Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтерактивного календаря завдань для оптимізації робочого процесу IT-фахівців

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Бойко А.А.
(підпис) (прізвище та ініціали)

Керівник _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Бойко Анастасії Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтерактивного календаря завдань для оптимізації робочого процесу ІТ-фахівців

Керівник роботи Млинко Богдана Богданівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*7*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 23 червня 2025р.

3. Вихідні дані до роботи базуються на основі досліджених літературних та інтернет-джерелах

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання на розробку. 1.1 Актуальність теми та опис предметної області. 1.2 Огляд існуючих рішень та їх аналіз. 1.3 Постановка завдання для розробки програмної системи. 1.4 Висновок до першого розділу. 2 Теоретичні основи та вимоги до розробки програмної системи. 2.1 Загальні концепції та принципи розробки програмного забезпечення. 2.1.1 Вибір методології розробки та дослідницьких методів для проєкту. 2.2 Аналіз та формалізації вимог до системи управління завданнями. 2.3 Огляд та обґрунтування вибору технологій для розробки десктопного застосунку. 2.4 Висновок до другого розділу. 3 Проектування та реалізація програмної частини. 3.1 Архітектура та реалізація програмної системи. 3.1.1 Реалізація рівня користувацького інтерфейсу (GUI). 3.1.2 Реалізація рівня бізнес-логіки. 3.1.3 Реалізація рівня доступу до даних та системних сповіщень. 3.2 Демонстрація функціональних можливостей та користувацького інтерфейсу. 3.3 Апробація та тестування системи. 3.4 Висновок до третього розділу. 4 Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки. 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, Слайдів). Повний перелік слайдів у презентації, включаючи перший та останній слайди номерами, через крапку). 1. Титульний лист. 2. Мета об'єкт та предмет дослідження. 3.Огляд існуючих рішень. 4. Актуальність обраної теми. 5. Завдання на розробку інтерактивного календаря завдань. 6. Теоретичні основи та вимоги до системи. 7. Діаграма варіантів використання. 8. Архітектура програмної системи. 9. Користувацький інтерфейс. 10. Функція керування завданнями. 11. Налаштування теми інтерактивного календаря завдань. 12. Порівняння функціональних можливостей існуючих рішень та розробленого календаря завдань. 13. Апробація та тестування. 14. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С.	09.06.2025	20.06.2025

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	31.01.2025-03.02.2025	
3.	Виконання дослідження щодо функціональних та нефункціональних вимог	04.02.2025-06.02.2025	
	Розроблення інтерактивного календаря завдань		
4.	Оформлення розділу «Аналіз предметної області та постановка завдання на розробку»	02.06.2025-05.06.2025	
5.	Оформлення розділу «Теоретичні основи та вимоги до розробки програмної системи»	06.06.2025-08.06.2025	
6.	Оформлення розділу «Реалізація та проектування програмної частини»	09.06.2025-11.06.2025	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2025-13.06.2025	
8.	Виконання завдання до підрозділу «Основи охорони	14.06.2025-15.06.2025	
	праці»		
9.	Оформлення кваліфікаційної роботи	16.06.2025-17.06.2025	
10.	Нормоконтроль	18.06.2025-19.06.2025	
11.	Перевірка на плагіат	20.06.2025	
12.	Попередній захист кваліфікаційної роботи	21.06.2025	
13.	Захист кваліфікаційної роботи	29.06.2025	

Студент _____ Бойко А.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтерактивного календаря завдань для оптимізації робочого процесу ІТ-фахівців // Кваліфікаційна робота освітнього рівня «Бакалавр» // Бойко Анастасія Андріївна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2025 // С. 70, рис. – 24 , табл. – 2 , слайди – 14 , додат. – 3 , бібліогр. – 46 .

Ключові слова: інтерактивний календар, управління завданнями, тайм-менеджмент, оптимізація робочого процесу, ІТ-фахівці, персоналізація, десктопний застосунок, автономність.

Кваліфікаційна робота присвячена дослідженню та розробці інструменту для ефективного управління часом та завданнями. В першому розділі кваліфікаційної роботи обґрунтовано актуальність теми управління завданнями та її вплив на продуктивність ІТ-фахівців. Висвітлено огляд та проведено порівняльний аналіз існуючих програмних рішень для планування та управління завданнями. Розглянуто виявлені прогалини на ринку, а також сформульовано проблему, мету та завдання розробки програмної системи.

В другому розділі кваліфікаційної роботи досліджено загальні концепції та принципи розробки програмного забезпечення, включаючи моделі життєвого циклу та принципи проєктування. Обґрунтовано вибір методології розробки та дослідницьких методів для проєкту. Подано аналіз та формалізацію функціональних та нефункціональних вимог до системи, а також обґрунтовано вибір технологій для розробки десктопного застосунку.

В третьому розділі кваліфікаційної роботи описано архітектуру та безпосередню реалізацію програмної системи, включаючи рівні користувацького інтерфейсу, бізнес-логіки та доступу до даних. Проаналізовано демонстрацію функціональних можливостей та

користувацького інтерфейсу з візуальними прикладами. Проведено апробацію та тестування розробленої системи серед цільової аудиторії (ІТ-фахівців). Об’єкт дослідження – процес управління часом та завданнями, що впливає на продуктивність ІТ-фахівців. Предмет дослідження – методи та програмні засоби для оптимізації управління завданнями та часом в індивідуальному робочому процесі ІТ-фахівців.

ANNOTATION

Development of an Interactive Task Calendar for IT Specialists' Workflow Optimization // Qualification work of the educational level «Bachelor» // Boiko Anastasiia // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2025 // P. 70, fig. – 24 , tabl. – 2 , slides – 14 , annexes. – 3 , references – 46.

Keywords: interactive calendar, task management, time management, workflow optimization, IT specialists, personalization, desktop application, autonomy.

The qualification work is dedicated to the research and development of a tool for effective time and task management. The first section of the qualification work substantiates the relevance of the topic of task management and its impact on the productivity of IT specialists. An overview and comparative analysis of existing software solutions for planning and task management are presented. Identified market gaps are reviewed, and the problem, aim, and objectives for the development of the software system are formulated.

The second section of the qualification work explores general concepts and principles of software development, including life cycle models and design principles. The choice of development methodology and research methods for the project is justified. An analysis and formalization of functional and non-functional requirements for the system are provided, along with a justification for the selection of technologies for desktop application development.

The third section of the qualification work describes the architecture and direct implementation of the software system, including the user interface, business logic, and data access layers. A demonstration of the functional capabilities and user interface with visual examples is analyzed. The approbation and testing of the

developed system among the target audience of IT specialists are conducted. The object of research is the process of time and task management that affects the productivity of IT specialists. The subject of research is the methods and software tools for optimizing task and time management in the individual workflow of IT specialists.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

Agile (Agile software development) – гнучка методологія розробки програмного забезпечення.

CRM (Customer Relationship Management) – управління взаємовідносинами з клієнтами.

DRY (Don't Repeat Yourself) – принцип розробки програмного забезпечення, що закликає уникати дублювання коду.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

IT (Information Technology) – інформаційні технології.

KISS (Keep It Short and Simple) – принцип розробки програмного забезпечення, що заохочує створення простих рішень.

MVC (Model-View-Controller) – архітектурний шаблон програмної інженерії.

SDLC (Software Development Life Cycle) – життєвий цикл розробки програмного забезпечення.

SQL (Structured Query Language) – структурована мова запитів.

UX (User Experience) – досвід користувача.

ВВК – вуглекислотний вогнегасник.

ВП – порошковий вогнегасник.

ПЗ – програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок швидкої цифрової трансформації бізнесу та зростання обсягів інформації ІТ-фахівці працюють у середовищі багатозадачності, жорстких дедлайнів і віддаленої співпраці. Велика кількість розробників скаржаться про регулярне перевантаження та дефіцит уваги, що знижує якість роботи та підвищує ризик прокрастинації. Хмарні таск-менеджери потребують постійного підключення до Інтернету й часто перевантажені надлишковим функціоналом, натомість прості локальні інструменти не підтримують гнучких нагадувань і персоналізації. Тому створення автономного інтерактивного календаря завдань, орієнтованого саме на потреби ІТ-спеціалістів, є актуальним напрямком сучасних досліджень у галузі програмної інженерії, UX та тайм-менеджменту.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення ефективності особистого тайм-менеджменту ІТ-фахівців шляхом розроблення десктопного застосунку «Інтерактивний календар завдань. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати сучасні наукові та прикладні дослідження з тайм-менеджменту в ІТ-індустрії;
- оцінити популярні інструменти управління завданнями та виявити їхні переваги й недоліки щодо автономності, гнучкості й UX;
- сформулювати функціональні та нефункціональні вимоги до системи з урахуванням потреб розробників;
- спроєктувати структуру програмного забезпечення з урахуванням принципів модульності та розмежування відповідальностей;
- реалізувати програмний прототип із використанням сучасних засобів розробки;
- провести тестування за участю цільової аудиторії, зібрати зворотний зв’язок та оцінити вплив використання системи на продуктивність.

Практичне значення одержаних результатів. Розроблений інтерактивний календар завдань дозволяє ефективно організовувати особистий робочий процес IT-фахівців в автономному середовищі без необхідності підключення до серверів або мережі. Система забезпечує зручне керування завданнями, візуалізацію їхнього статусу на календарі та своєчасне сповіщення про наближення дедлайнів, що сприяє підвищенню дисципліни, продуктивності та зниженню рівня прокрастинації. Отримані результати можуть бути використані для подальшого вдосконалення персональних або корпоративних інструментів планування, а також у навчальному процесі при вивченні дисциплін, пов’язаних із розробкою програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ

1.1 Огляд існуючих рішень та їх аналіз для оптимізації робочого процесу IT-фахівців

Ринок програмних продуктів для організації роботи та планування часу дуже широкий і різноманітний, що пропонує широкий вибір відповідей – від простих блокнотів до складних проєктів, які допомагають спланувати будь-яке завдання. Серед найпопулярніших аналогів можна виділити найбільш використовувані програми, такі як Google Calendar, Microsoft To Do, Todoist, Trello та ClickUp [1]. Кожен з цих інструментів має свої унікальні особливості, цільову аудиторію та модель розповсюдження.

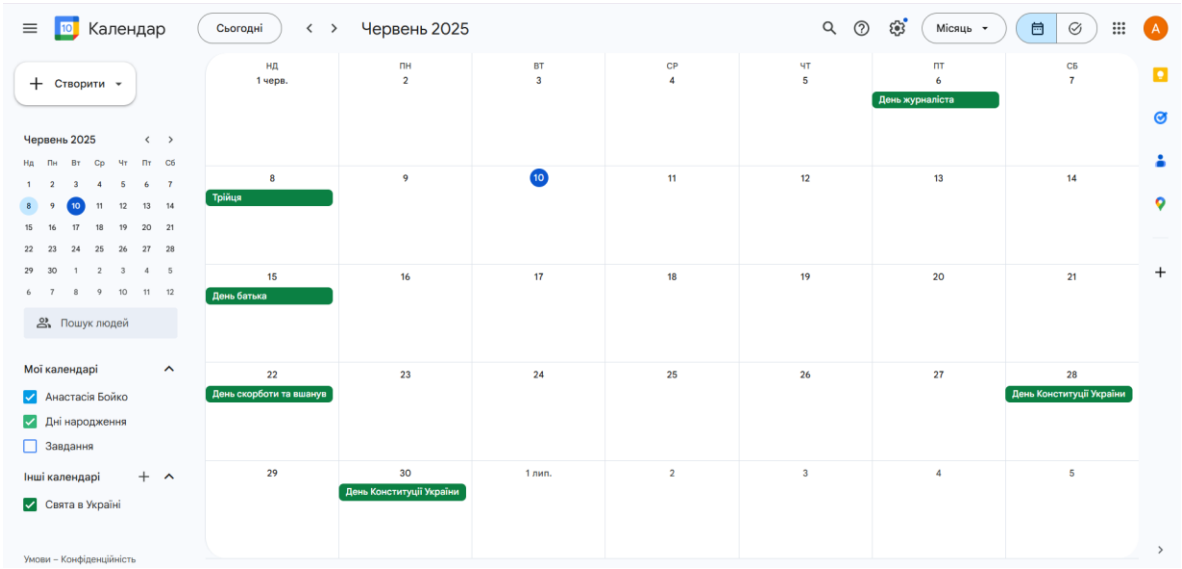

Рисунок 1.1 – Інтерфейс Google Calendar

Google Calendar – це відоме хмарне рішення, яке допомагає легко планувати події та завдання, а також інтегрується з іншими службами Google. Його головні переваги – універсальна доступність (веб, мобільні пристрої), можливість обміну даними та інтеграція з екосистемою Google. Однак, для

деяких користувачів його функціональність для детального керування завданнями може бути дещо обмеженою, і для роботи потрібне постійне підключення до Інтернету. На рисунку 1.1 представлено інтерфейс Google Calendar.

Microsoft To Do (раніше Wunderlist) зосереджується на простих списках завдань. Він забезпечує їх синхронізацію між пристроями та інтеграцію з продуктами Microsoft. Цей інструмент є інтуїтивно зрозумілим для швидкого створення та відстеження завдань, що робить його привабливим для особистого використання (див. рисунок 1.2). Проте він покладається на хмарну інфраструктуру, яка не завжди зручна для роботи в автономному режимі.

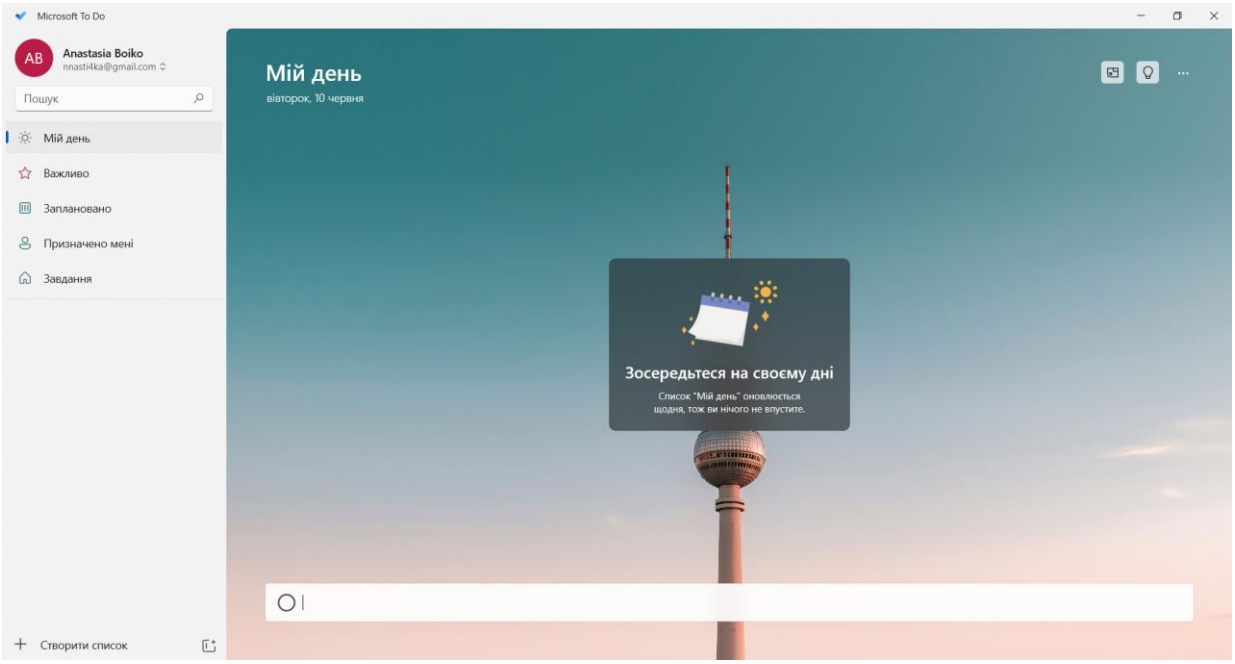

Рисунок 1.2 – Вигляд Microsoft To Do

Todoist позиціонується як потужний таск-менеджер для особистого та командного використання, який пропонує розширені можливості для організації завдань, пріоритизації, проектів та інтеграції з численними сторонніми сервісами [1]. Він відрізняється гнучкістю та багатством функціоналу, але значна частина цих можливостей доступна лише у платних версіях. На рисунку 1.3 представлено вікно програми Todoist.

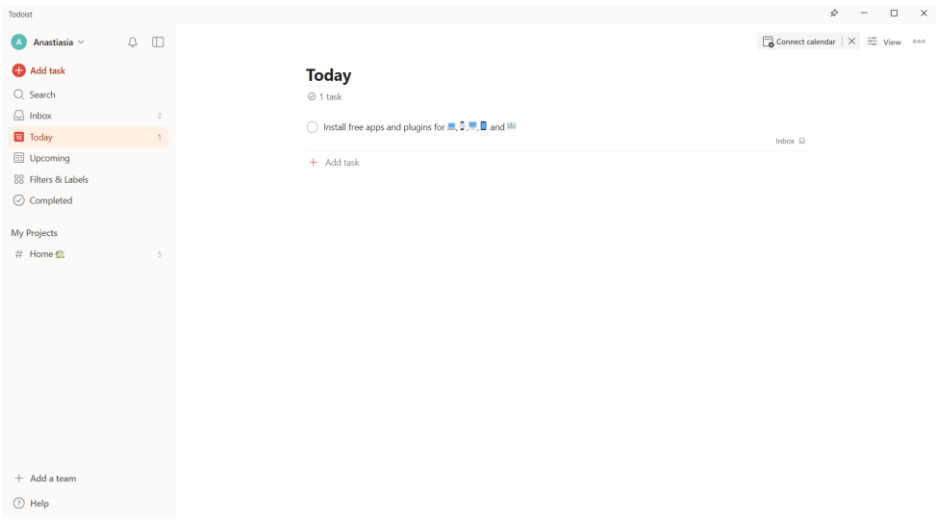

Рисунок 1.3 – Вікно програми Todoist

Trello використовує візуальний підхід Kanban-дошок – метод управління роботою, що візуалізує робочий процес і обмежує незавершену роботу – для управління завданнями, що робить його ідеальним для командної роботи та візуалізації прогресу [2]. Він дозволяє легко переміщувати картки (завдання) між стовпчиками («До виконання», «В роботі», «Виконано»), що зручно для відстеження робочого процесу. Незважаючи на свою простоту у використанні, Trello може бути надлишковим для суто індивідуального планування, а його основна функціональність також залежить від хмарних сервісів. На рисунку 1.4 зображено головну сторінку планувальника Trello.

