Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ДОДАТКУ ДЛЯ ОСОБИСТОГО ПЛАНУВАННЯ ЧАСУ

DEVELOPMENT OF AM APPLICATION FOR PERSONAL TIME PLANNING

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Шайнюк Владислав Олександрович
_____
(підпис)

Керівник:
Гордєєв Олександр Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«__8__» червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення*»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Шайнюку Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: розробка додатку для особистого планування часу
Керівник роботи: д.т.н., професор Гордєєв О. О.
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № _____
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*

3. Вихідні дані до роботи: апаратне та програмне забезпечення комп'ютера, бібліотеки для роботи з даними, засоби тестування та налагодження програмного продукту, фреймоворки для розробки інтерфейсу користувача, технології Python ,

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу:
1.Приклад додатку
2. Діаграма класів
3.Архітектура додатку

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гордєєв О. О.*		
Специфікація вимог до розробленої системи	*Гордєєв О. О.*		
Розробка об'єкта проєктування	*Гордєєв О. О.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту		92 %	
Академічна доброчесність	*Яшук А. А.*		

7. Дата видачі завдання «2» *лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик.
2.	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*26.02.2024 р.*	вик.
3.	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*13.03.2024 р.*	вик.
4.	*Розробка функціонально-структурної схеми роботи об'єкта проектування та розробки бази даних*	*12.04.2024 р.*	вик.
5.	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик.
6.	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик.
7.	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	вик.

Здобувач вищої освіти _____ (підпис) Шайнюк В. О. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) О. Гордєєв (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет *комп'ютерних та інформаційних технологій*
(назва)

Кафедра *інженерії програмного забезпечення*
(назва)

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: Шайнюк Владислав Олександрович
(ПІБ)

група *ІПЗ-42*
(шифр)

Тема кваліфікаційної роботи:

Розробка додатку для особистого планування часу

(повна назва згідно наказу)

Керівник: *д.т.н професор Гордєєв О.О.*
(науковий ступінь, вчене звання, посада, ПІБ)

Актуальність теми:

Необхідність створення нових, більш зручних та функціональних додатків для планування часу обумовлено тим, що існуючі додатки часто не задовольняють усіх потреб користувачів в частині їх зручності та адаптованості.

Об'єкт дослідження:

Процеси особистого планування часу та управління завданнями за допомогою програмного забезпечення.

Характеристика теоретичного рівня роботи, наявності самостійних розробок і практичної значущості роботи:

Кваліфікаційна робота містить усі необхідна розділи, обсяг та на змістовне наповнення. Дипломна робота відповідає усім вимогам, які висуваються для таких робот бакалаврського рівня.

Зауваження та недоліки:

Аналіз недоліків в існуючих рішеннях був проведений не в повному обсязі.

Загальний висновок:

Кваліфікаційна робота відповідає вимогам, студент був допущений до захисту. Рекомендована оцінка «добре».

Керівник _____ (*Гордєєв Олександр Олександрович*)
(підпис) (ПІБ)

«7» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу бакалавра

Здобувач вищої освіти: Шайнюк Владислав Олександрович

Тема: Розробка додатку для особистого планування часу

Коротка характеристика кваліфікаційної роботи:
Розробка додатку для особистого планування часу є актуальною темою в сучасному світі, де ефективне управління часом стає все більш важливим. Даний додаток дозволяє користувачам створювати та редагувати завдання, вибирати дату та час нагадувань, а також отримувати звукові сповіщення в потрібний момент. Це сприяє підвищенню продуктивності та організованості користувачів.

Самостійні розробки і пропозиції автора:
Проект використовує сучасні технології Python та бібліотеки для роботи з графічним інтерфейсом (Tkinter), звуком (pygame), зображеннями (Pillow) та календарем (tkcalendar). Основні функції додатку включають додавання та редагування завдань з можливістю вибору дати та часу, відтворення звукових сповіщень у зазначений час, встановлення фону зображення для покращення візуального сприйняття, збереження та завантаження завдань з файлу для зручності користувача.

Практичне значення роботи:
Дана робота має широке практичне значення і може бути використана будь-якою людиною, що бажає ефективно керувати своїм часом та завданнями. Такий додаток може стати незамінним помічником для студентів, працівників офісів, фрілансерів та будь-кого, хто прагне організувати свій робочий день. Автоматизовані нагадування допоможуть знизити ризик забуття важливих справ та підвищити особисту продуктивність.

Загальний висновок:
Роботу виконано у повному обсязі, у відповідності із завданням. Здобувач освіти Шайнюк Владислав Олександрович заслуговує на оцінку «добре».

Рецензент: Федорук Анатолій Васильович, к.т.н., доцент кафедри ЕІТК
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«07» 06 2024 р.

АНОТАЦІЯ

Шайнюк В. О. Розробка додатку для особистого планування часу. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Дана робота є персональним планувальником для управління часом, призначеним для використання людьми, які бажають ефективно організувати свій час. Система складається з двох частин: користувацького інтерфейсу і логіки, які взаємодіють між собою через внутрішні процеси. Ця програма дозволяє автоматизувати багато аспектів планування часу, наприклад, створення, редагування та видалення завдань, встановлення нагадувань, збереження та завантаження даних, а також створення звітів про виконані завдання. Користувачі можуть легко додавати нові завдання, редагувати їх, переглядати історію виконаних завдань та отримувати нагадування про важливі події. Програма забезпечує зручне збереження всіх даних в одному місці, що значно полегшує управління часом.

Ключові слова: персональний планувальник, управління часом, мобільний додаток, автоматизація, ефективність.

