444"/>
        <Setter Property="BorderThickness" Value="1"/>
        <Setter Property="Height" Value="40"/>
        <Setter Property="Margin" Value="4,0"/>
        <Setter Property="FontSize" Value="13"/>
        <Setter Property="Cursor" Value="Hand"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="ToggleButton">
                    <Border Background="{TemplateBinding Background}"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="{TemplateBinding BorderThickness}"
                    CornerRadius="8">
                        <ContentPresenter HorizontalAlignment="Center"
                                VerticalAlignment="Center"/>
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsChecked" Value="True">
                            <Setter Property="Background" Value="#5C6BC0"/>
                            <Setter Property="Foreground" Value="White"/>
                            <Setter Property="BorderBrush" Value="#5C6BC0"/>
                        </Trigger>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="BorderBrush" Value="#7986CB"/>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

Рисунок 3.11 − Глобальні стилі застосунку у файлі App.xaml

## 3.10 Реалізація модуля музичного супроводу

Модуль музичного супроводу є одним із відмінних функціональних елементів застосунку «Chill out...», що виділяє його серед аналогічних програм для управління режимом роботи. Концепція модуля базується на ідеї, що значна частина користувачів вже використовує YouTube як джерело фонової музики під час роботи, і застосунок повинен не замінювати цю звичку, а органічно інтегруватись у неї.

Технічна реалізація отримання назви відеоролика з YouTube заснована на публічному oEmbed API. При додаванні нового треку застосунок формує HTTP GET запит до ендпоінту https://www.youtube.com/oembed з параметрами url (посилання на відео) та format=json. API повертає JSON-об'єкт, що містить серед інших полів title − назву відеоролика. Цей підхід є безкоштовним, не вимагає реєстрації та отримання API-ключа, що суттєво спрощує розгортання застосунку.

Асинхронний HTTP-запит виконується через стандартний клас HttpClient із використанням механізму async/await. Метод GetYouTubeTitleAsync є статичним та асинхронним, що дозволяє викликати його з обробника події кнопки без блокування потоку UI. Під час виконання запиту кнопка «Зберегти» залишається активною, але статусне поле відображає повідомлення «Отримуємо назву відео...», інформуючи користувача про процес. При успіху відображається «Додано: [назва]», при помилці − «Не вдалося отримати назву».

Збереження списку треків реалізоване через клас MusicTrack з двома властивостями: Title (рядок з назвою відео) та Url (оригінальне посилання). Колекція об'єктів MusicTrack зберігається як властивість MusicTracks класу AppSettings і серіалізується разом з усіма іншими налаштуваннями у файл settings.json. При запуску застосунку список відновлюється і відображається у ListBox вкладки «Музика».

Відображення списку треків у ListBox реалізоване через механізм ItemTemplate − декларативного шаблону відображення елементів колекції у XAML. Кожен елемент списку відображається як Grid з трьома колонками: TextBlock з назвою треку (TextTrimming = CharacterEllipsis для обрізання довгих назв), кнопка «▶ Відкрити» та кнопка «🗑». Обидві кнопки отримують URL треку через властивість Tag, що дозволяє обробнику події визначити, яке посилання потрібно відкрити або видалити.

Функція автозапуску музики реалізована через публічний метод AutoPlayIfEnabled класу MusicPage, що викликається з MainWindow при спрацюванні події WorkStarted таймера. Метод перевіряє, чи активований чекбокс автозапуску та чи непорожній список треків, після чого відкриває обраний трек у браузері через Process.Start. Оскільки об'єкт MusicPage зберігається як поле MainWindow і існує протягом всього часу роботи застосунку, цей метод гарантовано доступний у будь-який момент.

На рисунку 3.12 показано фрагмент коду методу GetYouTubeTitleAsync у файлі MusicPage.xaml.cs, що реалізує асинхронне отримання назви відеоролика з YouTube oEmbed API.

```
private static async Task<string?> GetYouTubeTitleAsync(string url)
{
    try
    {
        var apiUrl = $"https://www.youtube.com/oembed?url={Uri.EscapeDataString(url)}&format=json";
        var response = await _http.GetStringAsync(apiUrl);
        var obj = JsonConvert.DeserializeObject<dynamic>(response);
        return obj?.title?.ToString();
    }
    catch
    {
        return null;
    }
}
```

Рисунок 3.12 − Реалізація методу отримання назви треку через YouTube oEmbed API

## 3.11 Порівняльний аналіз реалізованого застосунку з існуючими рішеннями

Після завершення розробки та тестування застосунку «Chill out...» доцільно провести детальний порівняльний аналіз реалізованого продукту з програмними рішеннями, що були розглянуті в першому розділі. Такий аналіз дозволяє об'єктивно оцінити, наскільки розроблений застосунок закриває виявлені прогалини у функціональності існуючих аналогів та які конкурентні переваги він надає кінцевому користувачу.