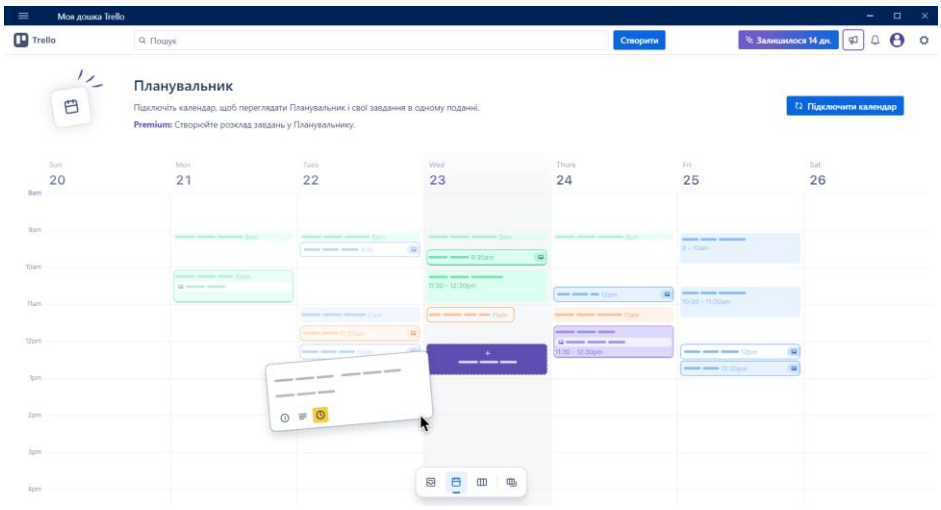

Рисунок 1.4 – Головна сторінка планувальника Trello

ClickUp є комплексним рішенням, що поєднує можливості таск-менеджера, Customer Relationship Management (CRM), інструментів для спільної роботи та багато іншого. Він пропонує величезну кількість функцій та гнучких налаштувань, що робить його потужним інструментом для великих проєктів та команд. Однак, ця універсальність може стати недоліком для користувачів, яким потрібен простий та спеціалізований інструмент, оскільки інтерфейс ClickUp може здаватися перевантаженим, а для ефективної роботи потрібне постійне підключення до мережі (див. рисунок 1.5).

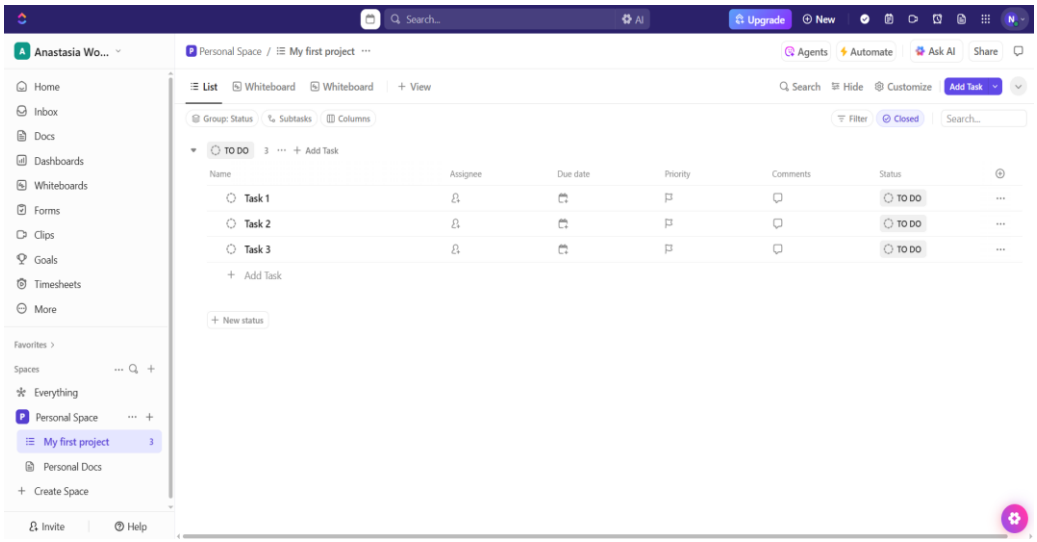

Рисунок 1.5 – Приклад зовнішнього вигляду ClickUp

Результати проведеного аналізу функціональних можливостей існуючих рішень наведені в таблиці 1.1. Всебічне та структуроване порівняння дозволяє наочно виявити ключові відмінності, сильні та слабкі сторони наявних інструментів, а також чітко продемонструвати, яким чином ідеальне рішення зможе заповнити існуючі прогалини та краще відповідати специфічним потребам IT-фахівців. Для порівняння були обрані представники таких категорій інструментів: корпоративний/командний менеджер завдань (як-от Trello), персональні хмарні органайзери (зокрема Google Calendar, Todoist, Microsoft To Do). Порівняння проводилося за основними критеріями, які є

найбільш значущими для цільової аудиторії, зокрема для IT-фахівців, які цінують конфіденційність, гнучкість та автономність.

Таблиця 1.1 – Порівняння функціональних можливостей існуючих рішень

Категорія характеристик	**Trello/ ClickUp**	**Google Calendar/Todoist/MS To Do**
Призначення	Командне управління проєктами, співпраця	Персональне планування, синхронізація
Тип застосунку	Веб-орієнтований, хмарний	Хмарний, кросплатформенний (веб/мобільний)
Авторизація	Обов’язкова	Обов’язкова
Конфіденційність даних	Залежить від політики провайдера	Залежить від політики провайдера
Вартість	Часто платні підписки	Є безкоштовні та платні версії
Гнучкість налаштування GUI	Обмежена (стандартні теми)	Обмежена (стандартні теми)
Конфіденційність даних	Залежить від політики провайдера	Залежить від політики провайдера
Вартість	Часто платні підписки	Є безкоштовні та платні версії
Гнучкість налаштування GUI	Обмежена (стандартні теми)	Обмежена (стандартні теми)
Візуалізація завдань на календарі	Залежить від інструменту, часто менш деталізована	Стандартна (відображення подій, без агрегованих статусів)
Складність використання	Висока	Середня
Вимоги до ресурсів	Залежить від браузера/додатку, інколи високі	Середні
Підтримка мотиваційних елементів	Відсутня	Відсутня

Проведений порівняльний аналіз, представлений у табл. 1.1, дозволяє виявити низку суттєвих проблем та незадоволених потреб у сегменті існуючих програм для управління завданнями, що є особливо актуальним з погляду IT-

фахівців. Перш за все, домінуюча більшість проаналізованих рішень є хмарними. Це створює низку обмежень, включаючи залежність від стабільного інтернет-з'єднання та потенційні питання щодо конфіденційності даних, оскільки їх зберігання повністю залежить від політики сторонніх провайдерів. Для ІТ-фахівців, які часто працюють з конфіденційною інформацією, це є значним недоліком. Крім того, обов'язкова авторизація у таких системах може ускладнювати швидкий доступ до власних даних, у випадку коли необхідно терміново перевірити плани.

По-друге, існуючі рішення демонструють розбіжності у своєму призначенні та, як наслідок, у рівні функціональної універсальності. Якщо корпоративні менеджери завдань пропонують широкий спектр можливостей для командної роботи, то часто цей функціонал є надлишковим для потреб індивідуального ІТ-фахівця, що шукає зосереджений інструмент для особистої продуктивності. Персональні ж хмарні органайзери, хоча й є простішими, часто мають обмежені можливості візуалізації завдань на календарі та не надають деталізованих агрегованих статусів.

По-третє, спільними недоліками для багатьох рішень є обмежена гнучкість налаштування користувацького інтерфейсу (GUI) та відсутність вбудованих мотиваційних елементів. Це створює певний дискомфорт для користувачів, яким потрібен інструмент, що адаптується до їхніх індивідуальних візуальних уподобань та сприяє підтримці продуктивності через психологічні стимули. Більше того, значна частина потужних функцій у багатьох аналогів доступна лише за платною підпискою, що створює бар'єр для безкоштовного використання та може бути невиправданим для індивідуальних потреб.

Таким чином, результати аналізу існуючих рішень чітко вказують на необхідність створення автономного, десктопного інструменту для управління завданнями. Такий ідеальний інструмент мав би забезпечувати високу конфіденційність даних через локальне зберігання, бути інтуїтивно зрозумілим у використанні, надавати розширені можливості візуалізації завдань на

календарі з деталізацією статусів, пропонувати високу гнучкість налаштування інтерфейсу, демонструвати не тільки заплановані дії на день, але й на цілий місяць та включати елементи підтримки мотивації. Розробка подібного рішення дозволить заповнити існуючу прогалину на ринку та забезпечити IT-фахівців ефективним інструментом для оптимізації їхнього індивідуального робочого процесу, що працює незалежно від зовнішніх сервісів.

1.2 Актуальність теми та опис предметної області

Виходячи з проведеного огляду та виявлених обмежень існуючих рішень, стає очевидною нагальна потреба в нових підходах до організації робочого процесу IT-фахівців. У сучасному швидкоплинному світі, що характеризується постійно зростаючим обсягом інформації та динамічним розвитком технологій, ефективне управління часом і завданнями стає одним із ключових аспектів як особистого, так і, особливо, професійного успіху спеціалістів у галузі інформаційних технологій (IT) [3]. Специфіка їхньої роботи часто передбачає управління великою кількістю проєктів і різноманітних технічних завдань у стислі терміни з розподіленими членами команди. Стрімкий розвиток технологій і зміни в методах роботи – насамперед через збільшення віддаленої роботи – вимагають від фахівців та їхніх команд чіткого планування, організації та контролю за виконанням професійних завдань [4].

Неефективне управління часом серед таких працівників призводить до зниження продуктивності, спричиняє стресові перевантаження, професійне вигорання та ризик пропуску важливих дедлайнів у проєктах [5]. Одним з головних викликів, що стосуються ефективності в професійній діяльності, є схильність відкладати справи, попри усвідомлення негативних наслідків (прокрастинація). Це явище може виникати через відсутність мотивації до рутинних завдань, страх перед невдачею у складних технічних проєктах або просто через надмірне навантаження та великий обсяг зобов'язань.

Боротьба з подібними викликами та підвищення загальної продуктивності вимагає не лише самодисципліни, а й наявності якісних, упорядкованих інструментів, що допомагають систематизувати хаотичну роботу та своєчасно нагадувати про важливі етапи проєкту [5]. У цьому контексті, цифрові календарі, планувальники та диспетчери завдань трансформували традиційні способи організації робочого часу, пропонуючи зручні інструменти для візуалізації, відстеження та контролю власних планів.

Область «управління завданнями» охоплює різні процеси, методи та інструменти, орієнтовані на організацію та моніторинг роботи, виконаної для досягнення конкретних цілей. Це базове поняття як для управління проектами, так і для особистої професійної організації, яке особливо актуальне для IT-фахівців, які зазвичай виконують великий обсяг різноманітних завдань. Основними етапами управління завданнями є:

- формулювання завдання, тобто чітке визначення мети, обсягу та вимог до завдання, які в IT-сфері можуть стосуватися розробки функціональності, виправлення помилок або дослідницької роботи;
- пріоритезація – оцінка важливості та терміновості завдання з метою ефективного розподілу зусиль, що дозволяє зосередитися на критичних елементах проекту планування, а саме розробка послідовності кроків, визначення термінів, необхідних ресурсів та відповідальних осіб (у контексті особистого календаря – власних ресурсів);
- виконання: робота над завданням, наприклад написання коду, тестування, документування;
- моніторинг: відстеження ходу виконання завдання та проблем для коригування планів для своєчасного завершення проекту;
- завершення: підсумкове виконання завдання з інтеграцією виконаної роботи та оцінкою результатів.

Хороша система управління завданнями може як структурувати, так і оптимізувати робочі процеси. Це підвищує мотивацію та зменшує стрес, оскільки представляє чітке уявлення про те, що відбувається зараз і в

майбутньому [3]. Це дає користувачеві можливість розбивати великі цілі на маленькі кроки, бачити досягнутий прогрес і належним чином реагувати, коли обставини змінюються. У цьому контексті інтерактивний календар завдань є цифровим інструментом, що поєднує функції щоденника, функції планувальника та можливості нагадування. Він дозволяє IT-фахівцям візуалізувати свій розклад, встановлювати дедлайни для технічних завдань, відстежувати статус їх виконання та отримувати нагадування у зручному та доступному форматі, що є особливо цінним для індивідуальної організації робочого часу та покращення професійної продуктивності [5].

1.3 Постановка завдання на розробку програмної системи

На основі всебічного аналізу існуючих рішень та виявлених прогалин ринку, було чітко окреслено специфічну проблему, яка потребує вирішення в межах даного проєкту. Існуючі інструменти для управління завданнями, попри свою функціональність, не повною мірою задовольняють потреби IT-фахівців у простому, автономному та гнучкому десктопному рішенні для організації їхнього індивідуального робочого процесу.

З огляду на зазначене, проблемою, яку необхідно вирішити в межах даної роботи, є відсутність легкодоступного, інтуїтивно зрозумілого та локально працюючого інструменту для ефективного управління професійними завданнями та їх візуалізації у форматі інтерактивного календаря, що враховуватиме специфіку роботи IT-фахівців.

Метою розробки даної програмної системи є створення десктопного застосунку "Інтерактивний календар завдань", який забезпечуватиме зручне та ефективне планування робочих завдань, контроль дедлайнів та оптимізацію робочого процесу IT-фахівців в автономному режимі.

Для досягнення поставленої мети було визначено наступні завдання розробки:

1. Проаналізувати існуючі рішення.

2. Розробити архітектуру програмної системи, що забезпечить її модульність, легкість супроводу та подальшого розширення.

3. Реалізувати функціонал управління завданнями, що включає можливість створення, перегляду, редагування та видалення записів із зазначенням назви, опису, дати початку та дедлайну, адаптованих для робочих потреб IT-фахівців.

4. Забезпечити інтерактивну візуалізацію завдань на календарі, з використанням кольорових позначень для відображення статусу виконання.

5. Впровадити механізм спливаючих підказок, що надаватимуть короткі відомості про завдання при наведенні курсору на елемент календаря.

6. Реалізувати систему автоматичної генерації системних сповіщень про наближення дедлайнів завдань.

7. Впровадити функціонал персоналізації інтерфейсу, що дозволить користувачу змінювати тему оформлення та кольорову гаму застосунку, враховуючи можливі вподобання професійного середовища.

8. Забезпечити автоматичне збереження та завантаження персональних налаштувань при кожному запуску програми.

9. Реалізувати зручну навігацію календарем між днями, тижнями та місяцями.

10. Додати елемент підтримки мотивації користувача через періодичну демонстрацію випадкових надихаючих фраз, що можуть бути релевантними для підвищення ефективності в роботі.

11. Забезпечити роботу застосунку в автономному режимі, без потреби у постійному підключенні до інтернету для базових функцій.

1.4 Висновок до першого розділу

У даному розділі було проведено комплексний аналіз предметної області управління завданнями та робочим часом, а також здійснено детальний огляд існуючих програмних рішень. Було встановлено, що в умовах сучасного

динамічного IT-середовища ефективне управління часом та робочими завданнями є критично важливим для професійної успішності, тоді як такі проблеми, як прокрастинація та зниження продуктивності, підкреслюють нагальну потребу в дієвих інструментах для структурування роботи.

Детальний порівняльний аналіз популярних програмних продуктів для планування та управління завданнями виявив, що, попри їхню багатофункціональність та широку доступність, більшість з них є хмарними, часто перевантаженими для індивідуального використання, залежними від інтернет-з'єднання та мають обмежені можливості персоналізації. Виявлені прогалини на ринку та незадоволені потреби серед IT-фахівців підкреслили актуальність розробки рішення, яке здатне забезпечити оптимізацію індивідуального робочого процесу. Таким чином, у даному розділі було всебічно обґрунтовано актуальність обраної теми та визначено основні проблеми, які існують у контексті ефективного управління завданнями в IT-сфері. Зібрані та проаналізовані дані слугуватимуть міцною теоретичною та практичною основою для подальшої роботи над програмною системою, що врахує виявлені потреби.

2 ТЕОРЕТИЧНІ ОСНОВИ ТА ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальні концепції та принципи розробки програмного забезпечення

Процес створення будь-якого програмного забезпечення є комплексною діяльністю, що вимагає системного та організованого підходу. Ефективна розробка ґрунтується на застосуванні відповідних методологій та дотриманні певних принципів, які спільно визначають якість, швидкість та економічну ефективність кінцевого продукту. Актуальність проблем оцінювання якості програмного забезпечення постійно зростає в умовах складності сучасних IT-проєктів [7]. Правильний вибір методології є критично важливим для успішної реалізації програмного продукту, оскільки він впливає на всі етапи життєвого циклу ПЗ, від початкового аналізу вимог до розгортання та супроводу.

Життєвий цикл розробки програмного забезпечення – це структурований процес, який описує всі фази створення ПЗ, починаючи від ідеї та закінчуючи виведенням продукту з експлуатації [8]. Існує декілька основних моделей SDLC, кожна з яких має свої архітектурні особливості та найкраще підходить для різних типів проєктів: водоспадна модель, ітеративні та інкрементальні моделі, спіральна модель та гнучкі методології.

Водоспадна модель – одна з найстаріших і найпростіших, передбачає послідовне проходження етапів: аналіз, проєктування, кодування, тестування, розгортання, супровід. Кожен етап завершується повністю перед переходом до наступного, що забезпечує передбачуваність, але знижує гнучкість [9]. Переваги: простота, чітка документація, ефективність при стабільних вимогах. Недоліки: складність внесення змін, виявлення помилок лише на пізніх етапах, відсутність готового продукту до завершення циклу.

Ітеративні та інкрементальні моделі поділяють проєкт на невеликі ітерації, кожна з яких охоплює повний цикл розробки – від аналізу до

тестування. Продукт поступово доповнюється новим функціоналом, що дозволяє рано отримати зворотний зв'язок і швидко реагувати на зміни [8]. Серед переваг – гнучкість, швидке створення прототипів, раннє виявлення помилок. Недолік – ризик неконтрольованого розширення вимог.

Спіральна модель поєднує водоспадний і ітеративний підходи з акцентом на управління ризиками. Кожен цикл охоплює цілі, аналіз ризиків, розробку та планування наступного етапу [8]. Вона добре підходить для великих проєктів, забезпечуючи гнучкість і контроль над ризиками. Основні недоліки – складність управління та високі витрати на оцінку ризиків.

Гнучкі методології (Agile) – це підхід, що базується на ітеративній розробці, адаптивності, тісній співпраці та самоорганізації команди. Замість акценту на документації, Agile зосереджується на робочому продукті та швидкому реагуванні на зміни [10]. Серед переваг – гнучкість, швидкість, якість і залучення команди. Серед недоліків – складність прогнозування термінів і бюджету, потреба у високій дисципліні.

Окрім вибору моделі життєвого циклу, важливо дотримуватись принципів проєктування та кодування. Вони сприяють створенню систем, які легко змінювати, масштабувати та тестувати.

Принцип DRY (Don't Repeat Yourself) закликає уникати дублювання коду. Повторювану логіку варто виносити у функції або модулі для повторного використання. Це зменшує код, спрощує супровід і знижує ймовірність помилок. У календарі задач передбачається реалізація спільних функцій збереження налаштувань і роботи з базою в окремих методах.