ABSTRACT

Shainyuk V. O. Development of a Personal Time Management Planner Using Python. Manuscript. Bachelor's qualification work of the OP "Software Engineering" specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

This work is a personal time management planner designed for people who wish to effectively organize their time. The system consists of two parts: the user interface and the logic, which interact with each other through internal processes. This program allows automating many aspects of time management, such as creating, editing, and deleting tasks, setting reminders, saving and loading data, and generating reports on completed tasks. Users can easily add new tasks, edit them, view the history of completed tasks, and receive reminders about important events. The program provides convenient storage of all data in one place, significantly facilitating time management.

Keywords: personal planner, time management, mobile application, automation, efficiency.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі управління особистим часом стає все більш складним завданням через постійне збільшення кількості завдань та обов'язків. Ефективне планування є ключовим для успішної професійної та особистої діяльності. У зв’язку з цим, розробка зручних та ефективних інструментів для організації часу стає надзвичайно важливою.

Розробка персонального планувальника є актуальною з кількох причин:

– збільшення навантаження. Сучасні люди стикаються з дедалі більшими обсягами інформації та завдань, які необхідно виконати у визначені строки. Це вимагає наявності ефективних інструментів для планування та управління часом;

– потреба в автоматизації. Багато людей досі використовують ручні методи або застарілі системи для організації свого часу, що призводить до неефективності, помилок та втрати часу. Автоматизація за допомогою персонального планувальника може значно покращити ці процеси;

– популярність мобільних додатків. Сучасні смартфони стали невід’ємною частиною життя більшості людей. Мобільні додатки є зручним інструментом для управління різноманітними аспектами життя, включаючи планування часу;

– конкурентні переваги. Люди, які використовують сучасні технологічні рішення для управління своїм часом, мають значні конкурентні переваги. Вони можуть більш ефективно організовувати свої завдання, краще контролювати виконання планів та досягати поставлених цілей;

– покращення взаємодії із завданнями. Персональний планувальник дозволяє користувачам легко додавати, редагувати та видаляти завдання, отримувати нагадування про важливі події та зберігати історію виконаних завдань. Це підвищує продуктивність та задоволеність користувачів.

Розробка персонального планувальника стає необхідністю через поширеність мобільних пристроїв та зростаючий попит на ефективні інструменти управління часом. Таким чином, метою даної кваліфікаційної роботи є створення програми для організації особистого часу.

Об’єктом дослідження є процеси особистого планування часу та управління завданнями за допомогою програмного забезпечення.

Предметом дослідження є розробка персонального планувальника для організації особистого часу. Цей додаток має автоматизувати процеси планування, покращити взаємодію з завданнями, оптимізувати розклад та забезпечити підвищення ефективності та зручності у діяльності користувачів.

Для досягнення поставленої мети були визначені наступні завдання:

– аналіз існуючих рішень у сфері планування особистого часу;
– визначення вимог до функціональності та дизайну додатку;
– вибір технологій та інструментів для розробки додатку;
– розробка прототипу додатку та його тестування;
– реалізація функціональних можливостей додатку;
– тестування та налагодження програмного забезпечення;
– написання технічної та користувацької документації.

Таким чином, розробка персонального планувальника для організації особистого часу є актуальною та своєчасною. Вона сприятиме покращенню якості управління часом, підвищенню ефективності та продуктивності користувачів у сучасних умовах.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Аналіз сучасного стану проблеми

Предметною областю даного дослідження є інструменти планування особистого часу. Сьогодні ефективне управління часом є однією з ключових складових успішної діяльності як у професійному, так і в особистому житті. Програми для планування часу або так звані тайм-менеджмент додатки допомагають користувачам організовувати свій день, планувати завдання, встановлювати нагадування та відстежувати свій прогрес.

Актуальність розробки додатка для персонального планування визначається кількома ключовими факторами:

1) зростаюча складність повсякденних завдань: сучасне життя стає все більш насиченим і складним, що вимагає від людей вміння ефективно управляти своїм часом. Багатозадачність, необхідність швидко реагувати на зміни і постійне оновлення інформації створюють значне навантаження на когнітивні ресурси людини;

2) необхідність автоматизації планування: багато людей досі використовують ручні методи для планування своїх справ, що може призводити до неефективності, помилок та зайвих витрат часу. Ручне ведення списків завдань або використання паперових планувальників може бути незручним і неефективним у сучасному цифровому світі. Автоматизація цих процесів за допомогою спеціалізованих додатків дозволяє значно підвищити ефективність управління часом[1];

3) зручність мобільних додатків: сучасні технології і поширеність смартфонів роблять мобільні додатки ідеальним інструментом для управління своїм часом. Вони дозволяють користувачам мати постійний доступ до своїх планів незалежно від місця знаходження. Мобільні додатки також дозволяють синхронізувати дані між різними пристроями, що забезпечує безперервний доступ до інформації;

4) конкурентні переваги: люди, які використовують сучасні технологічні рішення для управління своїм часом, мають значні конкурентні переваги. Вони можуть більш ефективно організовувати свої завдання, краще контролювати виконання планів та досягати поставлених цілей;

5) покращення взаємодії із завданнями: персональний планувальник дозволяє користувачам легко додавати, редагувати та видаляти завдання, отримувати нагадування про важливі події та зберігати історію виконаних завдань. Це підвищує продуктивність та задоволеність користувачів.