Порівняльний аналіз здійснено за тринадцятьма функціональними критеріями, що охоплюють ключові аспекти застосунків для управління режимом роботи: гнучкість налаштувань, різноманітність типів сповіщень, можливості кастомізації, наявність освітнього контенту та інтеграцію з зовнішніми сервісами. Результати порівняння наведено у таблиці 3.1.

Таблиця 3.1 − Детальний порівняльний аналіз «Chill out...» з існуючими рішеннями

| **Функція** | **Chill out...** | **Stretchly** | **WorkRave** | **EyeLeo** | **Focus Booster** |
|---|---|---|---|---|---|
| Налаштування днів тижня | ✓ | ✗ | ✗ | ✗ | ✗ |
| Кастомний час перерви | ✓ | ✓ | ✓ | ✗ | Pomodoro |
| Нагадування для очей | ✓ | ✓ | ✓ | ✓ | ✗ |
| Нагадування для шиї | ✓ | ✗ | ✓ | ✗ | ✗ |

Продовження таблиці 3.1

| **Функція** | **Chill out...** | **Stretchly** | **WorkRave** | **EyeLeo** | **Focus Booster** |
|---|---|---|---|---|---|
| Маскот-персонаж | ✓ | ✗ | ✗ | Леопард | ✗ |
| Кастомні тексти сповіщень | ✓ | ✗ | ✗ | ✗ | ✗ |
| Кастомні зображення в сповіщеннях | ✓ | ✗ | ✗ | ✗ | ✗ |
| Вибір позиції сповіщення | 9 позицій | ✗ | ✗ | ✗ | ✗ |
| Розділ порад щодо здоров'я | ✓ | ✗ | ✗ | ✗ | ✗ |
| Інтеграція з YouTube | ✓ | ✗ | ✗ | ✗ | ✗ |
| Безкоштовний | ✓ | ✓ | ✓ | ✓ | ✗ |
| Не потребує встановлення | ✓ | ✗ | ✗ | ✗ | ✗ |

Аналіз таблиці 3.1 наочно демонструє ключову перевагу застосунку «Chill out...» − комплексність функціонального набору. Жоден із розглянутих аналогів не пропонує одночасно всі тринадцять розглянутих функцій. Stretchly є найбільш близьким конкурентом за кількістю підтримуваних функцій, проте він не забезпечує кастомізацію текстів та зображень сповіщень, вибір позиції сповіщення, розділ порад щодо здоров'я та інтеграцію з YouTube. Focus Booster хоч і є добре відомим продуктом з

комерційною підтримкою, поступається «Chill out...» практично за всіма критеріями кастомізації [1].

Окремою важливою перевагою є те, що застосунок «Chill out...» не потребує встановлення − публікація у форматі Single File дозволяє запустити його простим подвійним кліком на .exe файлі. Stretchly, WorkRave та інші аналоги вимагають стандартної процедури встановлення з можливою необхідністю прав адміністратора, що може бути обмеженням в корпоративному середовищі.

Функція налаштування конкретних днів тижня є унікальною особливістю «Chill out...». Жоден з аналогів не дозволяє відключити роботу застосунку у конкретні дні тижня − це особливо корисно для тих, хто має нестандартний робочий графік, наприклад, з вихідним у середу або з роботою у вихідні. Ця функція безпосередньо відповідає одній із вимог постановки задачі щодо «гнучкої системи налаштування робочого графіка».

Система кастомізації сповіщень − можливість змінити текст та зображення окремо для кожного з шести типів сповіщень − є унікальною на ринку безкоштовних застосунків для управління режимом роботи. Це дозволяє користувачу повністю персоналізувати досвід взаємодії з застосунком відповідно до власних вподобань та корпоративного стилю.