Принцип KISS (Keep It Short and Simple) заохочує створення простих рішень без зайвої складності. Це робить код зрозумілішим і зменшує кількість помилок [11]. У календарі завдань планується використовувати прості вбудовані інструменти, які відповідають основним вимогам.

Принципи SOLID – це п’ять основ об'єктно-орієнтованого проєктування, сформульовані Робертом Мартіном. Вони допомагають створювати гнучке, масштабоване та легко підтримуване ПЗ [12]:

- SRP (Single Responsibility) – кожен клас відповідає лише за одну задачу.
- OCP (Open/Closed) – класи мають бути відкритими для розширення, але закритими для змін.
- LSP (Liskov Substitution) – підкласи повинні повністю заміняти базові класи без порушення логіки.
- ISP (Interface Segregation) – краще кілька вузьких інтерфейсів, ніж один загальний.
- DIP (Dependency Inversion) – залежати потрібно від абстракцій, а не конкретних реалізацій

Застосування вищезгаданих моделей життєвого циклу та принципів проєктування є фундаментальним для створення високоякісного програмного забезпечення. Вони дозволяють систематизувати процес розробки, підвищити його ефективність, забезпечити гнучкість у реагуванні на зміни та створити продукт, який буде надійним, легким у супроводі та придатним до подальшого масштабування. Комплексний підхід до вибору методології та дотримання принципів кодування є запорукою успішної розробки, що відповідає сучасним стандартам якості.

2.1.1 Вибір методології розробки та дослідницьких методів для проєкту

Для розробки інтерактивного календаря завдань обрано гнучку ітеративну модель розробки, адаптовану до принципів Agile-методологій. Цей підхід зумовлений кількома ключовими факторами:

- необхідність швидкого прототипування та тестування;
- можливість адаптації до змін;
- ефективність для індивідуальної розробки;
- фокус на цінності для користувача.

Проєкт передбачає швидке створення функціонального продукту для отримання зворотного зв'язку. Ітеративна розробка дозволить випускати версії на ранніх етапах, що критично для апробації та перевірки гіпотез щодо інтерфейсу і функціоналу. Також вимоги можуть коригуватися або розширюватися. Гнучка модель дозволяє легко інтегрувати зміни, оскільки розробка відбувається невеликими циклами [13].

Класичні водоспадні моделі надмірні для невеликих проєктів, тому Agile-принципи (самоорганізація, адаптивність, фокус на працюючому ПЗ) ідеально підходять для таких умов [10]. Ітеративний підхід дозволить зосередитися на розробці найважливіших функцій першими, постійно перевіряючи їхню цінність для кінцевого користувача. Це забезпечить відповідність продукту потребам аудиторії.

Процес розробки включатиме циклічні ітерації: планування (цілі, пріоритети, обсяг робіт), аналіз та проєктування (деталізація вимог, архітектурні рішення, дизайн), реалізація (кодування, впровадження функціоналу із застосуванням DRY, KISS, SOLID), тестування (перевірка працездатності, виявлення помилок, апробація користувачами для збору відгуків [14]), розгортання (підготовка функціоналу для демонстрації), супровід та розвиток (виправлення помилок, оновлення, додавання нових можливостей). Такий підхід дозволить оперативно реагувати на виклики та забезпечити високу якість і відповідність функціоналу очікуванням користувачів.

У процесі розробки також буде застосовано комплекс наукових методів дослідження.

Системний аналіз є ключовим методом для розуміння складної структури проєкту та її декомпозиції. Він передбачає розгляд об'єкта як єдиної системи з взаємопов'язаних елементів. Це дозволить чітко визначити функціональні модулі (GUI, бізнес-логіка, доступ до даних), їх взаємозв'язки та спроектувати ефективну архітектуру [15].

Порівняльний аналіз застосовуватиметься для дослідження існуючих аналогів та вибору технологій. Метод передбачає зіставлення характеристик

різних об'єктів (календарів-органайзерів, технологій) для виявлення переваг та недоліків [16]. Це дозволить обґрунтувати розробку власного рішення та зробити усвідомлений вибір технологічних інструментів.

Емпіричне тестування (апробація) буде використано для об'єктивної оцінки зручності, ефективності та задоволеності користувачів календарем. Метод передбачає збирання даних шляхом спостереження або опитувань. Апробація здійснюватиметься через онлайн-опитування для збору якісних та кількісних відгуків від цільової аудиторії – ІТ-фахівців [14].

Метод проєктування використовуватиметься для розробки архітектури ПЗ, дизайну інтерфейсу та деталізації логіки компонентів. Включаючи етапи від загального концептуального до детального проєктування модулів. Принципи дизайн-мислення [17] будуть використані для створення інтуїтивного інтерфейсу, а принципи модульності [18] для забезпечення гнучкості системи.

2.2 Аналіз та формалізація вимог до системи управління завданнями

Ефективна розробка програмного забезпечення неможлива без чіткого розуміння того, що саме має виконувати система та яким критеріям якості вона повинна відповідати. Вимоги є основою для проєктування, реалізації та тестування, забезпечуючи, що кінцевий продукт задовольнятиме потреби користувачів та бізнес-цілі. Вони поділяються на функціональні та нефункціональні, кожен тип яких відіграє критично важливу роль для успішної розробки.

Функціональні вимоги визначають конкретні можливості, які система повинна надавати користувачам, і описують її поведінку у відповідь на дії користувача або зовнішні події. Для інтерактивного календаря завдань ці вимоги охоплюють ключові аспекти взаємодії, зберігання інформації та персоналізації.

Основним функціоналом системи передбачається повноцінне управління завданнями, що включає створення нових записів із зазначенням назви, опису та дати початку дедлайну; перегляд та редагування та видалення наявних завдань. Всі завдання повинні візуалізуватися на інтерактивному календарі, де використовуються кольорові позначення для відображення статусу. При наведенні курсору на елемент календаря має з’являтися спливаюча підказка з короткими відомостями про відповідне завдання.

Ще однією важливою функцією є автоматична генерація системних сповіщень про наближення дедлайну. Крім того, передбачено можливість персоналізації інтерфейсу, зокрема зміну теми оформлення та кольорової гами, що дозволяє адаптувати вигляд календаря відповідно до індивідуальних вподобань.

Для підтримки мотивації користувача система періодично демонструватиме випадкові надихаючі фрази. Інтерфейс повинен забезпечувати зручну навігацію між днями, тижнями та місяцями, дозволяючи легко знаходити потрібні записи. Усі персональні налаштування користувача, включаючи вибрану тему, кольори та параметри сповіщень, повинні зберігатися автоматично та завантажуватись при кожному запуску програми, забезпечуючи безперервність користувацького досвіду. На рисунку 2.1 зображено діаграму варіантів використання (Use Case Diagram).

Цей тип діаграм є одним з основних елементів Unified Modeling Language і використовується для моделювання функціональних вимог системи, відображаючи взаємодію користувачів (акторів) із системою [18]

Окрім функціональних вимог, не менш важливим є визначення нефункціональних характеристик системи, таких як продуктивність, надійність, зручність використання та безпека. Забезпечення інформаційної безпеки у комп'ютерних системах є критично важливим для захисту конфіденційності та цілісності даних користувачів, особливо в контексті зберігання особистих завдань та планів [19].

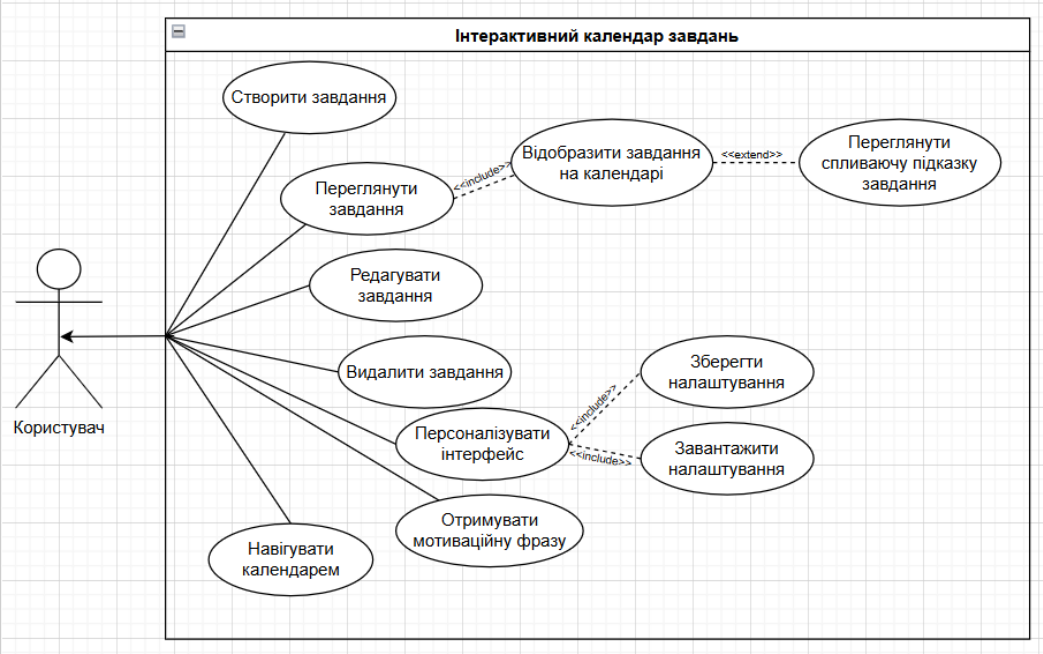

Рисунок 2.1 – Діаграма варіантів використання

Нефункціональні вимоги встановлюють якісні характеристики системи, її обмеження та критерії ефективності, які не стосуються безпосередньо конкретного функціоналу, але мають вирішальний вплив на загальну стабільність, зручність використання та можливість подальшого розвитку.

Насамперед система повинна демонструвати високу продуктивність: забезпечувати швидкий запуск, мінімальне споживання оперативної пам’яті та процесорного часу, а також миттєву реакцію на дії користувача, зокрема під час перемикання між календарними періодами чи додавання нових записів. Надійність реалізується через стабільну роботу без збоїв або аварійного завершення, з коректною обробкою помилок вводу й збереженням даних навіть у непередбачених ситуаціях.

Ключовою вимогою є зручність використання: інтерфейс повинен бути інтуїтивно зрозумілим, з простою навігацією та доступом до основних функцій, що дозволить працювати з календарем навіть користувачам без технічної підготовки.

Важливим аспектом є портативність – система має підтримувати стабільну роботу на різних операційних системах, таких як Windows, macOS і

Linux, забезпечуючи кросплатформенність. Уся інформація про завдання та персональні налаштування зберігається виключно локально на пристрої користувача, без передачі даних на зовнішні сервери, що гарантує високий рівень приватності та безпеки.

Архітектура програми повинна бути модульною та масштабованою, що полегшить її супровід, оновлення і розширення функціональності у майбутньому. Окрім того, календар має працювати в автономному режимі, без потреби у постійному підключенні до інтернету чи використанні зовнішніх онлайн-сервісів для базових можливостей

2.3 Огляд та обґрунтування вибору технологій для розробки десктопного застосунку

Вибір технологічного стеку для десктопного застосунку є критичним рішенням, що впливає на процес розробки, продуктивність, зручність супроводу та мультиплатформність продукту. Сучасний ринок пропонує широкий спектр фреймворків та бібліотек, які можна умовно розділити на кілька категорій за їхньою архітектурою та підходом до розробки. Одним із популярних підходів є використання веб-технологій (HTML, CSS, JavaScript) для створення десктопних застосунків. У цій категорії провідним фреймворком є Electron, який поєднує рушій Chromium для рендерингу інтерфейсу та Node.js для роботи з системними функціями [20]. Переваги Electron включають сумісність з різними операційними системами (Windows, macOS, Linux), швидку розробку завдяки знайомим веб-технологіям і широку спільноту. Однак недоліки – велике споживання ресурсів та великий розмір виконуваних файлів [21].

Більш низькорівневий контроль над інтерфейсом і системою надають фреймворки типу Qt. Це потужний інструмент для створення продуктивних та привабливих застосунків [22]. Для Python існують відповідники Qt – PyQt та

PySide [23]. Перевагами є висока продуктивність, нативний інтерфейс, багатий функціонал. Недоліки: складність вивчення, більші розміри проєктів.

.NET-фреймворки, зокрема Windows Forms, WPF та .NET MAUI, добре інтегруються в екосистему Microsoft і мають багатий набір інструментів [24]. Вони особливо ефективні в розробці для Windows, хоча .NET MAUI також підтримує мультиплатформну розробку [25]. Недоліки включають залежність від .NET Runtime і історичну орієнтацію на Windows.

Для Python доступно кілька бібліотек для створення GUI:

- Tkinter – стандартна бібліотека для створення простих графічних інтерфейсів. Має переваги у вигляді простоти, сумісності з різними ОС та швидкого прототипування, хоча її вигляд віджетів дещо базовий.
- Kivy – кросплатформенний фреймворк для створення мультитач-додатків з унікальним дизайном [26]. Підтримує Android, iOS, Windows, Linux [27]. Недоліки: не використовує нативні віджети ОС, що може призвести до нестандартного вигляду, менша спільнота порівняно з Tkinter чи PyQt/PySide, інколи складність у налагодженні для новачків [28].
- GTK/PyGObject – особливо популярні у Linux-середовищі, з активною спільнотою [29], але менш зручні на Windows та macOS [30].

Для розробки інтерактивного календаря завдань був обраний наступний стек технологій:

- Python – як основна мова розробки;
- Tkinter – для створення графічного інтерфейсу;
- SQLite – для зберігання даних локально;
- Plyer – для реалізації системних сповіщень.

Цей вибір обумовлений як функціональними, так і нефункціональними вимогами проєкту.

Python обрано через простоту синтаксису, швидкість розробки, читабельність коду та потужну екосистему бібліотек [31]. Його

кросплатформенність дозволяє розгортати застосунок на різних операційних системах без значних змін у коді.

Tkinter – це вбудована бібліотека Python, яка не потребує окремої установки і дозволяє швидко реалізувати інтуїтивний GUI із використанням кнопок, канвасів, форм введення тощо. Можливості кастомізації через ttk і Canvas дозволяють створити гнучкий інтерфейс, достатній для цілей проєкту.

SQLite забезпечує ефективне та просте безсерверне зберігання даних у форматі локального файлу. Це ідеальне рішення для автономної десктопної програми, оскільки не вимагає налаштування сервера або зовнішнього підключення.

Plyer дозволяє надсилати нативні системні сповіщення, що значно покращує користувацький досвід. Бібліотека підтримує всі основні платформи, не потребує складної конфігурації й легко інтегрується в Python-застосунок.

Таким чином, обраний стек технологій повністю задовольняє вимоги проєкту до кросплатформенності, швидкої розробки, простоти розгортання та ефективності. Це дозволяє реалізувати функціональний, зручний та стабільний календар завдань, адаптований до потреб користувача.

2.4 Висновок до другого розділу

У другому розділі було досліджено загальні підходи до розробки ПЗ, де обґрунтовано вибір гнучкої ітеративної моделі, заснованої на Agile-принципах. Такий підхід дозволив забезпечити швидке прототипування, можливість адаптації до змін і ефективну реалізацію в умовах індивідуальної розробки.

На основі системного та порівняльного аналізу були визначені як функціональні, так і нефункціональні вимоги до системи. Сформульовані вимоги охоплюють ключові очікування користувача: зручність управління завданнями, персоналізацію інтерфейсу, швидкодію, кросплатформеність, автономність та приватність. Це забезпечило чітке бачення функціоналу, до якого має прагнути реалізація.

Було проведено огляд сучасних технологій для розробки десктопних застосунків, після чого обґрунтовано вибір основного стеку: Python (як основна мова розробки), Tkinter (для створення графічного інтерфейсу), SQLite (для зберігання даних) та Plyer (для реалізації сповіщень). Обрані інструменти відповідають вимогам щодо простоти розгортання, кросплатформенності, легкості супроводу та низького порогу входження для розробника.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ

3.1 Архітектура та реалізація програмної системи

Реалізація програмної системи "Інтерактивний календар завдань" здійснювалася з використанням модульного підходу, що передбачає чітке розділення відповідальності між компонентами. Побудова чіткої та функціонуючої структури програмного ресурсу є ключовим для його ефективності та супроводжуваності [32]. Такий підхід має на меті спрощення процесу розробки, тестування, подальшої підтримки та потенційного масштабування застосунку. Архітектура системи базується на принципах багатошарового дизайну, що дозволяє відокремити рівень представлення даних від бізнес-логіки та механізмів зберігання даних.

Застосунок проєктується як автономна десктопна програма, що функціонуватиме на комп'ютері користувача. Загальна архітектура системи прагне дотримуватися принципів моделі Модель-Вид-Контролер (MVC), що забезпечує логічне відокремлення даних (модель), користувацького інтерфейсу (вид) та обробки користувацьких запитів (контролер). Одним з ключових аспектів успішної реалізації є ефективна візуалізація завдань на інтерактивному календарі, що базується на принципах побудови інформаційних технологій візуалізації ресурсів [33].

У контексті даної розробки, система поділяється на наступні основні шари: рівень представлення, рівень бізнес-логіки та рівень доступу до даних.

Рівень представлення відповідає за графічний інтерфейс користувача (GUI). Його основна функція полягає у відображенні інформації користувачеві та прийомі його вводу.

Рівень бізнес-логіки є центральним компонентом, який обробляє взаємодії користувача, виконує обчислення, керує потоком даних між рівнем

представлення та рівнем доступу до даних, а також ініціює системні події (наприклад, сповіщення).

Рівень доступу до даних відповідає виключно за взаємодію зі сховищем даних (базою даних), забезпечуючи механізми для зберігання, отримання, оновлення та видалення інформації.

На рисунку 3.1 представлено організацію файлів та папок проєкту інтерактивного календаря завдань.

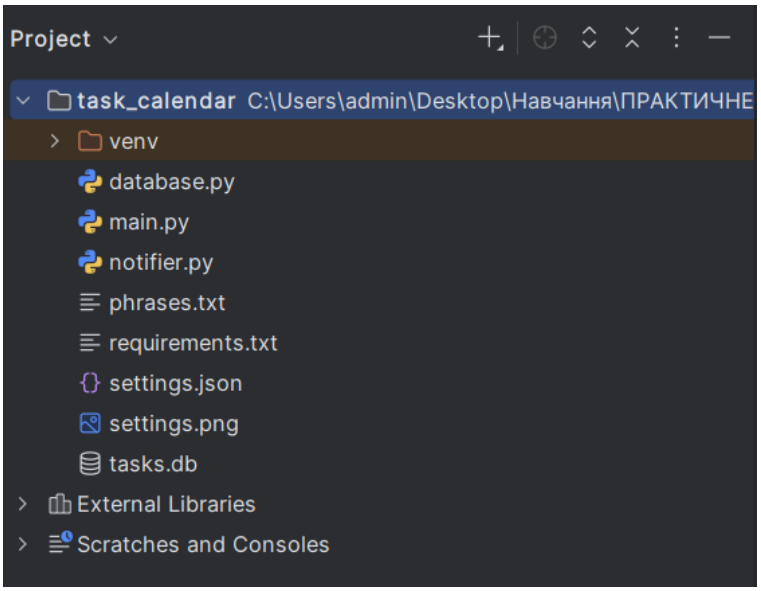

Рисунок 3.1 – Структура файлів проєкту інтерактивного календаря завдань

Елементи проєкту організовані таким чином, щоб відображати модульність системи:

- main.py – головний компонент, що містить основний клас застосунку, логіку ініціалізації GUI та інтеграції інших частин системи;
- database.py – відповідає за всі операції з базою даних;
- notifier.py – реалізовує функціонал системних сповіщень;
- phrases.txt – текстовий документ для зберігання мотиваційних фраз;
- requirements.txt – текстовий перелік усіх зовнішніх бібліотек Python та їх версій, необхідних для коректної роботи застосунку;

- settings.json – JSON-документ для збереження користувацьких налаштувань інтерфейсу;
- settings.png – зображення, яке відповідає за налаштування;
- tasks.db – файл бази даних SQLite.