Ринок додатків для планування часу є досить насиченим. Існує безліч додатків, які пропонують різноманітні функції для управління завданнями і часом. Розглянемо деякі з них:

1) Todoist: додаток, який дозволяє користувачам створювати списки завдань,встановлювати дедлайни та отримувати нагадування. Перевагами є простота використання, багатофункціональність та інтеграція з іншими сервісами, такими як Google Calendar, Dropbox, Zapier та інші. Недоліком є обмежений функціонал у безкоштовній версії рисунок 1.1, що може стримувати деяких користувачів від повного використання додатку. Todoist також підтримує функції пріоритизації завдань та можливість створення підзадач, що допомагає краще структурувати роботу. Крім того, інтерфейс додатку може бути занадто простим для просунутих користувачів, які потребують більш розширених функцій;

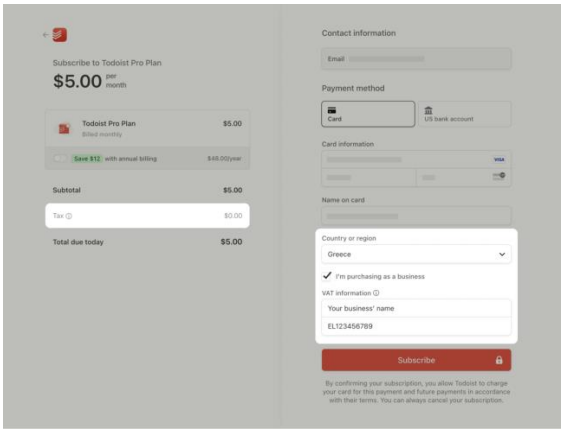

Рисунок 1.1 – Оформлення платної підписки Todoist [2]

2) Trello: дозволяє користувачам створювати дошки, списки і картки для організації своїх завдань і проектів. Підтримує інтеграцію з іншими сервісами і має зручний інтерфейс. Основні переваги Trello включають його візуальну привабливість, гнучкість у налаштуванні та можливість спільної роботи. Недоліками є можливість заплутування в великій кількості карток та обмежений безкоштовний функціонал рисунок 1.2. Trello є популярним серед команд, що працюють над спільними проектами, завдяки своїй здатності візуалізувати завдання та їх статус. Інші функції включають можливість додавання міток, дедлайнів, коментарів та вкладень, що допомагає краще організувати робочий процес. Однак, для індивідуальних користувачів, які не працюють в командах, Trello може бути занадто складним і перевантаженим;

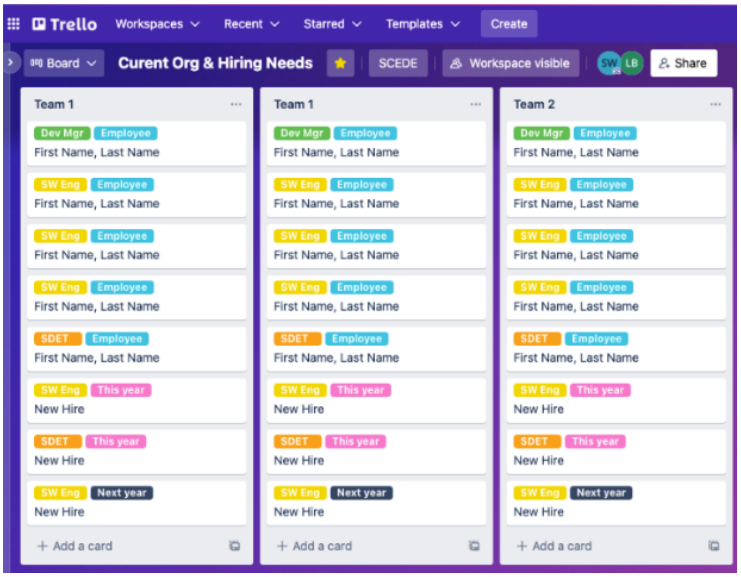

Рисунок 1.2 – Головна сторінка Trello [3]

3) Microsoft To Do рисунок 1.3: інструмент для управління завданнями, який пропонує простий інтерфейс для створення завдань, списків та нагадувань. Переваги включають інтеграцію з іншими продуктами Microsoft, такими як Outlook та Microsoft 365, а також хмарне зберігання даних, що забезпечує доступ з будь-якого пристрою. Недоліком є обмежена функціональність у порівнянні з конкурентами, такими як Todoist та Trello. Однак, для користувачів, які вже

використовують продукти Microsoft, цей додаток може бути дуже зручним і інтегрованим рішенням. Для інших користувачів, які не використовують екосистему Microsoft, функціональність цього додатку може бути недостатньою;

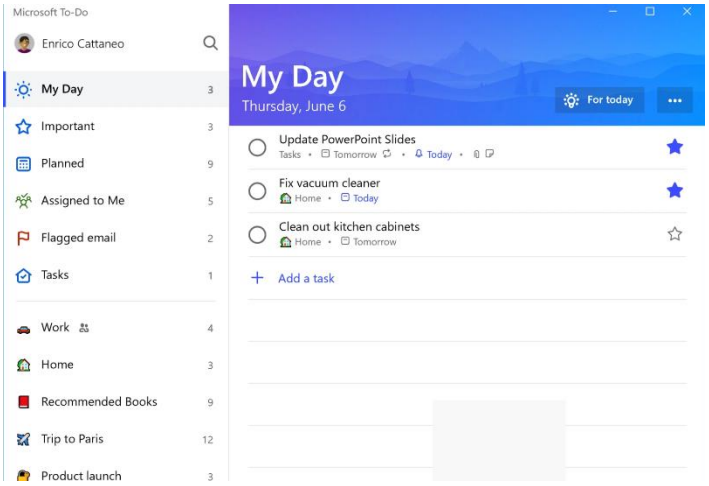

Рисунок 1.3 – Головна сторінка Microsoft To Do [4]

4) Google Keep рисунок 1.4: додаток для створення нотаток і списків завдань. GoogleKeep дозволяє користувачам швидко записувати ідеї, створювати списки справ, додавати зображення та голосові записи. Переваги включають інтеграцію з Google Drive і можливість спільного доступу до нотаток. Недоліком є відносно простий інтерфейс і обмежена функціональність у порівнянні з іншими додатками для управління завданнями. Google Keep також підтримує функцію кольорового кодування нотаток, що дозволяє легко організувати інформацію за категоріями. Проте, відсутність розширених функцій для управління проектами і завданнями може бути недоліком для користувачів, які потребують більш комплексних інструментів.