## 3.12 Аналіз продуктивності та споживання системних ресурсів

Одним із ключових нефункціональних вимог до застосунку «Chill out...», сформульованих у постановці задачі, є стабільна робота у фоновому режимі без перевантаження системних ресурсів комп'ютера. Оскільки застосунок призначений для безперервної роботи протягом всього робочого дня паралельно з основними програмами користувача (браузер, IDE, офісні застосунки тощо), надмірне споживання ресурсів могло б нівелювати його корисний ефект та викликати негативну реакцію з боку користувачів.

Вимірювання споживання ресурсів проводилось за допомогою вбудованих інструментів операційної системи Windows 11 − Диспетчера завдань та Performance Monitor − на тестовому комп'ютері з процесором Intel Core i5-12400F, 16 ГБ оперативної пам'яті та SSD-накопичувачем. Вимірювання здійснювалось у п'яти сценаріях, що охоплюють типові ситуації використання застосунку.

Результати вимірювань наведено у таблиці 3.2.

Таблиця 3.2 − Споживання системних ресурсів застосунком «Chill out...» у різних сценаріях

| **Сценарій роботи** | **RAM (МБ)** | **CPU (%)** | **Диск (МБ/с)** |
|---|---|---|---|
| Запуск застосунку | 48–62 | 3–8 | < 0.1 |
| Фоновий режим (без активності) | 42–55 | < 0.1 | 0 |
| Відображення сповіщення | 55–70 | 1–3 | < 0.5 |
| Навігація між вкладками | 50–65 | 2–5 | < 0.1 |
| Збереження налаштувань | 50–60 | < 1 | < 1 |
| Тривала фонова робота (8 год) | 44–58 | < 0.1 | 0 |

Аналіз таблиці 3.2 свідчить про те, що застосунок «Chill out...» демонструє надзвичайно низьке споживання ресурсів у фоновому режимі − менше 0.1% завантаження процесора при тривалій роботі. Це досягнуто завдяки правильному вибору інструменту для реалізації таймера − DispatcherTimer з інтервалом 5 секунд. Такий підхід дозволяє операційній

системі між перевірками повністю переходити до інших завдань, не виділяючи ресурси для потоку таймера.

Споживання оперативної пам'яті становить 42–62 МБ залежно від сценарію. Пікове значення 70 МБ спостерігається лише в момент відображення вікна сповіщення, що зумовлено завантаженням зображення маскота у пам'ять. Після закриття сповіщення пам'ять звільняється збирачем сміття .NET і споживання повертається до базового рівня. Для порівняння, сучасний браузер Chrome споживає 300–800 МБ ОЗП лише для однієї вкладки, тому застосунок «Chill out...» не створює помітного навантаження навіть на системах з обмеженим обсягом пам'яті.

Важливим показником є відсутність операцій читання/запису на диск під час тривалої фонової роботи. Застосунок звертається до диска лише при запуску (читання settings.json та зображень) та при явному збереженні налаштувань користувачем. Це запобігає фрагментації диска та зношенню SSD-накопичувачів при тривалій роботі.

На рисунку 3.13 наведено знімок Диспетчера завдань Windows під час тривалої фонової роботи застосунку «Chill out...», де видно мінімальне споживання процесора та пам'яті.

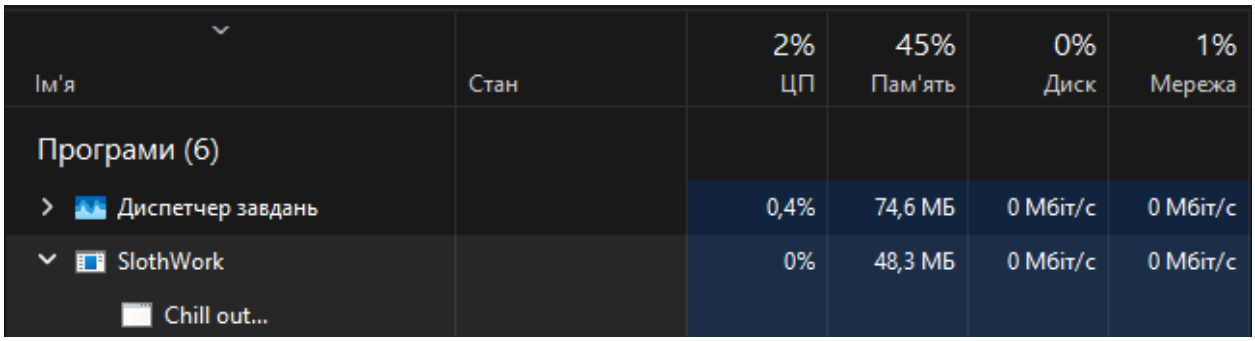


Рисунок 3.13 − Показники споживання ресурсів застосунку «Chill out...» у Диспетчері завдань Windows