Додаткові папки, такі як .idea, __pycache__ та venv, є службовими і використовуються середовищем розробки та інтерпретатором Python..

3.1.1 Реалізація рівня користувацького інтерфейсу (GUI)

Графічний інтерфейс користувача у розробленому застосунку реалізовано за допомогою бібліотеки Tkinter, яка входить до стандартної бібліотеки Python. Основний візуальний компонент – це кастомізований календар, який дозволяє переглядати, додавати, редагувати та видаляти завдання.

Під час ініціалізації календаря створюються віджети керування місяцем і роком, кнопки додавання подій, а також область для графічного відображення днів місяця (див. лістинг 3.1).

Лістинг 3.1 – Ініціалізація елементів керування місяцем, роком і кнопки додавання подій у календарі

```
self.month_menu = ttk.Combobox(left_frame, values=months,
textvariable=self.month_var, state="readonly")
self.year_menu = ttk.Combobox(left_frame, values=years,
textvariable=self.year_var, state="readonly")
add_btn = tk.Button(right_frame, text="➕ Додати",
command=self.on_add_task)
```

Кожен день календаря зображено як прямокутник із числом. Якщо на день заплановані завдання, відображається кольоровий індикатор (зелений, синій, червоний або помаранчевий) із підказкою про стан завдань (див. лістинг 3.2).

Лістинг 3.2 – Відображення дня календаря з прямокутником, числом і підказкою про стан завдань

```
rect_id = self.canvas.create_rectangle(x, y, x+50, y+50,
fill=self.cell_color, outline="gray")
text_id = self.canvas.create_text(x+25, y+25, text=str(day),
font=self.font, fill=self.text_color)
      TooltipCanvas(self.canvas, oval_id, tooltip_text)
```

Для зручності додано можливість редагування, позначення як виконаного та видалення завдань безпосередньо зі списку. Всі події місяця виводяться у скролюваній панелі (див. лістинг 3.3)

Лістинг 3.3 – Кнопки редагування, позначення як виконаного та видалення завдань у скролюваній панелі

```
btn = tk.Button(frame, text=btn_text, command=lambda tid=task_id:
self.show_event_details(tid))
delete_btn = tk.Button(frame, text="❌",
command=partial(self.delete_task_with_confirm, task_id))
mark_done_btn = tk.Button(frame, text="✅",
command=partial(self.toggle_task_status_by_id, task_id, is_done))
```

Користувач також має змогу налаштовувати тему оформлення через відповідне вікно налаштувань. Підтримується як вибір попередньо заданої теми (світла/темна), так і користувацьке налаштування кольорів елементів (див. лістинг 3.4).

Лістинг 3.4 – Реалізація вибору теми оформлення та налаштування кольорів у вікні налаштувань

```
theme_choice = ttk.Combobox(settings_win, values=["Світла",
"Темна", "Користувацька"])
self.bg_color_btn = tk.Button(settings_win,
command=choose_bg_color, bg=selected_bg_color.get())
```

Таким чином, реалізація GUI забезпечує інтуїтивно зрозумілий інтерфейс із візуалізацією завдань та інструментами керування подіями календаря.

3.1.2 Реалізація рівня бізнес-логіки

Рівень бізнес-логіки у застосунку відповідає за обробку основних сценаріїв роботи з даними – створення, збереження, редагування, видалення та нагадування про завдання. Цей рівень є посередником між користувацьким інтерфейсом і базою даних, а також реалізовує автоматичну логіку нагадувань.

Ключові функції для роботи з завданнями розміщені в модулі database.py. Наприклад, функція додавання нового завдання виглядає так (див. лістинг 3.5).

Лістинг 3.5 – Функція додавання нового завдання до бази даних

```
def add_task(title, description, deadline):
     conn = sqlite3.connect('tasks.db')
     cursor = conn.cursor()
     cursor.execute('INSERT INTO tasks (title, description,
     deadline) VALUES (?, ?, ?)',
               (title, description, deadline))
     conn.commit()
     conn.close()
```

Окремі функції дозволяють отримувати завдання для конкретної дати або всього місяця, а також оновлювати їх статус або видаляти при потребі. Таке розділення забезпечує гнучкість і масштабованість бізнес-логіки.

Клас TaskCalendarApp (у main.py) зв’язує графічний інтерфейс із бізнес-логікою. Наприклад, метод add_task_dialog() приймає введення користувача та викликає add_task(...) для збереження даних (див. лістинг 3.6).

Лістинг 3.6 – Метод add_task_dialog() для додавання завдання

```
def add_task_dialog(self):
     title = simpledialog.askstring("Назва", "Введіть назву
завдання:")
     ...
     add_task(title, desc, deadline)
     self.load_tasks()
```

Таким чином, бізнес-логіка застосунку реалізована окремими модулями та функціями, що забезпечує прозорість, розширюваність і можливість повторного використання коду.

3.1.3 Реалізація рівня доступу до даних та системи сповіщень

У розробленому застосунку доступ до бази даних здійснюється через модуль database.py. Для збереження та обробки інформації про завдання використовується вбудована база даних SQLite, яка є простою у використанні та не потребує окремого встановлення сервера.

При запуску програми викликається функція init_db(), яка створює таблицю «tasks», якщо вона ще не існує (див. лістинг 3.7).

Лістинг 3.7 – Функція ініціалізації бази даних та створення таблиці завдань

```
def init_db():
     conn = sqlite3.connect('tasks.db')
     cursor = conn.cursor()
     cursor.execute('''
          CREATE TABLE IF NOT EXISTS tasks (
               id INTEGER PRIMARY KEY AUTOINCREMENT,
               title TEXT NOT NULL,
               description TEXT,
               deadline TEXT NOT NULL,
               status INTEGER DEFAULT 0
          )
     ''')
     conn.commit()
     conn.close()
```

Створена таблиця «tasks» має чітко визначені поля для зберігання всіх необхідних даних про завдання, включаючи унікальний ідентифікатор, назву, опис, дедлайн та статус виконання. Її візуальне представлення, що підтверджує структуру та типи даних полів, зображено на рисунку 3.2.

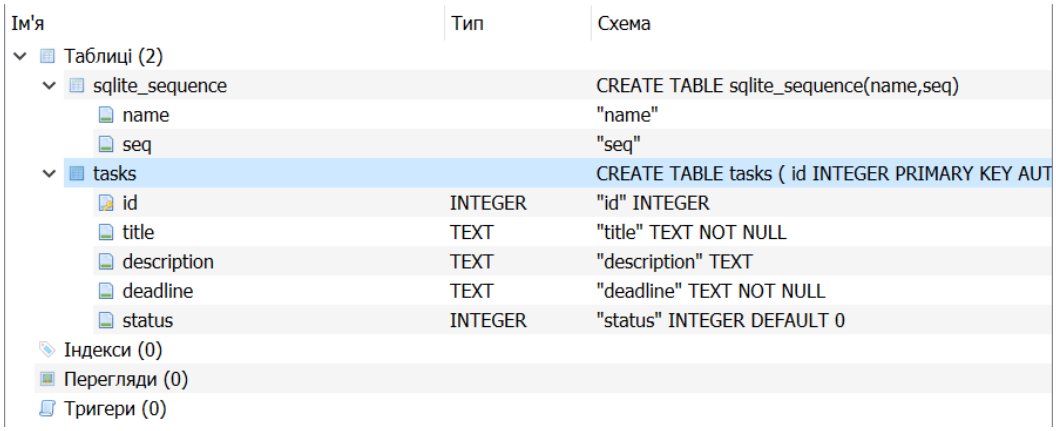

Рисунок 3.2 – Структура таблиці «tasks» у базі даних SQLite

Для взаємодії з даними реалізовано окремі функції:

- add_task(...) – додавання нового завдання;
- get_tasks_for_date(...) – отримання завдань на вказану дату;
- get_tasks_for_month(...) – отримання завдань на місяць;
- update_task_status(...) – зміна статусу завдання;
- delete_task(...) – видалення завдання.

Кожна функція використовує коротке підключення до бази, що гарантує надійність та простоту відлагодження (див. лістинг 3.8).

Лістинг 3.8 – Функція видалення завдання

```
def delete_task(task_id):
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('DELETE FROM tasks WHERE id = ?',(task_id,))
    conn.commit()
    conn.close()
```

Система сповіщень реалізована у модулі notifier.py з використанням бібліотеки plyer.notification, яка дозволяє виводити повідомлення у вигляді системних pop-up нагадувань.

Функція check_for_notifications() перевіряє всі активні завдання зі статусом "не виконано" і порівнює їхній дедлайн з поточним часом. Якщо

завдання має бути виконане протягом наступної години – викликається повідомлення (див. лістинг 3.9).

Лістинг 3.9 – Функція перевірки завдань для системних сповіщень

```
def check_for_notifications():
     now = datetime.now()
     next_hour = now + timedelta(hours=1)
     ...
               if now <= task_time <= next_hour:
                    notify(title)
```

У лістингу 3.10 представлено функцію надсилання системного сповіщення.

Лістинг 3.10 – Функція надсилання системного сповіщення

```
def notify(message):
     notification.notify(
          title='Нагадування про завдання',
          message=message,
          timeout=10
     )
```

У головному класі застосунку (TaskCalendarApp) налаштований періодичний виклик перевірки сповіщень кожну хвилину за допомогою after():

```
def run_notifications(self):
     check_for_notifications()
     self.root.after(60000, self.run_notifications)
```

Таким чином, рівень доступу до даних та система сповіщень інтегровані в архітектуру застосунку, що забезпечує надійне збереження завдань і своєчасне інформування користувача про важливі події.

3.2 Демонстрація функціональних можливостей та користувацького інтерфейсу

Після детального розгляду внутрішньої архітектури системи та технічних аспектів реалізації її ключових компонентів, цей підрозділ присвячений демонстрації кінцевого продукту з позиції користувача. Тут представлено, як реалізовані функціональні можливості відображаються у візуальній формі, демонструючи користувацький інтерфейс інтерактивного календаря завдань та інтерактивні елементи, що забезпечують взаємодію з системою.

При запуску програми користувач бачить головне вікно, яке є центральним елементом для взаємодії. Воно містить інтерактивний календар, панелі для відображення завдань на вибраний день та всіх подій за місяць, а також елементи керування для навігації та управління завданнями. У верхній частині вікна також відображається мотиваційна фраза, що динамічно оновлюється.

На рисунку 3.3 зображено головне вікно інтерактивного календаря завдань з основними елементами інтерфейсу.

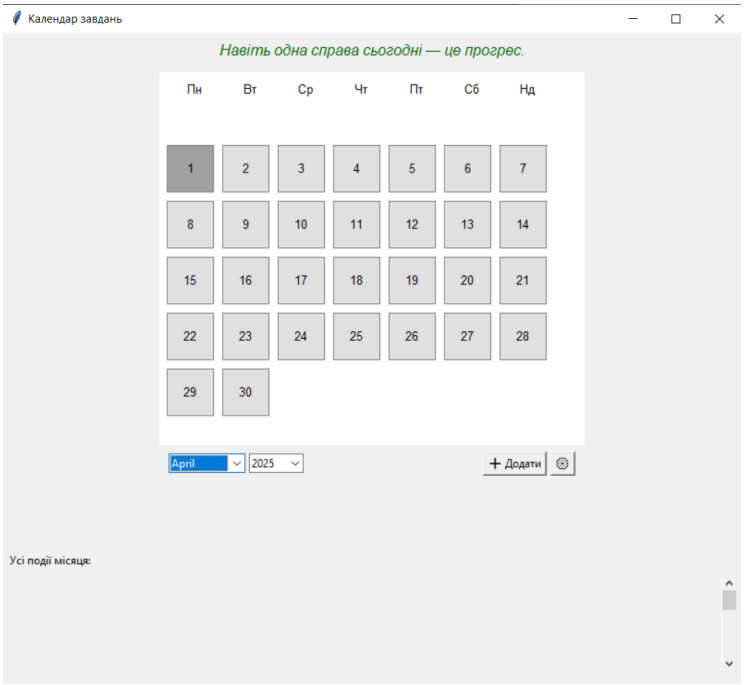

Рисунок 3.3 – Головне вікно інтерактивного календаря завдань

Календарна сітка є не просто відображенням дат, а потужним візуальним інструментом. Кожен день на календарі, що містить завдання, позначається кольоровим маркером – невеликим кружечком. Ця кольорова індикація надає миттєву інформацію про стан завдань на конкретну дату:

- зелений маркер вказує на те, що всі завдання на цей день успішно виконані;
- синій маркер свідчить про наявність актуальних (невиконаних) завдань, які, однак, ще не прострочені;
- помаранчевий маркер сигналізує про змішаний стан: на цей день є як ще актуальні, так і прострочені завдання;
- червоний маркер є попередженням, що на цей день присутні прострочені завдання.

Ця система індикації дозволяє користувачу швидко оцінити завантаженість та пріоритети, звернувши увагу на проблемні дати. При наведенні курсору на день з маркером з'являється детальна спливаюча підказка, яка надає точну інформацію про кількість виконаних, прострочених та актуальних завдань на цей день.

На рисунку 3.4 продемонстровано кольорову індикацію статусів завдань на календарній сітці та приклад спливаючої підказки.

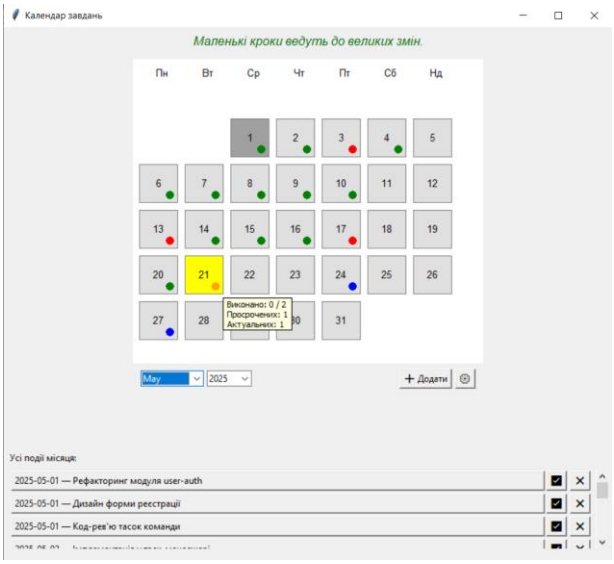

Рисунок 3.4 – Кольорова індикація завдань та деталізована підказка

При кожному запуску програми користувач отримує швидкий огляд свого розкладу на поточний день. З'являється інформативне спливаюче вікно, яке групує завдання на сьогодні за їхнім статусом: «Прострочені», «Актуальні»" та «Виконані». Це дозволяє одразу зосередитися на найважливіших або термінових справах.

На рисунку 3.5 показано спливаюче вікно з нагадуванням про завдання на поточний день при запуску програми.

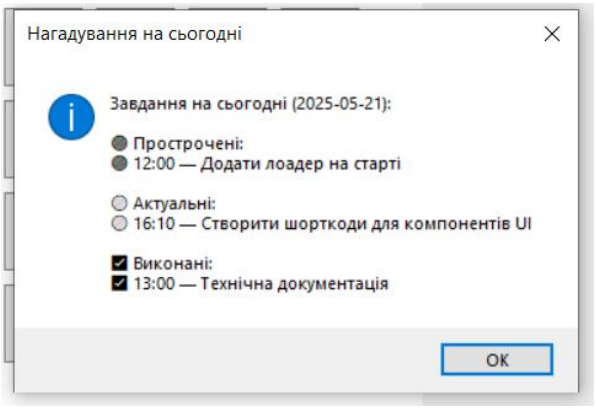

Рисунок 3.5 – Спливаюче вікно «Нагадування на сьогодні»

На рисунку 3.6 представлено діалогове вікно для додавання нового завдання, де користувач вводить необхідну інформацію.

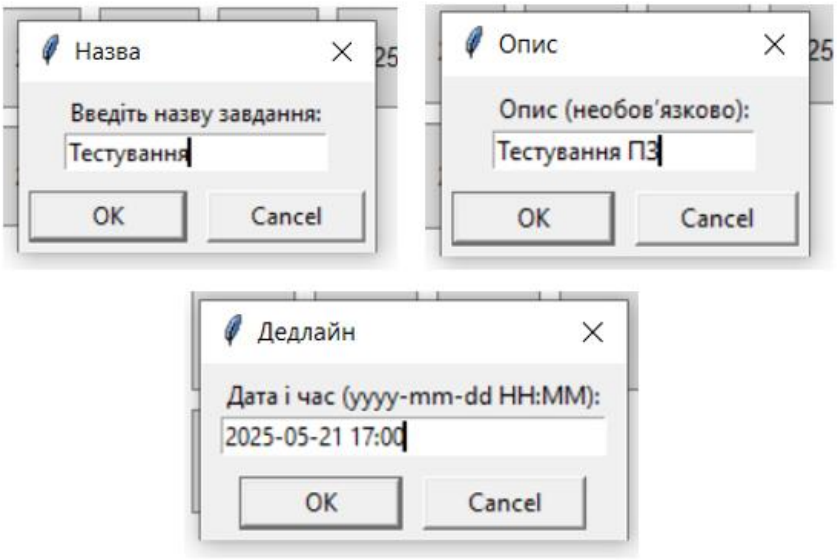

Рисунок 3.6 – Діалогове вікно «Додати завдання»

Після додавання завдання відображається у списку завдань на вибрану дату. Кожне завдання у списку супроводжується окремою іконкою статусу (галочка для виконаних, пісочний годинник для невиконаних) та кнопками швидкого доступу. Користувач може редагувати завдання, змінюючи його назву, опис або дедлайн, або видалити його з підтвердженням, щоб уникнути випадкової втрати даних. Також доступна функція зміни статусу завдання одним кліком, що миттєво оновлює візуальну індикацію як у списку, так і на календарній сітці.

На рисунку 3.7 зображено список завдань на вибраний день з можливостями редагування та зміни статусу.