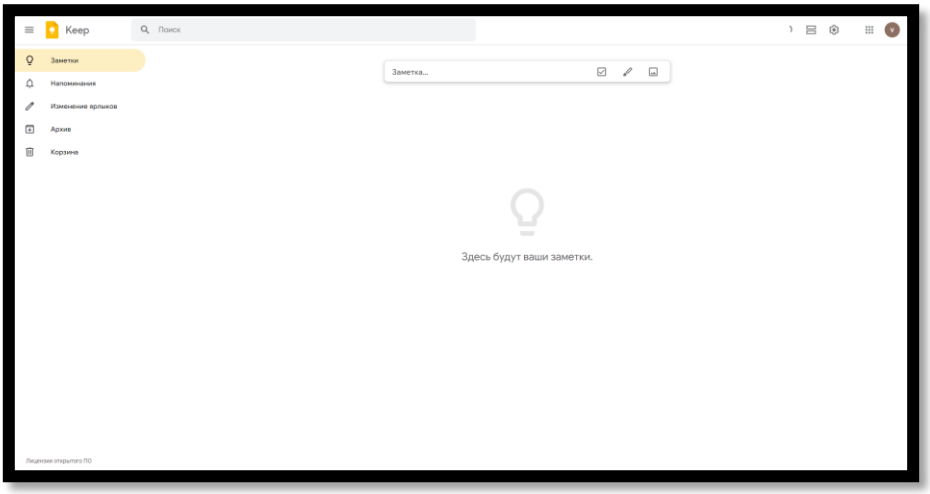

Рисунок 1.4 – Головна сторінка Google Keep [5]

Аналізуючи наведені приклади, можна зробити висновок, що ринок додатків для планування часу є досить насиченим. Однак більшість існуючих додатків є вузькоспеціалізованими і виконують лише одну-дві необхідні функції. Це створює потребу у створенні додатку, який задовольнить більшість запитів користувачів.

Персональний планувальник має на меті об'єднати в одному додатку можливості для створення завдань, встановлення нагадувань, відстеження прогресу та інших функцій, необхідних для ефективного управління часом. Це дозволить користувачам легко і зручно планувати свій день, підвищуючи свою продуктивність і ефективність.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Для досягнення мети розробки програми для організації особистого часу необхідно виконати наступні завдання:

1) аналіз існуючих рішень у сфері планування особистого часу. Провести детальний аналіз наявних на ринку рішень, включаючи конкурентні продукти [6]. Вивчити функціональність, переваги та недоліки аналогічних додатків для планування часу. Це дозволить визначити найкращі практики та уникнути

недоліків, притаманних існуючим рішенням. Визначити основні вимоги, які має задовольняти новий додаток, щоб бути конкурентоспроможним і ефективним;

2) визначення вимог до функціональності та дизайну додатку. На основі проведеного аналізу сформулювати вимоги до персонального планувальника. Вимоги включають функціональні аспекти (управління завданнями, нагадування, створення звітів), а також нефункціональні аспекти (продуктивність, зручність використання, безпека). Розробити технічні специфікації, що описують архітектуру системи та необхідні інтерфейси для взаємодії з користувачами;

3) вибір технологій та інструментів для розробки додатку. Здійснити вибір технологій та інструментів для розробки додатку, враховуючи вимоги до функціональності та продуктивності. Обрати відповідні мови програмування, фреймворки та бібліотеки для фронтенд і бекенд частин додатку. Вибрати інструменти, що забезпечують легкість розробки, підтримку та масштабованість додатку;

4) розробка прототипу додатку та його тестування. Створити прототип користувальницького інтерфейсу додатку. Прототипування включає розробку макетів та інтерактивних моделей, які дозволяють візуалізувати основні елементи та функціонал додатку. Це допоможе отримати зворотний зв’язок від потенційних користувачів та внести необхідні корективи ще на ранніх етапах розробки, що сприятиме покращенню кінцевого продукту;

5) реалізація функціональних можливостей додатку. Реалізувати основні етапи розробки додатку, включаючи написання коду, інтеграцію з бекендом, та проведення тестування. У процесі розробки забезпечити реалізацію всіх визначених функцій, інтеграцію з необхідними сервісами та системами, а також оптимізацію продуктивності додатку. Виконати тестування додатку для виявлення та виправлення помилок, що забезпечить стабільність та надійність кінцевого продукту;

6) тестування та налагодження програмного забезпечення. Провести всебічне тестування програмного забезпечення, включаючи функціональне, нефункціональне, інтеграційне та користувацьке тестування. Виявити та виправити помилки, оптимізувати продуктивність та забезпечити відповідність програмного забезпечення визначеним вимогам і стандартам. Провести завершальне тестування перед релізом для забезпечення надійності та стабільності додатку;

7) підготовка технічної та користувацької документації. Підготувати детальну технічну документацію, що описує архітектуру додатку, його функціональні можливості, інструкції з установки та використання, а також рекомендації для подальшого розвитку та підтримки системи. Документація включає керівництво для користувачів, технічні специфікації для розробників та адміністративні інструкції для адміністраторів системи.