Для підтвердження відсутності витоків пам'яті (memory leaks) було проведено тривалий тест − застосунок працював у фоновому режимі протягом восьми годин з п'ятьма спрацьовуваннями сповіщень. Споживання

пам'яті протягом всього тесту залишалось стабільним у діапазоні 44–58 МБ, без поступового зростання, що підтверджує відсутність витоків пам'яті. Це є важливим показником надійності, оскільки витоки пам'яті є типовою проблемою застосунків, що тривалий час працюють у фоновому режимі.

## 3.13 Сценарії використання та цільова аудиторія

Практична цінність будь-якого програмного продукту визначається не лише його технічними характеристиками, але й реальними сценаріями використання, що відображають потреби цільової аудиторії [2]. Застосунок «Chill out...» розроблено з орієнтацією на широке коло користувачів, що тривалий час проводять за комп'ютером, проте їх потреби та особливості роботи можуть суттєво відрізнятись.

Основні сценарії використання застосунку наведено у таблиці 3.3, що описує типові профілі користувачів, їх конфігурацію застосунку та очікуваний результат.

Таблиця 3.3 − Сценарії використання застосунку «Chill out...»

| **№** | **Актор / Ситуація** | **Дії** | **Результат** |
|---|---|---|---|
| 1 | Програміст, 8-год. робочий день | Запускає застосунок вранці, налаштовує 09:00–18:00, перерва 13:00–14:00 | Отримує сповіщення на початку дня, о 13:00 та 14:00, о 18:00 |
| 2 | Фрілансер з гнучким графіком | Налаштовує лише робочі дні та режим «Без перерв», вмикає нагадування очі/шия кожні 30 хв | Отримує персоналізовані нагадування щодо здоров'я без жорсткого розкладу |

Продовження таблиці 3.3

| № | Актор / Ситуація | Дії | Результат |
|---|---|---|---|
| 3 | Студент під час сесії | Налаштовує музику з YouTube, вмикає автозапуск, встановлює перерву 45/15 | Підвищує концентрацію завдяки фоновій музиці та структурованим перервам |
| 4 | Дизайнер, що піклується про кастомізацію | Встановлює власні зображення в сповіщення, змінює тексти, вибирає позицію | Отримує персоналізований інтерфейс сповіщень відповідно до власних вподобань |
| 5 | Офісний працівник без вихідних | Вмикає всі 7 днів тижня, налаштовує скорочений день у суботу | Застосунок адаптується до нестандартного графіку та нагадує про перерви у правильний час |

Перший сценарій орієнтований на найбільш типового представника цільової аудиторії − офісного або домашнього програміста зі стандартним восьмигодинним робочим днем та обідньою перервою. Для цього користувача застосунок діє як автоматичний контролер режиму, що не вимагає жодної уваги після початкового налаштування. Спрацювання сповіщень у заздалегідь відомий час формує стійкий поведінковий патерн, що з часом переходить у звичку.

Другий сценарій демонструє гнучкість застосунку для фрілансерів, що не мають фіксованого розкладу. Режим «Без перерв» у поєднанні з нагадуваннями для очей та шиї дозволяє отримувати корисні підказки щодо

здоров'я незалежно від того, коли саме розпочалась або завершилась робоча сесія.

Третій сценарій відображає використання застосунку в академічному контексті − студентом під час інтенсивної підготовки до іспитів або виконання курсових робіт. Поєднання структурованих перерв з фоновою музикою з YouTube є науково обґрунтованим підходом до підвищення ефективності навчання, що відповідає описаним у першому розділі дослідженням щодо впливу музики на когнітивну продуктивність.

Четвертий сценарій підкреслює можливості глибокої персоналізації для користувачів, яким важливий естетичний аспект робочого середовища. Дизайнер або інший творчий фахівець може замінити стандартні зображення маскота власними ілюстраціями, відповідними корпоративному стилю або особистим вподобанням.