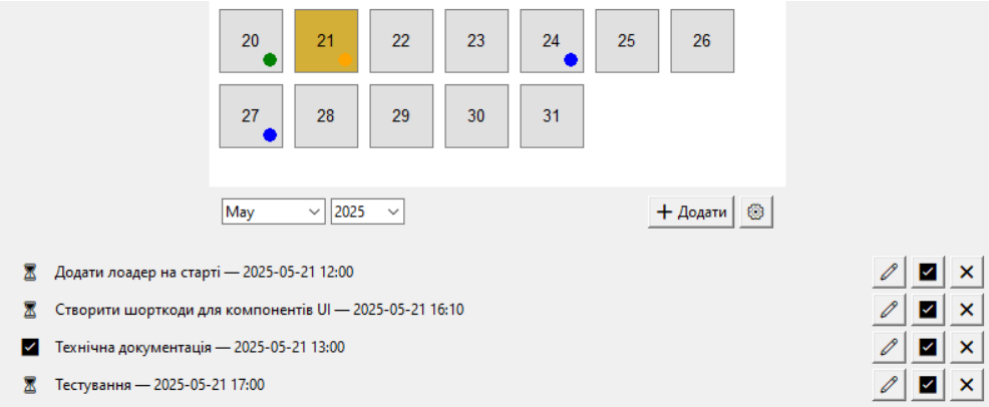

Рисунок 3.7 – Список завдань на вибраний день

Крім завдань на конкретний день, програма надає можливість перегляду всіх подій місяця у окремому скролюваному списку. Це дозволяє користувачу мати повний огляд своїх зобов'язань за поточний період, що є корисним для довгострокового планування (див. рисунок 3.8).

Рисунок 3.8 – Список завдань усіх подій місяця

Однією з переваг розробленого календаря є його висока гнучкість у налаштуванні зовнішнього вигляду. Користувач може адаптувати інтерфейс до своїх візуальних уподобань через вікно налаштувань.

Доступні три попередньо визначені теми оформлення: «Світла» (стандартний світлий фон), «Темна» (оптимізована для роботи в умовах низького освітлення) та «Користувацька» (див. рисунок 3.9).

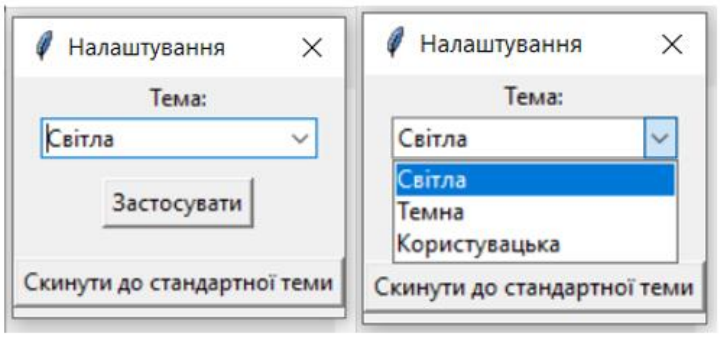

Рисунок 3.9 – Налаштування теми інтерактивного календаря

На рисунку 3.10 показано головне вікно програми у «Світлій» темі оформлення.

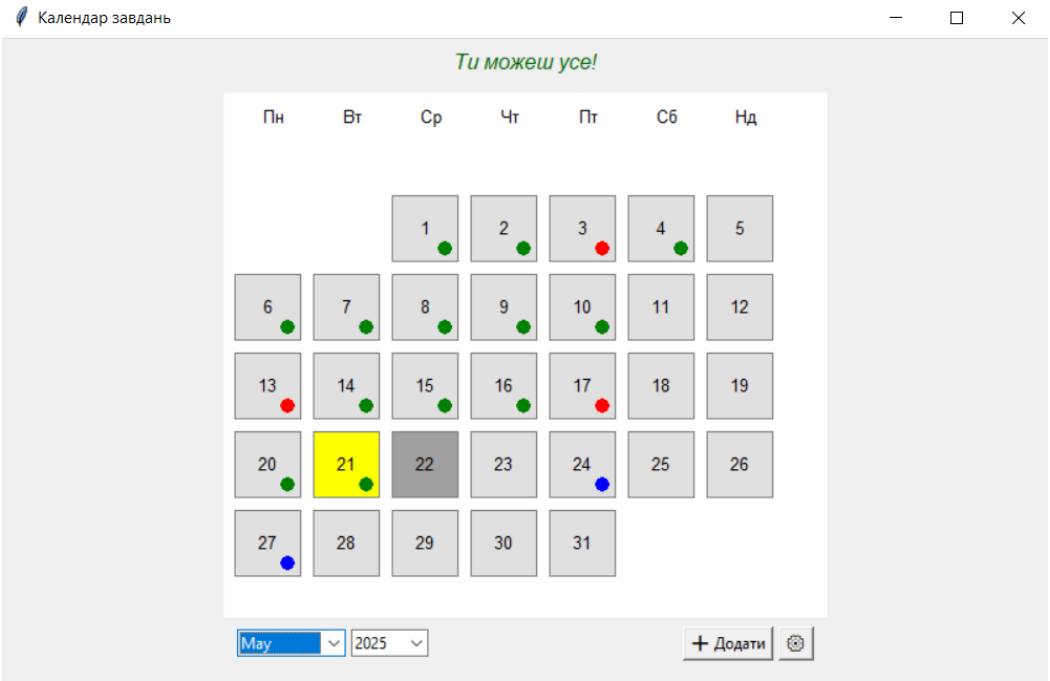

Рисунок 3.10 – Головне вікно у «Світлій» темі

На рисунку 3.11 представлено головне вікно програми у "Темній" темі оформлення, яка забезпечує зниження навантаження на очі в умовах недостатнього освітлення та є більш комфортною для тривалої роботи ввечері або вночі.

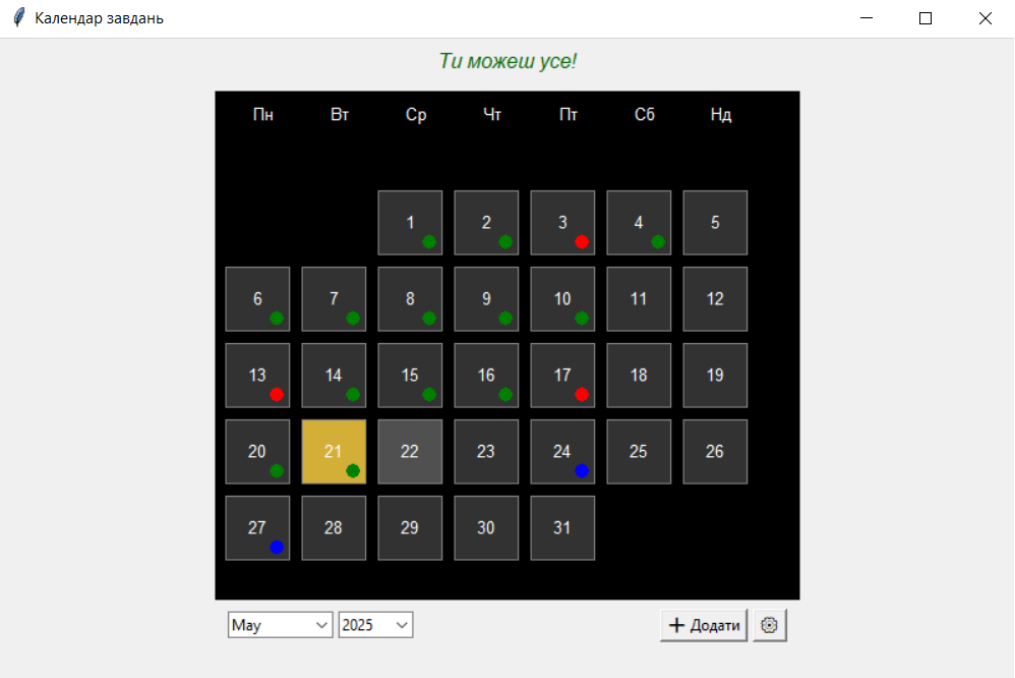

Рисунок 3.11 – Головне вікно у «Темній» темі

У режимі «Користувацька» тема програма надає повний контроль над палітрою кольорів. Користувач може самостійно обрати колір фону програми, колір комірок календаря та колір тексту за допомогою вбудованого діалогу вибору кольорів. Це дозволяє створити унікальний та повністю персоналізований візуальний стиль, що значно покращує досвід взаємодії з додатком. Всі зроблені налаштування автоматично зберігаються і застосовуються при наступних запусках програми.

На рисунку 3.12 зображено приклад обраних кольорів тексту, фону та комірок.

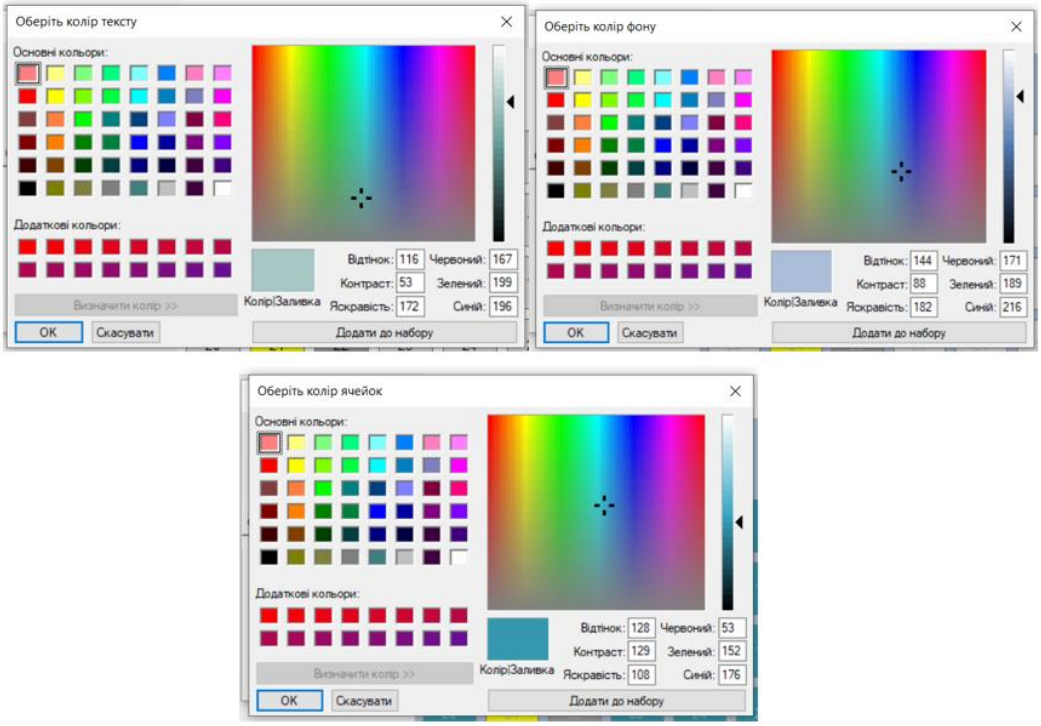

Рисунок 3.12 – Налаштування кольорів у «Користувацькій» темі

На рисунку 3.13 представлено результат головного вікна програми у «Користувацькій» темі з індивідуально обраними кольорами.

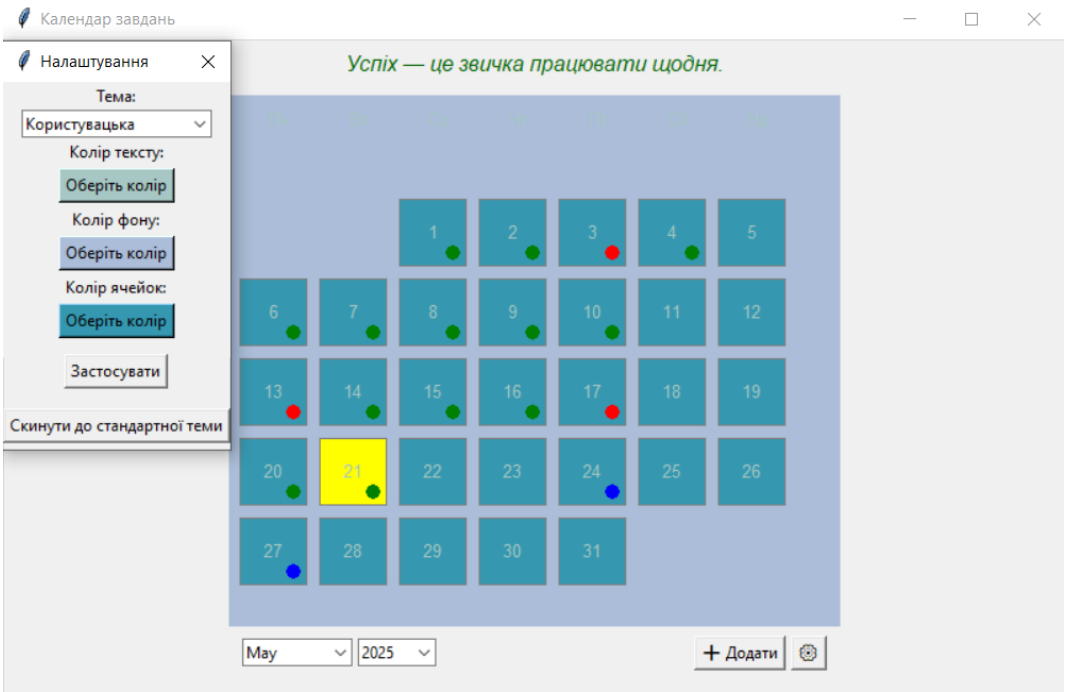

Рисунок 3.13 – Головне вікно у «Користувацькій» темі

Для підтримки позитивного настрою та підвищення продуктивності, у верхній частині головного вікна програми інтегрований рядок з випадковими мотиваційними фразами. Ці фрази періодично оновлюються, додаючи елемент психологічної підтримки та натхнення протягом робочого дня. Важливою особливістю є те, що ці фрази завантажуються з простого текстового файлу phrases.txt, розташованого у кореневій директорії проєкту.

На рисунку 3.14 зображено текстовий документ з мотиваційними фразами, які відображаються на головному екрані інтерактивного календаря.

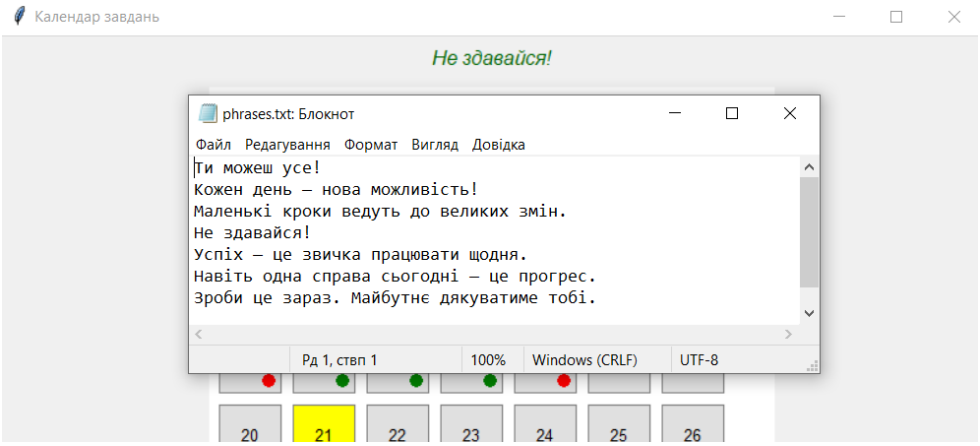

Рисунок 3.14 – Головне вікно з мотиваційною фразою

Це надає користувачу можливість легко додавати власні мотиваційні цитати або редагувати існуючі, адаптуючи контент під особисті вподобання без необхідності втручання в код програми. бебе

Для повноцінної оцінки та підтвердження переваг розробленої системи, а також для візуалізації її місця серед існуючих аналогів, нижче представлена розширена порівняльна таблиця, яка узагальнює ключові характеристики інтерактивного календаря завдань порівняно з іншими рішеннями на ринку.

Таблиця 3.1 – Порівняння функціональних можливостей існуючих рішень та розробленого інтерактивного календаря завдань

Категорія характеристик	**Trello/ ClickUp**	**Google Calendar/Todoist/MS To Do**	**Розроблений інтерактивний календар завдань**
Призначення	Командне управління проєктами, співпраця	Персональне планування, синхронізація	Індивідуальне планування, оптимізація робочого процесу ІТ-фахівця
Тип застосунку	Веб-орієнтований, хмарний	Хмарний, кросплатформенний (веб/мобільний)	Десктопний, автономний
Авторизація	Обов’язкова	Обов’язкова	Не вимагається
Конфіденційність даних	Залежить від політики провайдера	Залежить від політики провайдера	Висока (дані на ПК)
Вартість	Часто платні підписки	Є безкоштовні та платні версії	Безкоштовно
Гнучкість налаштування GUI	Обмежена (стандартні теми)	Обмежена (стандартні теми)	Висока (теми, кольори)
Візуалізація завдань на календарі	Залежить від інструменту, часто менш деталізована	Стандартна (відображення подій, без агрегованих статусів)	Детальна (кольорові маркери статусу, tooltips)
Складність використання	Висока	Середня	Низька (інтуїтивний)
Вимоги до ресурсів	Залежить від браузера/додат-ку, інколи високі	Середні	Низькі
Підтримка мотиваційних елементів	Відсутня	Відсутня	Так (персоналізовані фрази)

Ця таблиця демонструє, що розроблений інтерактивний календар завдань успішно заповнює існуючі прогалини на ринку, пропонуючи автономне та конфіденційне рішення, що відповідає специфічним потребам ІТ-фахівців,

підвищуючи їхню продуктивність за рахунок інтуїтивного інтерфейсу, детальної візуалізації та вбудованих мотиваційних елементів.

3.3 Апробація та тестування системи

Для об'єктивної оцінки зручності, ефективності та задоволеності користувачів розробленим інтерактивним календарем завдань було проведено апробацію системи серед цільової аудиторії – IT-фахівців. Метою апробації було збирання якісних та кількісних відгуків, що дозволили б виявити сильні сторони застосунку, а також потенційні напрямки для подальшого покращення. Апробація проводилася шляхом поширення онлайн-опитування через Google Forms серед групи IT-спеціалістів з різним досвідом роботи. Учасникам було запропоновано встановити та активно використовувати застосунок протягом певного періоду, а потім заповнити анонімну форму опитування. Форма включала питання, що охоплювали ключові аспекти взаємодії з календарем: загальне враження від інтерфейсу, інтуїтивність, зручність управління завданнями, ефективність візуальної індикації, корисність сповіщень, важливість персоналізації, вплив мотиваційних фраз, стабільність та продуктивність, а також пропозиції щодо покращення.

На рисунку 3.15 наведено приклад скріншоту онлайн-форми опитування, використаної для апробації.