Таким чином, виконання цих завдань дозволить досягти мети роботи та розробити ефективний персональний планувальник для організації особистого часу, що сприятиме покращенню якості управління часом, підвищенню ефективності та продуктивності користувачів у сучасних умовах.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЕНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Розробка персонального планувальника передбачає врахування ряду вимог, які забезпечать його ефективне функціонування та зручність використання. Аналіз цих вимог дозволяє визначити основні функціональні та нефункціональні характеристики додатка.

1) функціональні вимоги:

a) створення та редагування завдань. Користувачі повинні мати можливість створювати нові завдання, вказувати їх назву, дату та час виконання, а також додавати примітки. Завдання повинні бути відредаговані або видалені у будь-який час;

b) встановлення нагадувань. Додаток повинен дозволяти встановлювати нагадування для кожного завдання. Нагадування повинні автоматично сповіщати користувача про наближення часу виконання завдання за допомогою спливаючого вікна та звукового сигналу;

c) відстеження прогресу виконання завдань. Користувачі повинні мати можливість відмічати завдання як виконані та переглядати історію виконаних завдань. Це допоможе бачити свій прогрес та аналізувати ефективність управління часом;

2) нефункціональні вимоги:

a) продуктивність. Додаток повинен працювати швидко і без затримок, забезпечуючи оперативне виконання користувацьких дій;

b) безпека та конфіденційність даних. Всі дані користувачів повинні бути надійно захищені. Використання методів шифрування для захисту даних під час зберігання та передачі є обов’язковим [7];

c) надійність. Додаток повинен бути стабільним і надійним у роботі, мінімізуючи можливість виникнення помилок і збоїв;

3) проектування програмного забезпечення:

a) архітектура додатка. Архітектура додатка повинна бути модульною, що дозволяє легко розширювати та модифікувати його функціональність. Основні модулі включають:

– клієнтська частина: графічний інтерфейс користувача для взаємодії з додатком;

– серверна частина: логіка обробки даних та управління базою даних;

b) вибір технологій та інструментів. Для розробки додатка обрано такі технології:

– мова програмування: Python з використанням бібліотек tkinter для графічного інтерфейсу та pygame для звукових сповіщень;

– база даних[8]: SQLite для локального зберігання даних;

– дизайн бази даних. Структура бази даних повинна включати таблиці для зберігання інформації про завдання, нагадування та виконані завдання. Забезпечення цілісності даних та їх швидкого доступу є пріоритетом;

4) реалізація основних функцій додатка:

a) створення та редагування завдань лістинг 2.1. Реалізація інтерфейсу для створення нових завдань, редагування існуючих та додавання приміток;

Лістинг 2.1 – Основний код для створення та редагування завдань

```
import tkinter as tk
class PersonalPlanner:
    def __init__(self, root):
        self.root = root
        self.tasks = []
        self.task_entry = tk.Entry(root)
        self.task_entry.pack()
        self.add_button=tk.Button(root,text="Додати
завдання",)command=self.add_task)
```

```
        self.add_button.pack()
        self.task_listbox = tk.Listbox(root)
        self.task_listbox.pack()
    def add_task(self):
        task = self.task_entry.get()
        if task:
            self.tasks.append(task)
            self.task_listbox.insert(tk.END, task)
            self.task_entry.delete(0, tk.END)
root = tk.Tk()
app = PersonalPlanner(root)
root.mainloop()
```

Кінець лістингу 2.1

b) встановлення нагадувань лістинг 2.2. Реалізація системи нагадувань, яка буде автоматично сповіщати користувача про наближення часу виконання завдання;

Лістинг 2.2 – Основний код для встановлення нагадувань

```
import tkinter as tk
from datetime import datetime, timedelta
class ReminderApp:
    def __init__(self, root):
        self.root = root
        self.reminders = []
        self.time_entry = tk.Entry(root)
        self.time_entry.pack()
        self.message_entry = tk.Entry(root)
        self.message_entry.pack()
```

```
        self.add_button = tk.Button(root, text="Додати нагадування",
command=self.add_reminder)
        self.add_button.pack()
    def add_reminder(self):
        reminder_time = self.time_entry.get()
        reminder_message = self.message_entry.get()
        if reminder_time and reminder_message:
            reminder_datetime = datetime.strptime(reminder_time, "%H:%M")
            self.reminders.append((reminder_datetime, reminder_message))
            self.time_entry.delete(0, tk.END)
            self.message_entry.delete(0, tk.END)
root = tk.Tk()
app = ReminderApp(root)
root.mainloop()
```

Кінець лістингу 2.1

c) відстеження прогресу виконання завдань. Впровадження можливості відмічати завдання як виконані та переглядати історію виконаних завдань;

d) зручність інтерфейсу користувача. Створення інтуїтивно зрозумілого та зручного інтерфейсу для взаємодії користувачів з додатком;

e) можливість налаштування вигляду додатка. Надання користувачам можливості налаштовувати вигляд додатка відповідно до своїх уподобань.

Таким чином, виконання цих завдань дозволить створити багатофункціональний персональний планувальник, що забезпечить ефективне управління часом, організацію завдань та підвищення продуктивності користувачів.

Для кращого розуміння архітектури програмного забезпечення доцільно використовувати UML діаграми. Вони допоможуть візуалізувати структуру додатка, взаємодію між компонентами та потік даних.

Діаграма класів. Відображає класи додатка, їх властивості та методи, а також взаємозв’язки між ними (рис 2.1).