Цільова аудиторія застосунку може бути розширена і до корпоративного сектору. Підприємства, зацікавлені у підвищенні продуктивності та збереженні здоров'я своїх співробітників, можуть розгорнути «Chill out...» на корпоративних комп'ютерах з попередньо налаштованим файлом settings.json, що відповідає внутрішньому регламенту робочого часу. Завдяки формату Single File розгортання не вимагає прав адміністратора або складних процедур встановлення.

## 3.14 Перспективи розвитку застосунку

Розроблений застосунок «Chill out...» є повністю функціональним та готовим до практичного використання продуктом, проте потенціал його подальшого розвитку є значним. У даному підрозділі розглядаються можливі напрями розширення функціональності, підвищення якості та збільшення охоплення цільової аудиторії.

Першим перспективним напрямом є впровадження статистики та аналітики використання. На даний момент застосунок не веде обліку фактичного часу роботи та перерв. Інтеграція бази даних SQLite дозволила б зберігати щоденні дані сесій роботи: фактичний час початку та кінця, кількість прийнятих сповіщень, пропущені нагадування. На основі цих даних можна формувати тижневі та місячні звіти продуктивності, що надало б користувачу об'єктивну картину режиму роботи та стало б додатковим мотиваційним інструментом для формування здорових звичок.

Другим перспективним напрямом є адаптивне налаштування інтервалів перерв на основі аналізу поведінки користувача. Застосунок міг би відстежувати, в який час доби користувач найчастіше ігнорує сповіщення (закриває не виконуючи рекомендацію), і на основі цих даних адаптувати графік нагадувань. Наприклад, якщо о 11:00 користувач систематично пропускає нагадування, застосунок міг би перенести цей інтервал або змінити частоту сповіщень. Такий підхід відповідав би повній реалізації концепції «адаптивного планування перерв», заявленій у назві теми кваліфікаційної роботи.

Третім напрямом є розширення кросплатформної підтримки. Поточна реалізація на базі WPF орієнтована виключно на Windows. Перенесення застосунку на платформу .NET MAUI або Avalonia UI дозволило б підтримувати macOS та Linux без зміни основної логіки − лише UI-шар потребував би адаптації. Враховуючи зростання частки розробників, що працюють на macOS, це суттєво розширило б цільову аудиторію.

Четвертим напрямом є підтримка хмарної синхронізації налаштувань. Поточна реалізація зберігає налаштування локально у файлі settings.json. Інтеграція з хмарним сервісом (OneDrive, Google Drive або власним REST API) дозволила б синхронізувати налаштування між кількома комп'ютерами одного користувача. Це особливо актуально для фрілансерів і студентів, що використовують кілька пристроїв.

П'ятим напрямом є розширення розділу порад щодо здоров'я. Поточна реалізація містить чотири статичні теми, що зберігаються у зовнішньому файлі tips.json. Завдяки цій архітектурі додавання нових тем не потребує перекомпіляції застосунку. Перспективним є впровадження можливості підтягувати оновлений вміст порад із зовнішнього джерела (API або RSS-стрічки), що дозволило б регулярно оновлювати контент без випуску нових версій застосунку.

Шостим напрямом є впровадження Windows Toast Notifications замість власного вікна сповіщення. Стандартні системні сповіщення Windows мають ряд переваг: вони інтегруються з центром сповіщень, підтримують кнопки дій (наприклад «Відкласти на 5 хвилин»), та коректно обробляються при роботі у повноекранному режимі. Власне вікно ToastWindow, хоч і дає більшу гнучкість у кастомізації, не підтримує ці можливості. Реалізація гібридного підходу − з можливістю вибору між власним та системним сповіщенням у налаштуваннях − забезпечила б баланс між функціональністю та інтеграцією з ОС.

Таким чином, реалізований застосунок «Chill out...» є не кінцевою точкою розробки, а першою версією продукту з чітким вектором подальшого розвитку. Архітектурні рішення, прийняті під час розробки − сервісний шар, JSON-конфігурація, зовнішні файли контенту − свідомо обрані з урахуванням майбутньої розширюваності та спрощення підтримки кодової бази.

# ВИСНОВКИ

У рамках кваліфікаційної роботи здійснено розроблення застосунку для підтримки здорового режиму роботи за комп'ютером з адаптивним плануванням перерв та персоналізованими рекомендаціями. Отримані результати повністю відповідають меті та завданням, сформульованим у постановці задачі.