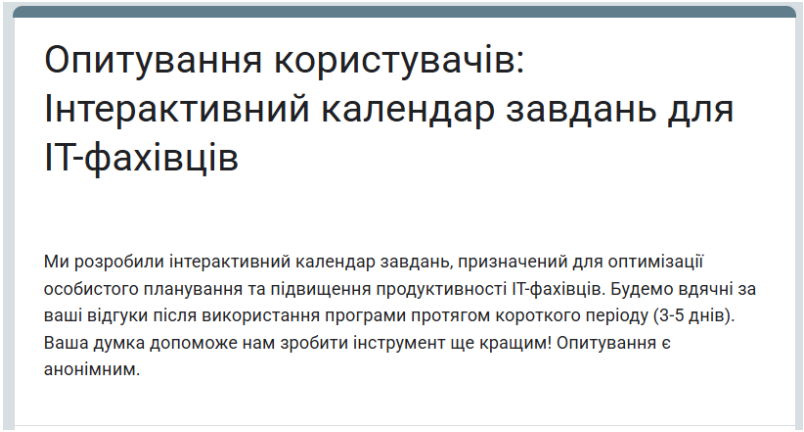

Рисунок 3.15 – Приклад форми опитування для апробації

Результати опитування, отримані від 15 респондентів, показали високий рівень задоволеності користувачів та підтвердили практичну цінність розробленого інструменту. Більшість IT-фахівців відзначили інтуїтивно зрозумілий інтерфейс та легкість освоєння програми.

1. Загальне враження від інтерфейсу: 80% респондентів оцінили інтерфейс на «5» (чудово), 20% – на «4». Це свідчить про високу позитивну оцінку користувацького досвіду.

2. Наскільки інтуїтивним та легким у освоєнні є інтерфейс календаря? 100% учасників оцінили інтерфейс як «дуже легкий» в освоєнні («5»). Це підкреслює простоту та доступність застосунку для нових користувачів.

3. Оцініть зручність додавання, редагування та видалення завдань. 93.3% респондентів визнали зручність додавання, редагування та видалення завдань на «5», а 6.7% – на «4». Це свідчить про високу ефективність функціоналу управління завданнями.

На рисунку 3.16 представлені узагальнені результати опитування щодо загального враження від інтерфейсу, його інтуїтивності та зручності управління завданнями.

C1 | Загальне враження від інтерфейсу програми:

Form_Responses1

	A	B	C	D	E
1	Позначка часу	Ваше ім'я та посада	Загальне враження від інтерфейсу програ	Наскільки інтуїтивним та легким у освоєні	Оцініть зручність додавання, редагування
2	21.05.2025 20:19:08	Максим, QA Engineer	5	5	5
3	21.05.2025 21:21:15	Анастасія, Data Scientist	5	5	5
4	21.05.2025 21:25:32	Станіслав, Backend Develc	5	5	4
5	21.05.2025 21:29:07	Артем, Frontend Developer	4	5	5
6	21.05.2025 21:31:15	Руслан, DevOps Engineer	5	5	5
7	21.05.2025 21:33:46	Вікторія, Mobile Developer	5	5	5
8	21.05.2025 21:35:22	Євген, System Administrat	4	5	5
9	21.05.2025 21:38:01	Олександр, Product Mana	5	5	5
10	21.05.2025 21:40:28	Назар, Junior Developer	5	5	5
11	21.05.2025 21:42:12	Олександр, Senior Develo	5	5	5
12	21.05.2025 21:47:14	Олена, UX/UI Designer	5	5	5
13	21.05.2025 21:48:46	Віталій, Project Manager	5	5	5
14	21.05.2025 21:50:18	Андрій, Software Architect	4	5	5
15	21.05.2025 21:51:38	Андрій, Network Engineer	5	5	5
16	21.05.2025 21:53:04	Роман, Cybersecurity Anal	5	5	5

Рисунок 3.16 – Узагальнені результати опитування користувачів (частина 1)

Особливо позитивно були оцінені наступні аспекти:

1. Кольорова індикація завдань на календарі: 100% респондентів оцінили її ефективність на «5». Учасники підкреслили, що це значно прискорює оцінку завантаженості дня та допомагає швидко виявити прострочені завдання.

2. Нативні системні сповіщення: 100% учасників визнали їх «дуже корисними» та «своєчасними» («5»). Були визнані дуже корисними для своєчасного нагадування про дедлайни, особливо під час роботи в інших програмах.

3. Можливість персоналізації: 100% респондентів визнали цю функцію «дуже важливою» («5»). Функція зміни тем та кольорів отримала високі оцінки важливості, оскільки дозволяє адаптувати вигляд програми під індивідуальні вподобання, що підвищує комфорт роботи.

4. Стабільність та продуктивність: 100% учасників оцінили роботу програми на «Відмінно» («5»).

5. Мотиваційні фрази: 86.7% респондентів вважають, що ця функція сприяє підвищенню продуктивності/настрою.

На рисунку 3.17 представлені узагальнені результати опитування щодо ефективності кольорової індикації, корисності сповіщень, важливості персоналізації, впливу мотиваційних фраз та стабільності роботи програми.

Наскільки ефективною є система кольоро	Наскільки корисними та своєчасними є си	Наскільки важлива для вас можливість пе	Чи вважаєте ви, що функція мотиваційних	Як ви оцінюєте стабільність та продуктивн
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Важко сказати	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Важко сказати	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5
5	5	5	Так	5

Рисунок 3.17 – Узагальнені результати опитування користувачів (частина 2)

Серед найчастіших пропозицій щодо покращення функціоналу, отриманих з відкритих відповідей, були: розширення функціоналу пошуку та фільтрації завдань, а також можливість експорту/імпорту даних (наприклад, у формати CSV/JSON). Деякі респонденти також запропонували розширити візуальні стилі інтерфейсу. Вищенаведені цінні відгуки будуть враховані для подальшого розвитку проєкту.

3.4 Висновок до третього розділу

У цьому розділі було детально висвітлено процес проєктування та безпосередньої реалізації програмної системи "Інтерактивний календар завдань". Розроблена архітектура базується на модульному та багатошаровому підході, що забезпечує чітке розділення відповідальності між рівнями представлення, бізнес-логіки та доступу до даних. Це рішення сприяє легкості супроводу, масштабованості та гнучкості застосунку. Була продемонстрована структура файлів проєкту та наведено ключові фрагменти коду, що ілюструють реалізацію основних функціональних модулів.

Презентація користувацького інтерфейсу візуально підтвердила ефективність реалізованих функцій. Було показано, як інтерактивний календар забезпечує зручне планування, деталізовану візуалізацію завдань за допомогою кольорових маркерів та спливаючих підказок, а також інтуїтивне управління записами. Окремо висвітлено функціонал персоналізації інтерфейсу, системних сповіщень про дедлайни та інтеграцію мотиваційних елементів, що значно покращують користувацький досвід та сприяють підвищенню продуктивності.

Проведена апробація та тестування системи серед цільової аудиторії IT-фахівців підтвердили її високу зручність, інтуїтивність та практичну цінність. Результати опитування свідчать про високий рівень задоволеності користувачів ключовими функціональними можливостями, ефективністю візуальної індикації, корисністю сповіщень та важливістю персоналізації. Виявлені

побажання щодо розширення функціоналу пошуку, фільтрації та експорту даних стануть основою для подальших покращень.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У цьому розділі буде розглянуто ключові аспекти безпеки життєдіяльності та охорони праці, які є актуальними для ІТ-фахівців, чия професійна діяльність пов'язана з інтенсивною роботою за комп'ютером. Особлива увага буде приділена питанням пожежної безпеки в ІТ-середовищі, як фундаментальній складовій безпеки життєдіяльності, а також питанням мінімізації ризиків для здоров'я. Розуміння та дотримання цих норм є невід'ємною складовою відповідальної професійної діяльності та важливим чинником підтримки високої продуктивності та благополуччя ІТ-спеціалістів.

4.1 Першочергові заходи пожежної безпеки для робочого місця ІТ-фахівця, враховуючи специфіку використання комп'ютерної техніки та електроніки

Робоче місце ІТ-фахівця, насичене комп'ютерною технікою, моніторами, серверами та мережевим обладнанням потенційно є джерелом підвищеного ризику виникнення пожежі через велике електричне навантаження, наявність горючих матеріалів (кабелі, пластикові корпуси, папір) та можливість перегріву. Тому, першочергові заходи пожежної безпеки для такого середовища мають бути сфокусовані на ефективній профілактиці та готовності до швидкої ліквідації можливого загоряння [35].

Фундаментальним аспектом є забезпечення справності та правильного використання електроустановок. Це включає регулярний огляд електропроводки, розеток, подовжувачів та іншого електричного обладнання на предмет пошкоджень, перегріву або іскріння [36]. Особливо важливо використовувати електропроводку та мережеві фільтри, які повністю відповідають сумарному електричному навантаженню підключеної техніки, аби уникнути перевантаження електромережі, що є однією з найпоширеніших

причин пожеж. Категорично забороняється підключати занадто багато пристроїв в одну розетку, а також використовувати пошкоджену або несправну електротехніку чи саморобні електронагрівальні прилади. Усе обладнання повинно бути сертифікованим та належним чином заземленим. Регулярний огляд та обслуговування електроустановок кваліфікованим персоналом є обов'язковим. Після закінчення робочого дня або під час тривалої відсутності персонал повинен відключати комп'ютери та іншу електроніку від електромережі, щоб мінімізувати ризики.

Другим важливим напрямком є підтримання чистоти та порядку на робочому місці. Накопичення пилу на корпусах комп'ютерів, у вентиляційних отворах та на клавіатурах є небезпечним, оскільки пил є добрим ізолятором, що може призвести до перегріву компонентів, а також сам по собі є горючим матеріалом [37]. Тому регулярне прибирання є критично важливим. Кабелі мають бути акуратно укладені, щоб уникнути їх пошкодження, заплутування та перегріву; їх розміщення під килимами чи в місцях, де вони можуть бути постійно пошкоджені, суворо заборонено. Важливо також не допускати захаращення робочих місць та проходів горючими матеріалами, такими як надмірна кількість паперу, порожні коробки тощо.

Третій аспект – це забезпечення первинними засобами пожежогасіння. Робоче місце або приміщення, де працює ІТ-фахівець, має бути обов'язково обладнане справними вуглекислотними (ВВК) або порошковими (ВП) вогнегасниками, які підходять для гасіння електрообладнання під напругою (клас пожежі Е) [35]. Ці вогнегасники повинні знаходитись у легкодоступних місцях, а всі працівники мають бути навчені правилам їх використання (див. рисунок 4.1).

Крім того, необхідно суворо дотримуватися режиму пожежної безпеки: це включає повну заборону паління в приміщеннях з комп'ютерною технікою та контроль за відсутністю джерел відкритого вогню.

Рисунок 4.1 – Приклад розміщення засобу пожежогасіння [38]

Кожен ІТ-фахівець повинен пройти відповідний інструктаж з пожежної безпеки, бути ознайомленим з планом евакуації та знати порядок дій у разі виникнення пожежі [39]. Дотримання цих першочергових заходів є запорукою безпеки та значного зниження ризиків виникнення пожежі, забезпечуючи захист персоналу та цінного обладнання в ІТ-середовищі.

4.2 Вплив інтерактивного календаря завдань на дотримання норм охорони праці та психоемоційну стабільність ІТ-фахівців

Застосування інтерактивного календаря завдань, окрім своєї основної функції з оптимізації робочого процесу, відіграє значну роль у сприянні дотриманню норм охорони праці та підвищенню психоемоційної стабільності ІТ-фахівців. Це досягається завдяки систематизації та візуалізації робочого навантаження, що зменшує рівень стресу та ризик професійного вигорання. Ефективне управління часом та завданнями безпосередньо впливає на психічний добробут, дозволяючи фахівцю відчувати більший контроль над своєю роботою [40].

Одним із ключових аспектів є оптимізація робочого навантаження та запобігання перевтомі. Інтерактивний календар дозволяє чітко планувати завдання, встановлювати реалістичні терміни та рівномірно розподіляти

навантаження протягом робочого дня та тижня. Це допомагає уникнути ситуацій авралу та інтенсивних "спринтів", які є значним джерелом фізичної та психологічної втоми. Крім того, інтерактивний календар завдань надає можливість ІТ-фахівцям свідомо інтегрувати у свій робочий графік регулярні короткі перерви – наприклад, за правилом "20-20-20" для відпочинку очей або для виконання легких фізичних вправ. Такі структуровані перерви є критично важливими для підтримки фізичного здоров'я, запобігання зоровому перевантаженню та м'язово-скелетним розладам, тим самим сприяючи дотриманню гігієнічних норм праці. Дослідження показують, що правильно організовані перерви не лише відновлюють ресурси, а й можуть парадоксально підвищувати пікову продуктивність, оскільки мозок отримує необхідний відпочинок.

Важливим внеском інтерактивного календаря є також зниження стресу та запобігання психоемоційному вигоранню. Чітка візуалізація всіх завдань, їх пріоритетів та дедлайнів створює відчуття прозорості та контролю над робочим процесом. Це зменшує тривожність, пов'язану з ризиком забути важливе завдання або пропустити термін, що є значним джерелом стресу для ІТ-фахівців. Можливість візуально відмічати виконані завдання надає відчуття досягнення, що позитивно впливає на мотивацію та психологічний комфорт. Фахівці, які відчувають більший контроль над своїм часом та завданнями, схильні відчувати менший стрес і мають вищий рівень задоволення від роботи [40]. Ефективне планування за допомогою календаря також дозволяє більш чітко розмежовувати робочий та особистий час, що є фундаментальним для профілактики хронічної втоми та емоційного вигорання – станів, які є значним викликом для ІТ-індустрії. Таким чином, інтерактивний календар завдань виступає не лише як інструмент продуктивності, а й як елемент системи підтримки психофізичного благополуччя ІТ-спеціалістів.

4.3 Висновок до четвертого розділу

У цьому розділі було всебічно розглянуто ключові аспекти безпеки життєдіяльності та охорони праці, які є невід'ємною частиною професійної діяльності IT-фахівців, чия робота безпосередньо пов'язана з інтенсивним використанням комп'ютерної техніки. Аналіз показав, що ефективне управління ризиками у цьому середовищі потребує комплексного підходу, що охоплює як безпосередні фізичні загрози, так і психоемоційні аспекти.

У цьому розділі ми розглянули важливі аспекти безпеки праці для IT-фахівців. Зокрема, наголошено на пожежній безпеці, де висвітлено ключові заходи: правильне поводження з електроприладами, підтримка чистоти на робочому місці та забезпечення первинними засобами пожежогасіння. Ці базові кроки є фундаментом для запобігання небезпечним ситуаціям та збереження здоров'я.

Крім того, ми показали, як розроблений інтерактивний календар завдань ефективно сприяє підтримці психоемоційної стабільності та дотриманню норм охорони праці. Календар дозволяє фахівцям раціонально розподіляти робоче навантаження, що допомагає уникати перевтоми та знижує рівень стресу. Можливість візуального контролю над завданнями, пріоритизація та чітке розмежування робочого та особистого часу за допомогою календаря є дієвими інструментами для профілактики професійного вигорання та підвищення загального благополуччя, створюючи більш безпечне та комфортне робоче середовище.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» було розроблено автономний десктопний застосунок "Інтерактивний календар завдань", призначений для оптимізації індивідуального робочого процесу IT-фахівців.

В першому розділі кваліфікаційної роботи:

- обґрунтовано актуальність проблеми управління завданнями в сучасному IT-середовищі та її вплив на продуктивність фахівців, зокрема проблему прокрастинації;
- проведено огляд та порівняльний аналіз існуючих програмних рішень для планування та управління завданнями;
- висвітлено прогалини на ринку, які обґрунтовують необхідність розробки нового інструменту;
- сформульовано проблему, мету та завдання розробки програмної системи;

В другому розділі кваліфікаційної роботи:

- досліджено загальні концепції та принципи розробки програмного забезпечення, включаючи моделі життєвого циклу та принципи проєктування;
- обґрунтовано вибір гнучкої ітеративної методології розробки та дослідницьких методів для проєкту;
- проведено аналіз та формалізацію функціональних та нефункціональних вимог до системи;
- обґрунтовано вибір технологічного стеку для розробки десктопного застосунку.

В третьому розділі кваліфікаційної роботи:

- описано архітектуру та безпосередню реалізацію програмної системи, включаючи її багатошарову структуру та організацію файлів;

- проаналізовано демонстрацію функціональних можливостей та користувацького інтерфейсу, включаючи візуалізацію завдань, систему сповіщень, персоналізацію та мотиваційні елементи;
- проведено апробацію та тестування розробленої системи серед цільової аудиторії ІТ-фахівців, підтвердивши її зручність та ефективність.

У розділі «Безпека життєдіяльності, основи охорони праці» розглянуто ключові аспекти пожежної безпеки на робочому місці ІТ-фахівця та висвітлено важливість підтримки фізичного та психоемоційного стану. Описано запобіжні методи для уникнення та мінімізації негативних наслідків, пов'язаних з тривалою роботою за комп'ютером, підкресливши роль інтерактивного календаря завдань у цьому процесі.

ПЕРЕЛІК ДЖЕРЕЛ

1 Todoist vs. Microsoft To Do: Which Tool Is Best? [2024] | ClickUp [Електронний ресурс]. Режим доступу: https://clickup.com/blog/todoist-vs-microsoft-to-do/ (дата звернення: 09.06.2025).

2 Jira vs Trello vs Asana: Best for PM. Productive [Електронний ресурс]. Режим доступу: https://productive.io/blog/jira-vs-trello-vs-asana/ (дата звернення: 09.06.2025).

3 Institute of Advanced Science Extension (IASE) [Електронний ресурс]. Режим доступу: https://science-gate.com/IJAAS/Articles/2023/2023-10-10/1021833ijaas202310018.pdf (дата звернення: 09.06.2025).

4 Psychometric test for time management in companies - Evalutime [Електронний ресурс]. Режим доступу: https://evalutime.com/blogs/blog-what-role-do-digital-tools-play-in-enhancing-time-management-among-remote-workers-150651 (дата звернення: 09.06.2025).

5 Time Pressure in Software Engineering: A Systematic Review. arXiv.org [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1901.05771 (дата звернення: 09.06.2025).

6 How to Overcome Procrastination. Clockdiary: Your Ultimate Time Management Companion [Електронний ресурс]. Режим доступу: https://clockdiary.com/blog/productivity/how-to-overcome-procrastination (дата звернення: 09.06.2025).

7 Слободян П. С., Небесний Р. М. Актуальність проблем оцінювання якості програмного забезпечення // Матеріали Міжнародної науково-технічної конференції „Фундаментальні та прикладні проблеми сучасних технологій“..., 2018. — Т. : ТНТУ, 2018 (дата звернення: 09.06.2025).

8 Життєві цикли розробки ПЗ [Електронний ресурс]. Режим доступу: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/ (дата звернення: 09.06.2025).

9 Каскадна модель. QALight Waterfall [Електронний ресурс]. Режим доступу: https://qalight.ua/baza-znaniy/kaskadna-model-waterfall-model/ (дата звернення: 09.06.2025).