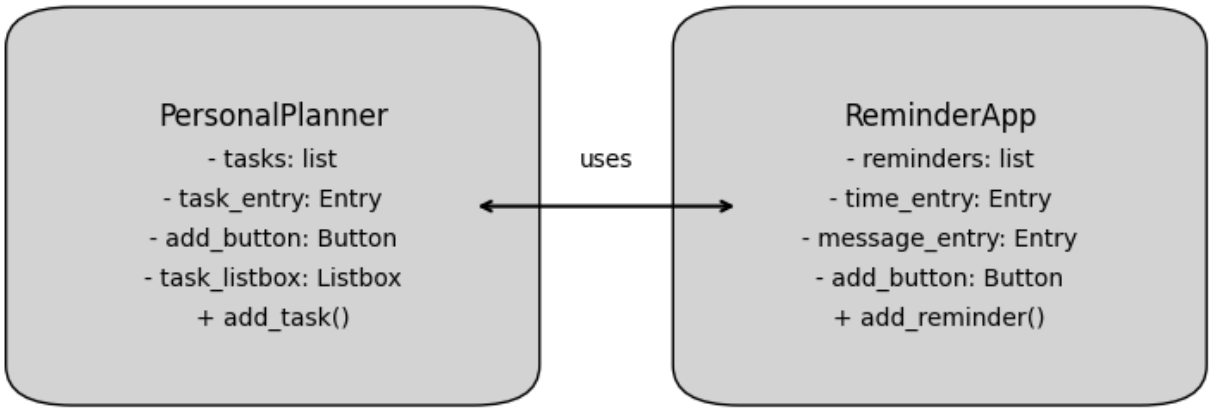

Рисунок 2.1 – Діаграма класів

Діаграма послідовності. Демонструє послідовність викликів методів та взаємодію між об'єктами в процесі виконання певної функції (рис. 2.2).

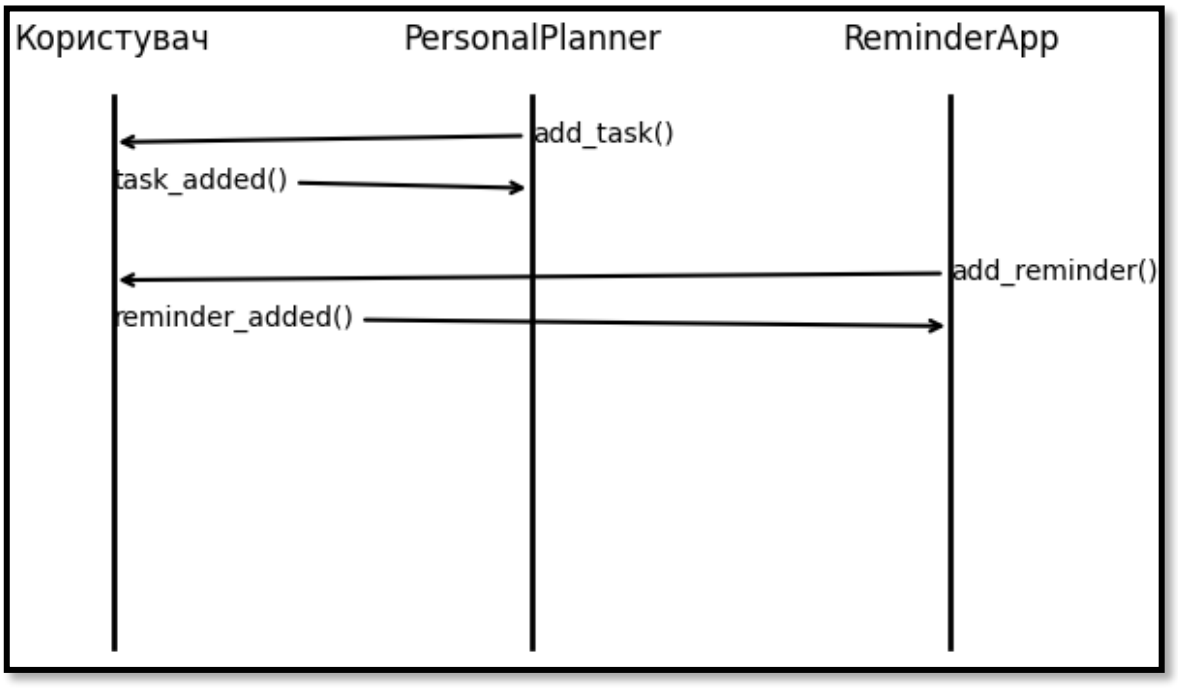

Рисунок 2.2 – Діаграма послідовності

Діаграма діяльності. Описує потік виконання дій та процесів в додатку (рис. 2.3).