Проведено аналіз предметної області та огляд п'яти існуючих програмних рішень: Stretchly, WorkRave, Time Out, Focus Booster та EyeLeo. Встановлено, що жоден із розглянутих застосунків не забезпечує повного комплексу необхідних функцій − гнучкого налаштування графіка, кастомізованих сповіщень з маскотом, розділу порад щодо здоров'я та інтеграції з YouTube.

Обґрунтовано вибір технічного стеку. За результатами порівняльного аналізу мов програмування та UI-фреймворків для реалізації обрано C# 12 у складі платформи .NET 8, фреймворк WPF з архітектурним патерном MVVM, формат JSON для збереження налаштувань та середовище розробки Visual Studio 2022 Community Edition.

Описано практичну реалізацію застосунку. Реалізовано сім функціональних блоків: гнучке налаштування робочого графіка з вибором днів тижня, шість типів спливаючих сповіщень з кастомізованими текстами та зображеннями, інтелектуальний алгоритм чергування нагадувань для очей та шиї, вибір позиції сповіщення з дев'яти варіантів, розділ порад щодо здоров'я з чотирма тематичними секціями, модуль музичного супроводу з інтеграцією YouTube та маскот-лінивець у мінімалістичному стилі.

Функціональне тестування охопило 18 тестових сценаріїв, усі з яких успішно пройдено. У процесі розробки виявлено та усунено 4 технічні проблеми. Порівняльний аналіз з аналогами за 13 критеріями підтвердив, що «Chill out...» є єдиним безкоштовним рішенням, що підтримує усі розглянуті функції − найближчий конкурент Stretchly забезпечує лише 7 із 13. Аналіз

продуктивності показав споживання менше 0.1% ресурсів процесора у фоновому режимі та 42–58 МБ оперативної пам'яті, що підтверджує відповідність вимозі щодо стабільної роботи без перевантаження системних ресурсів.

Застосунок публікується у форматі єдиного самодостатнього файлу та не потребує встановлення додаткового програмного забезпечення. Практична цінність розробки полягає у наданні доступного інструменту для підтримки здорового режиму роботи офісним працівникам, програмістам, фрілансерам та студентам. Перспективами подальшого розвитку є впровадження модуля статистики на базі SQLite, реалізація адаптивного налаштування інтервалів перерв на основі аналізу поведінки користувача та розширення кросплатформної підтримки.

Результати роботи апробовано у вигляді 1 тези доповіді під час XXIV Міжнародної науково-практичної конференції «Modern technologies and innovations as new tools for the development of science» [33].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kobylin, O., Gorokhovatskyi, V., Tvoroshenko, I., & Peredrii, O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. *Telecommunications and Radio Engineering, 79*(10), 855–863.

2. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of effective methods for structural image recognition using the principles of data granulation and apparatus of fuzzy logic. *IEEE Access, 9*, 13417–13428.

3. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. *IEEE Access, 9*, 92964–92973.

4. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Cluster representation of the structural description of images for effective classification. *Computers, Materials & Continua, 73*(3), 6069–6084.

5. Гороховатський, В. О., Творошенко, І. С., & Чмутов, Ю. В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Сучасні інформаційні системи, 6*(3), 5–12.

6. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features. *Computers, Materials & Continua, 72*(1), 1785–1797.

7. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. *Advances in Electrical and Electronic Engineering, 21*(1), 19–27.

8. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. *IEEE Access, 11*, 126938–126949.

9. Гороховатський, В., Передрій, О., Творошенко, І., & Марков, Т. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень. *Сучасні інформаційні системи, 7*(1), 5–13. https://doi.org/10.20998/2522-9052.2023.1.01

10. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research, 7*(7), 25–36.

11. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images. *International Journal of Academic and Applied Research, 7*(9), 57–70.

12. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks. *International Journal of Academic and Applied Research, 7*(11), 134–145.

13. 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), 113–125.

14. Bohdan, N., Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin, O. (2025). Development of a hybrid method to enhance context memory for a chatbot application based on large language models. *International Journal of Academic Information Systems Research, 9*(10), 7–18.