10 Що таке Agile-методології, принципи і методи в 2025 [Електронний ресурс]. Режим доступу: https://brainrain.com.ua/uk/chto-takoe-agile-ua/ (дата звернення: 09.06.2025).

11 KISS, DRY, S.O.L.I.D, YAGNI – навіщо дотримуватись принципів програмування? [Електронний ресурс]. Режим доступу: https://senior.ua/articles/kiss-dry-solid-yagni--navscho-dotrimuvatis-principv-programuvannya (дата звернення: 09.06.2025).

12 Патерни/шаблони проектування [Електронний ресурс]. Режим доступу: https://refactoring.guru/uk/design-patterns (дата звернення: 09.06.2025).

13 Гайд з використання DRY principle [Електронний ресурс]. Режим доступу: https://dou.ua/forums/topic/35712/ (дата звернення: 09.06.2025).

14 Як створити опитування через Google Форму. eSputnik [Електронний ресурс]. Режим доступу: https://esputnik.com/uk/blog/instrukciya-zi-stvorennya-opituvan-u-google-formah (дата звернення: 09.06.2025).

15 Системний аналіз: необхідна навичка для кожного IT-спеціаліста [Електронний ресурс]. Режим доступу: https://dan-it.com.ua/uk/blog/sistemnij-analiz-neobhidna-navichka-dlja-kozhnogo-it-specialista/ (дата звернення: 09.06.2025).

16 Сутність порівняльно-аналітичного виду дослідження [Електронний ресурс]. Режим доступу: https://naurok.com.ua/sutnist-porivnyalno-analitichnogo-vidu-doslidzhennya-285609.html (дата звернення: 09.06.2025).

17 Дизайн-мислення: як застосовувати метод на практиці [Електронний ресурс]. Режим доступу: https://beetroot.academy/blog/dizayn-mislennya-yak-zastosovuvati-metod-na-praktici (дата звернення: 09.06.2025).

18 Модульність, модуль, категорії модулів. Відношення між модулями [Електронний ресурс]. Режим доступу: https://www.kursak.com/modulnist-modul-katehorii-moduliv/ (дата звернення: 09.06.2025).

19 Петрик М. Моделювання та аналіз програмного забезпечення / Петрик М., Петрик О. -Тернопіль: Видавництво ТНТУ, 2015 (дата звернення: 09.06.2025).

20 [Шевчук, Небесний] - Шевчук О. В., Небесний Р. М. Заходи безпеки інформації у комп'ютерних системах // Матеріали Міжнародної науково-технічної конференції „Фундаментальні та прикладні проблеми сучасних технологій“..., 2018. — Т. : ТНТУ, 2018 (дата звернення: 09.06.2025).

21 Mehta K. Building Desktop Applications with Electron [Електронний ресурс]. Режим доступу: https://dev.to/kartikmehta8/building-desktop-applications-with-electron-1bbg (дата звернення: 10.06.2025).

22 Electron Framework Introduction [Електронний ресурс]. Режим доступу: https://gorillalogic.com/blog-and-resources/electron-framework-introduction (дата звернення: 10.06.2025).

23 Building GUI Applications with Qt and C++. [Електронний ресурс]. Режим доступу: https://geekpedia.com/building-gui-applications-qt-cpp/ (дата звернення: 10.06.2025).

24 PySide2 vs PyQt5 vs PyQt6 vs PySide6. Which to choose? Machine Koder [Електронний ресурс]. Режим доступу: https://machinekoder.com/pyqt-vs-qt-for-python-pyside2-pyside/ (дата звернення: 10.06.2025).

25 Windows Forms documentation. Microsoft Learn [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/ (дата звернення: 10.06.2025).

26 .NET Multi-platform App UI (.NET MAUI) [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-9.0 (дата звернення: 10.06.2025).

27 What is Kivy? EDUCBA [Електронний ресурс]. Режим доступу: https://www.educba.com/what-is-kivy/ (дата звернення: 10.06.2025).

28 Kivy. GeeksforGeeks [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/what-is-kivy/ (дата звернення: 10.06.2025).

29 Kivy Reviews: Pros and Cons. TrustRadius [Електронний ресурс]. Режим доступу: https://www.trustradius.com/products/kivy/reviews?qs=pros-and-cons (дата звернення: 10.06.2025).

30 Python GTK+ 3 Tutorial. GTK Official Documentation [Електронний ресурс]. Режим доступу: https://www.gtk.org/docs/language-bindings/python/ (дата звернення: 10.06.2025).

31 PyGTK, the Python GUI Creation Tool. DataScientest [Електронний ресурс]. Режим доступу: https://datascientest.com/en/pygtk-the-python-gui-creation-tool (дата звернення: 10.06.2025).

32 Which Python GUI Library To Choose? Python GUIs [Електронний ресурс]. Режим доступу: https://www.pythonguis.com/faq/which-python-gui-library/ (дата звернення: 10.06.2025).

33 Мартинюк С. В., Небесний Р. М. Розробка функціонуючої структури програмного консолідованого ресурсу // Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“..., 2017. — Т. : ТНТУ, 2017 (дата звернення: 10.06.2025).

34 Кормило І. В., Небесний Р. М. Побудова інформаційної технології візуалізації інформаційних ресурсів // Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“..., 2017. — Т. : ТНТУ, 2017 (дата звернення: 10.06.2025).

35 Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal. 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77–82 (дата звернення: 10.06.2025).

36 Zaporozhets, Y. Kuts, B. Mlynko, M. Fryz, and L. Scherbak, “EEG Signal Classification Using Linear Process Model-Based Feature Extraction and Supervised Learning,” in Advanced System Development Technologies II. Studies in Systems, Decision and Control, M. Bezuglyi, N. Bouraou, V. Mykytenko, G. Tymchyk, and A.

Zaporozhets, Eds., Cham: Springer Nature Switzerland, 2025, pp. 235–257. doi: 10.1007/978-3-031-82035-9_7 (дата звернення: 10.06.2025).

37 Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques. Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132 (дата звернення: 10.06.2025).

38 M. Fryz, B. Mlynko, Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals, Technology Audit and Production Reserves, 3/2(65), 2022, pp. 29–32 (дата звернення: 10.06.2025).

39 Бабак В. П., Марченко М. Є., Фриз. Б. Г. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с (дата звернення: 10.06.2025).

40 Fire Protection in Server Rooms: Ensuring Safety and Security [Електронний ресурс]. Режим доступу: https://constructionreviewonline.com/knowhow/safety-and-security/fire-protection-in-server-rooms-ensuring-safety-and-security/ (дата звернення: 10.06.2025).

41 The Basics of Office Fire Safety. Safeopedia [Електронний ресурс]. Режим доступу: https://www.safeopedia.com/2024/01/17/the-basics-of-office-fire-safety (дата звернення: 10.06.2025).

42 The Importance of Keeping Cables Dust-Free in Your Business. RCH-COM [Електронний ресурс]. Режим доступу: https://www.safeopedia.com/2024/01/17/the-basics-of-office-fire-safety (дата звернення: 10.06.2025).

43 Fire Extinguishers for Offices - Safelincs. Safelincs - Fire Safety Products [Електронний ресурс]. Режим доступу: https://www.safelincs.co.uk/fire-extinguishers-suitable-for-offices-section/ (дата звернення: 10.06.2025).

44 Office Fire Safety. HSEblog [Електронний ресурс]. Режим доступу: https://www.hseblog.com/office-fire-safety/ (дата звернення: 10.06.2025).

45 Arigoni, C., Arigoni, C., Rizzoli, D., Di Dio, C., Galli, L., Giupponi, B., ... & Zago, S. (2023). Psychological well-being and working from home: A systematic review. International Journal of Environmental Research and Public Health, 20(12), 6140 [Електронний ресурс]. Режим доступу: https://pubmed.ncbi.nlm.nih.gov/37359222/ (дата звернення: 10.06.2025).

46 Working too hard might be hurting you: Scientists reveal the unexpected break time secret to peak performance [Електронний ресурс]. Режим доступу: https://economictimes.indiatimes.com/magazines/panache/working-too-hard-might-be-hurting-you-scientists-reveal-the-unexpected-break-time-secret-to-peak-performance/articleshow/121680118.cms (дата звернення: 10.06.2025).