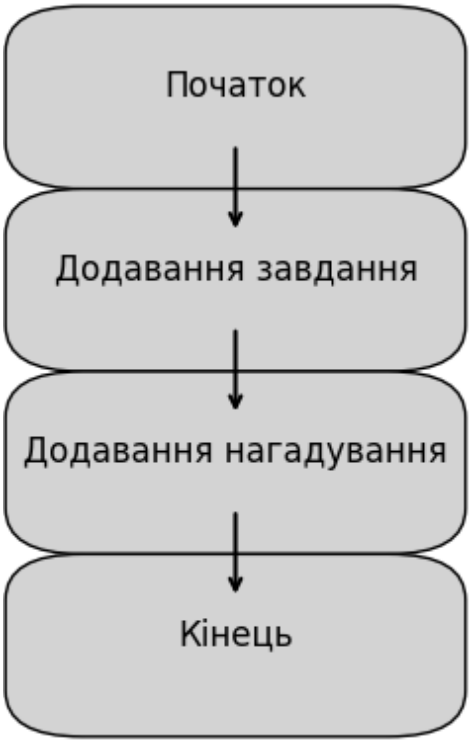

Рисунок 2.3 – Діаграма діяльності

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для реалізації додатку було обрано мову Python через його простоту, зручність та широкий спектр бібліотек, що дозволяють швидко та ефективно розробляти програмне забезпечення. Python має велику спільноту розробників та велику кількість готових рішень, що дозволяє скоротити час на розробку та зосередитись на основних функціональних вимогах. Мова підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване та функціональне програмування, що робить її гнучким інструментом для вирішення різноманітних завдань. Крім того, Python має велику кількість бібліотек та фреймворків, які дозволяють легко інтегрувати додаткову функціональність в додаток, що робить його ідеальним вибором для розробки персонального планувальника.

Бібліотека tkinter. Tkinter є стандартною бібліотекою Python для створення графічного інтерфейсу. Вона дозволяє створювати вікна, кнопки, текстові поля

та інші елементи інтерфейсу. Tkinter також забезпечує можливість обробки подій, що є важливим для реалізації взаємодії користувача з додатком. Вибір цієї бібліотеки обумовлений її простотою у використанні та широкою підтримкою у спільноті розробників. Tkinter забезпечує гнучкість та простоту у створенні графічного інтерфейсу, що дозволяє швидко прототипувати та тестувати додаток.

Бібліотека pygame. Pygame використовується для додавання звукових сповіщень у додаток. Вона забезпечує прості у використанні методи для відтворення аудіофайлів, що дозволяє сповіщати користувача про наближення часу виконання завдання. Pygame також підтримує роботу з графікою, що може бути корисним для майбутнього розширення функціональності додатка. Бібліотека має широкий спектр можливостей для роботи зі звуком та графікою, що робить її корисною для створення інтерактивних елементів у додатку.

База даних SQLite. SQLite обрано для зберігання даних додатка. Це легка та швидка база даних, яка не потребує налаштування сервера. SQLite забезпечує надійне зберігання даних та простоту використання у невеликих додатках. Вона добре підходить для додатків з невеликою кількістю даних, де важлива швидкість та простота розгортання. SQLite має мінімальні вимоги до ресурсів і забезпечує високий рівень продуктивності для невеликих обсягів даних.

Алгоритми управління завданнями. Для реалізації функціоналу додавання, редагування та видалення завдань використовуються прості алгоритми роботи зі списками. Завдання зберігаються у списку, який оновлюється при додаванні або видаленні завдання. Для пошуку та редагування завдань використовуються лінійні алгоритми пошуку, що забезпечує швидкий доступ до даних та їх зміну. Ці алгоритми прості у реалізації та ефективні для невеликих обсягів даних, що робить їх ідеальними для використання в персональному планувальнику.

Алгоритми встановлення нагадувань. Для встановлення нагадувань використовується бібліотека datetime, яка дозволяє працювати з датами та часом. Нагадування встановлюються за допомогою методів, які порівнюють поточний

час з часом, вказаним у завданні. При досягненні заданого часу спрацьовує звукове сповіщення. Такий підхід дозволяє легко налаштовувати та керувати нагадуваннями. Використання datetime забезпечує точність та надійність роботи з датами та часом, що є критично важливим для функціоналу нагадувань.

Обробка подій. Tkinter забезпечує обробку подій, таких як натискання кнопок, введення тексту тощо. Це дозволяє реалізувати інтерактивний інтерфейс користувача, де кожна дія користувача викликає відповідну функцію обробки події. Такий підхід робить додаток більш динамічним та зручним у використанні. Обробка подій є ключовим елементом у створенні інтерактивного додатка, що дозволяє користувачеві легко взаємодіяти з програмою.

Збереження та завантаження даних. Для збереження даних про завдання використовується база даних SQLite. Дані зберігаються у вигляді таблиць, які містять інформацію про назву завдання, дату, час та стан виконання. При запуску додатка дані завантажуються з бази даних, що дозволяє зберігати стан завдань між сеансами роботи. Такий підхід забезпечує надійність та збереження інформації користувача. Використання SQLite для збереження даних забезпечує високий рівень продуктивності та надійності, що є важливим для персонального планувальника.

Вибір засобів, методів та алгоритмів обґрунтований їх відповідністю вимогам проекту, простотою використання та надійністю. Python, tkinter та pygame забезпечують швидку розробку та тестування додатка, а SQLite забезпечує надійне зберігання даних. Використання цих інструментів дозволяє створити функціональний та зручний персональний планувальник, який задовольняє вимоги користувачів щодо управління завданнями та встановлення нагадувань.

Обрані засоби та методи також забезпечують високу гнучкість та можливість подальшого розширення функціональності додатка. Це дозволяє легко адаптувати додаток до змінюваних вимог користувачів та додавати нові функції без значних витрат часу та ресурсів. Наприклад, використання Python

дозволяє інтегрувати додаткові бібліотеки для аналізу даних або створення звітів, що може бути корисним для користувачів, які бажають мати детальну інформацію про свою продуктивність.