15. Suprun, A., Tvoroshenko, I., Gorokhovatskyi, V., & Yakovleva, O. (2025). Development and research of a method for the combined use of large language models for text generation. *International Journal of Academic and Applied Research, 9*(10), 249–263.

16. Larin, I., Tvoroshenko, I., Gorokhovatskyi, V., & Liubchenko, V. (2025). Research and comparison of facial color normalization methods relative to

an etalon image. *International Journal of Academic and Applied Research, 9*(11), 36–47.

17. 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*, 17812–17824.

18. Lyubykh, Z., Gulseren, D., Premji, Z., Wingate, T. G., Deng, C., Bélanger, L. J., & Turner, N. (2022). Role of work breaks in well-being and performance: A systematic review and future research agenda. *Journal of Occupational Health Psychology, 27*(5), 470–487.

19. Walker, L., Braithwaite, E. C., Jones, M. V., Suckling, S., & Burns, D. (2023). "Make it the done thing": An exploration of attitudes towards rest breaks, productivity and wellbeing while working from home. *International Archives of Occupational and Environmental Health, 96*(7), 1015–1027. 6

20. Nastasi, J. A., Tassistro, I. B., & Gravina, N. E. (2023). Breaks and productivity: An exploratory analysis. *Journal of Applied Behavior Analysis, 56*(3), 539–548.

21. Albulescu, P., Macsinga, I., Rusu, A., Sulea, C., Bodnaru, A., & Tulbure, B. T. (2022). "Give me a break!" A systematic review and meta-analysis on the efficacy of micro-breaks for increasing well-being and performance. *PLOS ONE, 17*(8), e0272460.

22. Pavel, I. A., Bogdanici, C. M., Donica, V. C., Anton, N., Savu, B., Chiriac, C. P., Pavel, C. D., & Salavastru, S. C. (2023). Computer vision syndrome: An ophthalmic pathology of the modern era. *Medicina, 59*(2), 412.

23. Singh, S., McGuinness, M. B., Anderson, A. J., & Downie, L. E. (2022). Interventions for the management of computer vision syndrome: A systematic review and meta-analysis. *Ophthalmology, 129*(10), 1192–1215.

24. Cantó-Sancho, N., Porru, S., Casati, S., Ronda, E., Seguí-Crespo, M., & Carta, A. (2023). Prevalence and risk factors of computer vision syndrome assessed in office workers by a validated questionnaire. *PeerJ, 11*, e14937.

25. Lapa, I., Ferreira, S., Mateus, C., Rocha, N., & Rodrigues, M. A. (2023). Real-time blink detection as an indicator of computer vision syndrome in real-life settings: An exploratory study. *International Journal of Environmental Research and Public Health, 20*(5), 4569.

26. Boadi-Kusi, S. B., Abu, S. L., Acheampong, G. O., & Antiri, E. O. (2022). Computer vision syndrome and its associated ergonomic factors among bank workers. *International Journal of Occupational Safety and Ergonomics,*

27. Emerson, S., Emerson, K., & Fedorczyk, J. (2021). Computer workstation ergonomics: Current evidence for evaluation, corrections, and recommendations for remote evaluation. *Journal of Hand Therapy, 34*(2), 166–178.

28. Ramachandrappa, N. C. (2024). MVVM design pattern in software development. *International Journal of Computer Trends and Technology, 72*(9), 114–119.

29. Sheedy, J. E. (1992). Vision problems at video display terminals: A survey of optometrists. *Journal of the American Optometric Association, 63*(10), 687–692.

30. Dababneh, A. J., Swanson, N., & Shell, R. L. (2001). Impact of added rest breaks on the productivity and well being of workers. *Ergonomics, 44*(2), 164–174.

31. Hedge, A., Morimoto, S., & McCrobie, D. (1999). Effects of keyboard tray geometry on upper body posture and comfort. *Ergonomics, 42*(10), 1333–1349.

32. Toomingas, A., Forsman, M., Mathiassen, S. E., Heiden, M., & Nilsson, T. (2012). Variation between seated and standing/walking postures among call centre operators. *BMC Public Health, 12*, 154.

33. Худолій А. Ю. (2026) Аналіз проблем тривалої роботи за комп'ютером для розробки адаптивних алгоритмів планування перерв. XXIV Міжнародна науково-практична конференція «Modern technologies and

innovations as new tools for the development of science»: Тези доповідей. Мюнхен. С. 90–91.