ДОДАТКИ

Додаток А

Вміст файлу main.py

```
import tkinter as tk
from tkinter import messagebox, simpledialog, Menu, ttk,
colorchooser, Label
from datetime import datetime
from database import get_tasks_for_month, init_db, add_task,
get_tasks_for_date, update_task_status, delete_task
from notifier import check_for_notifications
from functools import partial
import calendar
import json
import os
import random

SETTINGS_FILE = "settings.json"

DEFAULT_SETTINGS = {
    "theme": "Світла",
    "bg_color": "#ffffff",
    "cell_color": "#e0e0e0",
    "text_color": "#000000"
}

def load_motivational_phrases(filename="phrases.txt"):
    if os.path.exists(filename):
        with open(filename, "r", encoding="utf-8") as f:
            return [line.strip() for line in f if line.strip()]
    else:
        return ["Ти можеш усе!"]  # резервна фраза

def load_settings():
    if os.path.exists(SETTINGS_FILE):
        try:
```

```
            with open(SETTINGS_FILE, "r", encoding="utf-8") as f:
                content = f.read().strip()
                if content:
                    loaded = json.loads(content)
                    # доповнюємо відсутні ключі з дефолтних
                    for key, value in DEFAULT_SETTINGS.items():
                        loaded.setdefault(key, value)
                    return loaded
        except json.JSONDecodeError:
            print("⚠ Невірний формат settings.json.")
    return DEFAULT_SETTINGS.copy()

# Після завантаження — оновлюємо DEFAULT_SETTINGS значеннями
користувача
DEFAULT_SETTINGS = load_settings()

def save_settings(settings):
    try:
        with open(SETTINGS_FILE, "w", encoding="utf-8") as f:
            json.dump(settings, f, ensure_ascii=False, indent=4)
    except Exception as e:
        print(f"✖ Не вдалося зберегти налаштування: {e}")

def update_colors_for_theme(theme):
     #Оновлені кольори для тем
    return {
        "Світла": {"bg_color": "#ffffff", "cell_color": "#E0E0E0",
"text_color": "#000000"},
        "Темна": {"bg_color": "#000000", "cell_color": "#323232",
"text_color": "#ffffff"}
    }.get(theme, DEFAULT_SETTINGS)

init_db()

class TooltipCanvas:
```

```
    def __init__(self, canvas, item_id, text):
        self.canvas = canvas
        self.item_id = item_id
        self.text = text
        self.tipwindow = None

        canvas.tag_bind(item_id, "<Enter>", self.show_tip)
        canvas.tag_bind(item_id, "<Leave>", self.hide_tip)

    def show_tip(self, event=None):
        if self.tipwindow or not self.text:
            return
        x = self.canvas.winfo_rootx() + event.x + 10
        y = self.canvas.winfo_rooty() + event.y + 10

        self.tipwindow = tw = tk.Toplevel(self.canvas)
        tw.wm_overrideredirect(True)
        tw.wm_geometry(f"+{x}+{y}")

        label = tk.Label(tw, text=self.text, justify='left',
                         background="#ffffe0", relief='solid',
borderwidth=1,
                         font=("tahoma", "8"))
        label.pack(ipadx=4)

    def hide_tip(self, event=None):
        if self.tipwindow:
            self.tipwindow.destroy()
            self.tipwindow = None


class CustomCalendar(tk.Frame):
    def __init__(self, master=None, on_date_select=None,
on_month_change=None, settings=None, open_settings=None,
                 on_add_task=None, on_delete_task=None):
```

```
        super().__init__(master)
        self.on_date_select = on_date_select
        self.on_month_change = on_month_change
        self.on_add_task = on_add_task
        self.on_delete_task = on_delete_task
        self.settings = settings or DEFAULT_SETTINGS
        self.open_settings = open_settings

        self.theme = self.settings["theme"]
        self.bg_color = self.settings["bg_color"]
        self.cell_color = self.settings["cell_color"]
        self.text_color = self.settings["text_color"]
        self.font = ("Arial", 10)

        self.selected_day = None
        self.year = datetime.now().year
        self.month = datetime.now().month
        self.today = datetime.now().day  # Зберігаємо поточний
день
        self.tasks_by_date = {}
        self.day_item_ids = {}  # Словник для зберігання ID
елементів "день"

        self.canvas = tk.Canvas(self, width=460, height=400)
        self.canvas.pack()

        self.build_widgets()
        self.draw_calendar()

    def build_widgets(self):
        top_frame = tk.Frame(self)
        top_frame.pack(fill="x", pady=5, padx=10)

        # Ліва частина — комбобокси
        left_frame = tk.Frame(top_frame)
```

```
        left_frame.pack(side="left")

        self.month_var =
tk.StringVar(value=calendar.month_name[self.month])
        self.year_var = tk.IntVar(value=self.year)

        months = list(calendar.month_name)[1:]
        years = list(range(self.year - 10, self.year + 11))

        self.month_menu = ttk.Combobox(left_frame, values=months,
textvariable=self.month_var, state="readonly", width=10)
        self.year_menu = ttk.Combobox(left_frame, values=years,
textvariable=self.year_var, state="readonly", width=6)

        self.month_menu.grid(row=0, column=0, padx=2)
        self.year_menu.grid(row=0, column=1, padx=2)

        self.month_menu.bind("<<ComboboxSelected>>", lambda e:
self.update_calendar())
        self.year_menu.bind("<<ComboboxSelected>>", lambda e:
self.update_calendar())

        # Права частина — кнопки
        right_frame = tk.Frame(top_frame)
        right_frame.pack(side="right")

        add_btn = tk.Button(right_frame, text="➕ Додати",
command=self.on_add_task)
        settings_btn = tk.Button(right_frame, text="⚙",
command=self.open_settings)

        add_btn.grid(row=0, column=0, padx=2)
        settings_btn.grid(row=0, column=2, padx=2)

    def update_calendar(self):
```

```
        self.month =
list(calendar.month_name).index(self.month_var.get())
        self.year = self.year_var.get()
        self.draw_calendar()
        if self.on_month_change:
            self.on_month_change(self.year, self.month)

    def draw_calendar(self):
        self.canvas.delete("all")
        self.canvas.config(bg=self.bg_color)

        for i, day in enumerate(["Пн", "Вт", "Ср", "Чт", "Пт",
"Сб", "Нд"]):
            self.canvas.create_text(40 + i*60, 20, text=day,
font=self.font, fill=self.text_color)

        days_in_month = calendar.monthrange(self.year,
self.month)[1]
        start_day = (calendar.monthrange(self.year, self.month)[0]
+ 6) % 7
        row, col = 1, start_day

        self.tasks_by_date = {}
        self.day_item_ids = {}
        self.today_rect_id = None  # Скидаємо ID прямокутника
поточної дати
        for task in get_tasks_for_month(self.year, self.month):
            deadline = task[3][:10]
            is_done = task[4]

            if deadline not in self.tasks_by_date:
                self.tasks_by_date[deadline] = {"total": 0,
"done_count": 0}

            self.tasks_by_date[deadline]["total"] += 1
```

```
            if is_done:
                self.tasks_by_date[deadline]["done_count"] += 1

            # Check for overdue tasks
            if task[3] and datetime.strptime(task[3], '%Y-%m-%d
%H:%M') < datetime.now() and not is_done:
                self.tasks_by_date[deadline]["has_overdue"] = True

        for day in range(1, days_in_month + 1):
            x, y = col * 60 + 10, row * 60 + 20
            rect_id = self.canvas.create_rectangle(x, y, x+50,
y+50, fill=self.cell_color, outline="gray")

            if day == self.selected_day:
                self.canvas.itemconfig(rect_id, fill="#a0a0a0" if
self.theme == "Світла" else "#505050")

            date_str = f"{self.year}-{self.month:02}-{day:02}"
            if date_str in self.tasks_by_date:
                # Ensure 'has_overdue' exists even if no overdue
tasks initially
                if "has_overdue" not in
self.tasks_by_date[date_str]:
                    self.tasks_by_date[date_str]["has_overdue"] =
False

                done = self.tasks_by_date[date_str]["done_count"]
                total = self.tasks_by_date[date_str]["total"]
                color = "green" if done == total else "blue"
                tooltip_text = f"Виконано: {done} / {total}"

                 # Отримуємо всі завдання на цю дату
                tasks_today = get_tasks_for_date(date_str)
                overdue = 0
                upcoming = 0
```

```
                for task in tasks_today:
                    task_deadline = task[3]
                    is_done = task[4]
                    if not is_done and task_deadline:
                        dt = datetime.strptime(task_deadline, '%Y-
%m-%d %H:%M')
                        if dt < datetime.now():
                            overdue += 1
                        else:
                            upcoming += 1

                # Визначаємо колір
                if overdue and upcoming:
                    color = "orange"
                elif overdue:
                    color = "red"
                elif done == total:
                    color = "green"
                else:
                    color = "blue"

                # Текст тултіпу
                tooltip_text = f"Виконано: {done} / {total}"
                if overdue:
                    tooltip_text += f"\nПрострочених: {overdue}"
                if upcoming:
                    tooltip_text += f"\nАктуальних: {upcoming}"

                oval_id = self.canvas.create_oval(x+35, y+35,
x+45, y+45, fill=color, outline="")
                TooltipCanvas(self.canvas, oval_id, tooltip_text)

            text_id=self.canvas.create_text(x+25, y+25,
text=str(day), font=self.font, fill=self.text_color)
```

```
            self.day_item_ids[day] = (rect_id, text_id)  # 
Зберігаємо ID
            self.canvas.tag_bind(rect_id, "<Button-1>", lambda e, 
d=day: self.on_date_click(d))

            # Підсвічування сьогоднішньої дати
            if self.year == datetime.now().year and self.month == 
datetime.now().month and day == self.today:
                self.canvas.itemconfig(rect_id, fill="yellow")  # 
Виділяємо жовтим кольором
                self.today_rect_id = rect_id  # Зберігаємо ID 
прямокутника поточної дати

            self.canvas.tag_bind(rect_id, "<Button-1>", lambda e, 
d=day: self.on_date_click(d))
            self.canvas.tag_bind(text_id, "<Button-1>", lambda e, 
d=day: self.on_date_click(d))

            # Hover ефект
            self.canvas.tag_bind(rect_id, "<Enter>", lambda e, 
d=day: self.on_day_enter(d))
            self.canvas.tag_bind(rect_id, "<Leave>", lambda e, 
d=day: self.on_day_leave(d))
            self.canvas.tag_bind(text_id, "<Enter>", lambda e, 
d=day: self.on_day_enter(d))
            self.canvas.tag_bind(text_id, "<Leave>", lambda e, 
d=day: self.on_day_leave(d))

            col += 1
            if col > 6:
                col = 0
                row += 1

    def on_date_click(self, day):
        self.selected_day = day
```

```
        if self.on_date_select:
            date_str = f"{self.year}-{self.month:02}-{day:02}"
            self.on_date_select(date_str)

        # Оновлюємо стан is_today_selected
        self.is_today_selected = (self.year == datetime.now().year
and
                                  self.month == datetime.now().month
and
                                  day == datetime.now().day)

        self.draw_calendar() # Перемальовуємо календар після
вибору дати

    def on_day_enter(self, day):
        rect_id, text_id = self.day_item_ids.get(day)
        if self.year == datetime.now().year and self.month ==
datetime.now().month and day == datetime.now().day:
            self.canvas.itemconfig(rect_id, fill="#d4af37")
        else:
            self.canvas.itemconfig(rect_id, fill="#b0b0b0" if
self.theme == "Світла" else "#404040")

    def on_day_leave(self, day):
        rect_id, text_id = self.day_item_ids.get(day)
        if self.year == datetime.now().year and self.month ==
datetime.now().month and day == datetime.now().day:
            self.canvas.itemconfig(rect_id, fill="yellow")  #
Залишаємо жовтий колір для поточної дати
        elif day == self.selected_day:
            self.canvas.itemconfig(rect_id, fill="#a0a0a0" if
self.theme == "Світла" else "#505050")
        else:
            self.canvas.itemconfig(rect_id, fill=self.cell_color)
```

```
class TaskCalendarApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Календар завдань")
        self.motivational_phrases = load_motivational_phrases()
        self.motivation_label = tk.Label(self.root, text="",
font=("Arial", 12, "italic"), fg="darkgreen")
        self.motivation_label.pack(pady=5)
        self.show_motivational_phrase()

        self.root.geometry("800x700")

        self.settings = load_settings()
        self.selected_date = datetime.now().strftime('%Y-%m-%d')
        self.selected_task = None

        self.calendar = CustomCalendar(self.root,

on_date_select=self.on_date_select,

on_month_change=self.on_month_change,
                                       settings=self.settings,

open_settings=self.open_settings,

on_add_task=self.add_task_dialog,

on_delete_task=self.delete_task_with_confirm)
        self.calendar.pack(pady=5)

        self.task_frame = tk.Frame(self.root)
        self.task_frame.pack(fill="both", expand=True, pady=10)

        self.tasks_for_date_frame = tk.Frame(self.task_frame)
        self.tasks_for_date_frame.pack(fill="both", expand=True)
```

```
        all_events_label = tk.Label(self.task_frame, text="Усі
події місяця:")
        all_events_label.pack(anchor="w", padx=5)

        # Обгортка для скролюваних подій місяця
        events_wrapper = tk.Frame(self.task_frame)
        events_wrapper.pack(fill="both", expand=True, padx=5,
pady=5)

        # Canvas
        self.events_canvas = tk.Canvas(events_wrapper,
highlightthickness=0)
        self.events_canvas.pack(side="left", fill="both",
expand=True)

        # Scrollbar
        events_scrollbar = ttk.Scrollbar(events_wrapper,
orient="vertical", command=self.events_canvas.yview)
        events_scrollbar.pack(side="right", fill="y")


self.events_canvas.configure(yscrollcommand=events_scrollbar.set)

        # Фрейм усередині Canvas
        self.events_frame = tk.Frame(self.events_canvas)
        self.events_window = self.events_canvas.create_window((0,
0), window=self.events_frame, anchor="nw")

        # Динамічна прокрутка
        def on_configure(event):

self.events_canvas.configure(scrollregion=self.events_canvas.bbox(
"all"))
```

```
        def resize_canvas(event):
            self.events_canvas.itemconfig(self.events_window,
width=event.width)

        self.events_frame.bind("<Configure>", on_configure)
        self.events_canvas.bind("<Configure>", resize_canvas)



        self.load_tasks()
        self.load_all_month_events()
        self.show_today_tasks_popup()
        self.root.after(60000, self.run_notifications)

    def show_motivational_phrase(self):
        if self.motivational_phrases:
            phrase = random.choice(self.motivational_phrases)
            self.motivation_label.config(text=phrase)
        else:
            self.motivation_label.config(text="(Немає фраз)")

        self.root.after(60000, self.show_motivational_phrase)

    def show_today_tasks_popup(self):
        today_str = datetime.now().strftime('%Y-%m-%d')
        today_tasks = get_tasks_for_date(today_str)

        if not today_tasks:
            return  # якщо завдань нема — не показуємо

        done_lines = []
        overdue_lines = []
        upcoming_lines = []

        for task in today_tasks:
            title = task[1]
```

```
            deadline = task[3]
            is_done = task[4]
            time_part = deadline[11:16] if deadline else "без
часу"

            if is_done:
                done_lines.append(f"✅ {time_part} — {title}")
            else:
                dt = datetime.strptime(deadline, '%Y-%m-%d %H:%M')
if deadline else None
                if dt and dt < datetime.now():
                    overdue_lines.append(f"🔴 {time_part} —
{title}")
                else:
                    upcoming_lines.append(f"🔵 {time_part} —
{title}")

        task_text = ""
        if overdue_lines:
            task_text += "🔴 Прострочені:\n" +
"\n".join(overdue_lines) + "\n\n"
        if upcoming_lines:
            task_text += "🔵 Актуальні:\n" +
"\n".join(upcoming_lines) + "\n\n"
        if done_lines:
            task_text += "✅ Виконані:\n" + "\n".join(done_lines)

        messagebox.showinfo("Нагадування на сьогодні", f"Завдання
на сьогодні ({today_str}):\n\n{task_text}")

    def on_date_select(self, date_str):
        self.selected_date = date_str
        self.load_tasks()
```

```
    def on_month_change(self, year, month):
        self.load_all_month_events()

    def load_all_month_events(self):
        for widget in self.events_frame.winfo_children():
            widget.destroy()

        all_tasks = get_tasks_for_month(self.calendar.year,
self.calendar.month)
        if not all_tasks:
            return

        for task in all_tasks:
            task_id, title, desc, deadline, is_done = task
            frame = tk.Frame(self.events_frame)
            frame.pack(fill="x", padx=5, pady=2)

            btn_text = f"{deadline[:10]} — {title}"
            btn = tk.Button(frame, text=btn_text, anchor="w",
justify="left",
                            command=lambda tid=task_id:
self.show_event_details(tid))
            btn.pack(side="left", fill="x", expand=True)

            delete_btn = tk.Button(frame, text="❌",
command=partial(self.delete_task_with_confirm, task_id))
            delete_btn.pack(side="right", padx=2)

            mark_done_btn = tk.Button(frame, text="✅",
command=partial(self.toggle_task_status_by_id, task_id, is_done))
            mark_done_btn.pack(side="right", padx=2)

    def show_event_details(self, task_id):
        tasks = get_tasks_for_month(self.calendar.year,
self.calendar.month)
```

```
        task = next((t for t in tasks if t[0] == task_id), None)
        if task:
            _, title, desc, deadline, is_done = task
            messagebox.showinfo("Подія", f"Назва: {title}\nДата:
{deadline}\nОпис: {desc or '(немає)'}")

    def load_tasks(self):
        # Очистити всі попередні віджети
        for widget in self.tasks_for_date_frame.winfo_children():
            widget.destroy()

        tasks = get_tasks_for_date(self.selected_date)
        for task in tasks:
            task_id, title, desc, deadline, is_done = task

            row = tk.Frame(self.tasks_for_date_frame)
            row.pack(fill="x", padx=10, pady=2)

            status_icon = "✅" if is_done else "⏳"
            status_lbl = tk.Label(row, text=status_icon, width=4)
            status_lbl.pack(side="left")

            label_text = f"{title} — {deadline if deadline else
'без дедлайну'}"
            task_label = tk.Label(row, text=label_text,
anchor="w")
            task_label.pack(side="left", fill="x", expand=True)

            delete_btn = tk.Button(row, text="❌",
command=partial(self.delete_task_with_confirm, task_id))
            delete_btn.pack(side="right", padx=2)

            mark_done_btn = tk.Button(row, text="✅",
command=partial(self.toggle_task_status_by_id, task_id, is_done))
            mark_done_btn.pack(side="right", padx=2)
```

```
            edit_btn = tk.Button(row, text="🖉",
command=partial(self.edit_task, task_id))
            edit_btn.pack(side="right", padx=2)

        self.load_all_month_events()

    def add_task_dialog(self):
        title = simpledialog.askstring("Назва", "Введіть назву
завдання:", parent=self.root)
        if not title:
            return
        desc = simpledialog.askstring("Опис", "Опис
(необов’язково):", parent=self.root)
        deadline = simpledialog.askstring("Дедлайн", "Дата і час
(yyyy-mm-dd HH:MM):", initialvalue=self.selected_date + " 12:00",
parent=self.root)

        try:
            datetime.strptime(deadline, '%Y-%m-%d %H:%M')
            add_task(title, desc, deadline)
            self.load_tasks()
        except ValueError:
            messagebox.showerror("Помилка", "Невірний формат
дати")

    def toggle_task_status_by_id(self, task_id, is_done):
        update_task_status(task_id, 0 if is_done else 1)
        self.load_tasks()

    def show_context_menu_manual(self, task_id, widget):
        self.context_menu = Menu(self.root, tearoff=0)
        self.context_menu.add_command(label="Редагувати",
command=lambda: self.edit_task(task_id))
```

```
        self.context_menu.add_command(label="Видалити",
command=lambda: self.delete_task_with_confirm(task_id))
        self.context_menu.post(widget.winfo_rootx(),
widget.winfo_rooty() + widget.winfo_height())

    def edit_task(self, task_id):
        # Отримати завдання за id
        tasks = get_tasks_for_date(self.selected_date)
        task = next((t for t in tasks if t[0] == task_id), None)
        if not task:
            return

        old_title, old_desc, old_deadline = task[1], task[2],
task[3]

        new_title = simpledialog.askstring("Редагувати назву",
"Нова назва:", initialvalue=old_title, parent=self.root)
        if not new_title:
            return

        new_desc = simpledialog.askstring("Опис", "Новий опис
(необов’язково):", initialvalue=old_desc or "", parent=self.root)
        new_deadline = simpledialog.askstring("Дедлайн", "Нова
дата і час (yyyy-mm-dd HH:MM):", initialvalue=old_deadline,
parent=self.root)

        try:
            datetime.strptime(new_deadline, '%Y-%m-%d %H:%M')
            delete_task(task_id)  # видаляємо старе
            add_task(new_title, new_desc, new_deadline)  # додаємо
нове
            self.load_tasks()
        except ValueError:
            messagebox.showerror("Помилка", "Невірний формат
дати")
```

```
    def delete_task_with_confirm(self, task_id):
        confirm = messagebox.askyesno("Видалити завдання", "Ти
впевнений?")
        if confirm:
            delete_task(task_id)
            self.load_tasks()

    def run_notifications(self):
        check_for_notifications()
        self.root.after(60000, self.run_notifications)

    def open_settings(self):
        settings_win = tk.Toplevel(self.root)
        settings_win.title("Налаштування")
        settings_win.transient(self.root)
        settings_win.grab_set()
        settings_win.focus_set()
        settings_win.lift()
        text_color_label = None
        bg_color_label = None
        cell_color_label = None
        self.text_color_btn = None
        self.bg_color_btn = None
        self.cell_color_btn = None
        self.text_color_btn = None

        tk.Label(settings_win, text="Тема:").pack()
        theme_choice = ttk.Combobox(settings_win,
values=["Світла", "Темна", "Користувацька"])
        theme_choice.set(self.settings["theme"])
        theme_choice.pack()
```

```
        selected_bg_color =
tk.StringVar(value=self.settings["bg_color"])
        selected_cell_color =
tk.StringVar(value=self.settings["cell_color"])
        selected_text_color =
tk.StringVar(value=self.settings["text_color"])

        def update_text_color_visibility():
            nonlocal text_color_label, bg_color_label,
cell_color_label

            if theme_choice.get() == "Користувацька":
                # Текст
                if text_color_label is None:
                    text_color_label = tk.Label(settings_win,
text="Колір тексту:")
                if self.text_color_btn is None:
                    self.text_color_btn = tk.Button(settings_win,
text="Оберіть колір", command=choose_text_color,
bg=selected_text_color.get())
                text_color_label.pack()
                self.text_color_btn.pack(pady=2)

                # Фон
                if bg_color_label is None:
                    bg_color_label = tk.Label(settings_win,
text="Колір фону:")
                if self.bg_color_btn is None:
                    self.bg_color_btn = tk.Button(settings_win,
text="Оберіть колір", command=choose_bg_color,
bg=selected_bg_color.get())
                bg_color_label.pack()
                self.bg_color_btn.pack(pady=2)

                # Комірки
```

```
                if cell_color_label is None:
                    cell_color_label = tk.Label(settings_win,
text="Колір ячейок:")
                if self.cell_color_btn is None:
                    self.cell_color_btn = tk.Button(settings_win,
text="Оберіть колір", command=choose_cell_color,
bg=selected_cell_color.get())
                cell_color_label.pack()
                self.cell_color_btn.pack(pady=2)

            else:
                # Текст
                if text_color_label:
                    text_color_label.pack_forget()
                if self.text_color_btn:
                    self.text_color_btn.pack_forget()

                # Фон
                if bg_color_label:
                    bg_color_label.pack_forget()
                if self.bg_color_btn:
                    self.bg_color_btn.pack_forget()

                # Комірки
                if cell_color_label:
                    cell_color_label.pack_forget()
                if self.cell_color_btn:
                    self.cell_color_btn.pack_forget()

        def choose_bg_color():
            color = colorchooser.askcolor(title="Оберіть колір
фону", initialcolor=selected_bg_color.get())
            if color[1]:
                self.settings["bg_color"] = color[1]
```

```
                selected_bg_color.set(color[1])  # оновлюємо
значення для збереження
                self.bg_color_btn.config(bg=color[1])
                self.calendar.bg_color = color[1]
                self.calendar.draw_calendar()

        def choose_cell_color():
            color = colorchooser.askcolor(title="Оберіть колір
ячейок", initialcolor=selected_cell_color.get())
            if color[1]:
                self.settings["cell_color"] = color[1]
                selected_cell_color.set(color[1])  # оновлюємо
значення для збереження
                self.cell_color_btn.config(bg=color[1])
                self.calendar.cell_color = color[1]
                self.calendar.draw_calendar()


        def choose_text_color():
            color = colorchooser.askcolor(title="Оберіть колір
тексту")
            if color[1]:
                self.settings["text_color"] = color[1]
                selected_text_color.set(color[1])
                self.text_color_btn.config(bg=color[1])
                self.calendar.text_color = color[1]
                self.calendar.draw_calendar()

        update_text_color_visibility()

        theme_choice.bind("<<ComboboxSelected>>", lambda event:
update_text_color_visibility())


        def apply_settings():
```

```
            new_theme = theme_choice.get()

            # Якщо користувач вибрав одну з системних тем —
застосовуємо кольори теми
            if new_theme in ["Світла", "Темна"]:
                colors = update_colors_for_theme(new_theme)
                new_bg = colors["bg_color"]
                new_cell = colors["cell_color"]
                new_text = colors["text_color"]
            else:
                # Користувацька тема
                new_bg = selected_bg_color.get().strip()
                new_cell = selected_cell_color.get().strip()
                new_text = selected_text_color.get().strip()

            self.settings["theme"] = new_theme
            self.settings["bg_color"] = new_bg
            self.settings["cell_color"] = new_cell
            self.settings["text_color"] = new_text

            save_settings(self.settings)
            settings_win.destroy()

            self.calendar.settings = self.settings
            self.calendar.theme = new_theme
            self.calendar.bg_color = new_bg
            self.calendar.cell_color = new_cell
            self.calendar.text_color = new_text
            self.calendar.draw_calendar()


        def reset_to_default():
            theme_choice.set("Світла")  # <- встановлює тему
"Світла"
            selected_bg_color.set(DEFAULT_SETTINGS["bg_color"])
```

```
selected_cell_color.set(DEFAULT_SETTINGS["cell_color"])

selected_text_color.set(DEFAULT_SETTINGS["text_color"])

self.bg_color_btn.config(bg=DEFAULT_SETTINGS["bg_color"])

self.cell_color_btn.config(bg=DEFAULT_SETTINGS["cell_color"])
            if hasattr(self, "text_color_btn") and
self.text_color_btn:
                if self.text_color_btn:

self.text_color_btn.config(bg=DEFAULT_SETTINGS["text_color"])


        tk.Button(settings_win, text="Застосувати",
command=apply_settings).pack(pady=10)
        tk.Button(settings_win, text="Скинути до стандартної
теми", command=reset_to_default).pack(pady=5)

if __name__ == "__main__":
    root = tk.Tk()
    app = TaskCalendarApp(root)
    root.mainloop()
```

Додаток Б

Вміст файлу database.py

```
import sqlite3

def init_db():
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS tasks (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT NOT NULL,
            description TEXT,
            deadline TEXT NOT NULL,
            status INTEGER DEFAULT 0
        )
    ''')
    conn.commit()
    conn.close()

def add_task(title, description, deadline):
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO tasks (title, description,
deadline) VALUES (?, ?, ?)',
                   (title, description, deadline))
    conn.commit()
    conn.close()

def get_tasks_for_date(date_str):
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM tasks WHERE deadline LIKE ?',
(f'{date_str}%',))
    tasks = cursor.fetchall()
```

```
    conn.close()
    return tasks

def get_tasks_for_month(year, month):
    from calendar import monthrange
    start = f"{year}-{month:02d}-01"
    end_day = monthrange(year, month)[1]
    end = f"{year}-{month:02d}-{end_day:02d}"
    return get_tasks_for_range(start, end)


def get_tasks_for_range(start_date, end_date):
    conn = sqlite3.connect("tasks.db")
    cursor = conn.cursor()
    cursor.execute(
        "SELECT * FROM tasks WHERE deadline BETWEEN ? AND ?",
        (start_date + " 00:00", end_date + " 23:59")
    )
    tasks = cursor.fetchall()
    conn.close()
    return tasks


def update_task_status(task_id, status):
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('UPDATE tasks SET status = ? WHERE id = ?',
(status, task_id))
    conn.commit()
    conn.close()

def delete_task(task_id):
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute('DELETE FROM tasks WHERE id = ?', (task_id,))
```

```
conn.commit()
conn.close()
```

Додаток В

Вміст файлу notifier.py

```
from plyer import notification
from datetime import datetime, timedelta
import sqlite3

def check_for_notifications():
    now = datetime.now()
    next_hour = now + timedelta(hours=1)
    conn = sqlite3.connect('tasks.db')
    cursor = conn.cursor()
    cursor.execute("SELECT id, title, deadline FROM tasks WHERE
status = 0")
    tasks = cursor.fetchall()
    conn.close()

    for task in tasks:
        task_id, title, deadline = task
        try:
            task_time = datetime.strptime(deadline, '%Y-%m-%d
%H:%M')
            if now <= task_time <= next_hour:
                notify(title)
        except:
            continue

def notify(message):
    notification.notify(
        title='Нагадування про завдання',
        message=message,
        timeout=10
    )
```