Крім того, обрані засоби забезпечують високу кросплатформеність додатка. Python, tkinter та pygame підтримуються на різних операційних системах, що дозволяє використовувати додаток на будь-якому пристрої, незалежно від його операційної системи. Це робить додаток більш доступним для широкого кола користувачів.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ ДЛЯ ОСОБИСТОГО ПЛАНУВАННЯ ЧАСУ

3.1 Практична реалізація об’єкта проектування

Проектування інтерфейсу користувача:

a) вибір компонентів інтерфейсу:

– для створення інтерфейсу використано бібліотеку Tkinte, яка є стандартною для мови Python. Вона дозволяє створювати вікна, кнопки, текстові поля та інші елементи графічного інтерфейсу;

– інтерфейс містить такі основні елементи: поле введення для нових завдань, кнопку для додавання завдань, список для відображення завдань, а також кнопки для редагування та видалення завдань;

b) розміщення елементів інтерфейсу:

– поле введення завдань розміщене у верхній частині вікна додатка, під ним знаходяться кнопки для додавання та редагування завдань;

– список завдань розміщений у центральній частині вікна, що дозволяє легко переглядати та взаємодіяти з існуючими завданнями;

– кнопка видалення завдань розміщена під списком завдань для зручності користувача.

Реалізація функціональних можливостей:

1) додавання завдань:

– для додавання завдань використовується метод add_task, який зчитує введений користувачем текст та додає його до списку завдань;

– після додавання завдання, список оновлюється, а поле введення очищується для введення наступного завдання;

2) редагування завдань:

– реалізовано можливість вибору завдання зі списку та його редагування через відповідне поле введення;

– метод edit_task дозволяє змінювати текст обраного завдання та

зберігати зміни;

3) видалення завдань:

– видалення завдань здійснюється через метод delete_task, який видаляє обране завдання зі списку та оновлює відображення списку завдань;

4) створення звіту про виконані завдання лістинг 3.1:

– для створення звіту про виконані завдання використовується метод create_report, який зберігає список виконаних завдань у текстовий файл;

– кожне виконане завдання записується разом із часом його виконання що дозволяє користувачеві аналізувати свою продуктивність;

5) збереження та завантаження завдань лістинг 3.2:

– дані про завдання зберігаються у файлі за допомогою метод save_tasks, який записує кожне завдання у файл;

– метод load_tasks завантажує дані з файлу при запуску додатка, що дозволяє зберігати стан завдань між сеансами роботи.

Реалізація нагадувань:

1) встановлення нагадувань:

– для встановлення нагадувань використовується бібліотека datetime, яка дозволяє працювати з датами та часом;

– користувач може вибрати дату та час нагадування, які зберігаються разом із завданням;

2) звукові сповіщення:

– звукові сповіщення реалізовано за допомогою бібліотеки pygame, яка дозволяє відтворювати аудіофайли;

– при настанні часу нагадування, спрацьовує звуковий сигнал, що привертає увагу користувача до завдання.

Обробка подій:

1) взаємодія з користувачем:

– Tkinter забезпечує обробку подій, таких як натискання кнопок та введення тексту, що дозволяє створити інтерактивний інтерфейс користувача;

– кожна дія користувача викликає відповідну функцію обробки події, що забезпечує динамічність та зручність у використанні додатка.

Лістинг 3.1 – Код для створення звіту про виконані завдання

```
import tkinter as tk  # Імпорт бібліотеки tkinter для створення графічного інтерфейсу
from datetime import datetime  # Імпорт модуля datetime для роботи з датою і часом
class PersonalPlanner:
    def __init__(self, root):
        self.root = root  # Основне вікно додатка
        self.tasks = []  # Список для зберігання завдань
        self.completed_tasks = []  # Список для зберігання виконаних завдань
        self.task_entry = tk.Entry(root)  # Поле введення для нового завдання
        self.task_entry.pack()  # Додавання поля введення до основного вікна
        self.add_button = tk.Button(root, text="Додати завдання", command=self.add_task)  # Кнопка для додавання завдання
        self.add_button.pack()  # Додавання кнопки до основного вікна
        self.task_listbox = tk.Listbox(root)  # Список для відображення завдань
        self.task_listbox.pack()  # Додавання списку до основного вікна
        self.complete_button = tk.Button(root, text="Виконати завдання", command=self.complete_task)  # Кнопка для відмітки виконання завдання
        self.complete_button.pack()  # Додавання кнопки до основного вікна
        self.report_button = tk.Button(root, text="Створити звіт", command=self.create_report)  # Кнопка для створення звіту
        self.report_button.pack()  # Додавання кнопки до основного вікна
```

Кінець лістингу 3.1

Лістинг 3.2 – Основний код для збереження та завантаження завдань

```
import tkinter as tk  # Імпорт бібліотеки tkinter для створення графічного інтерфейсу

class PersonalPlanner:

    def __init__(self, root):
        self.root = root  # Основне вікно додатка
        self.tasks = []  # Список для зберігання завдань
        self.task_entry = tk.Entry(root)  # Поле введення для нового завдання
        self.task_entry.pack()  # Додавання поля введення до основного вікна
        self.add_button = tk.Button(root, text="Додати завдання", command=self.add_task)  # Кнопка для додавання завдання
        self.add_button.pack()  # Додавання кнопки до основного вікна
        self.task_listbox = tk.Listbox(root)  # Список для відображення завдань
        self.task_listbox.pack()  # Додавання списку до основного вікна
        self.save_button = tk.Button(root, text="Зберегти завдання", command=self.save_tasks)  # Кнопка для збереження завдань
        self.save_button.pack()  # Додавання кнопки до основного вікна
        self.load_button = tk.Button(root, text="Завантажити завдання", command=self.load_tasks)  # Кнопка для завантаження завдань
        self.load_button.pack()  # Додавання кнопки до основного вікна
```

Кінець лістингу 3.2

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Методологія тестування:

1) модульне тестування лістинг 3.3:

a) опис: модульне тестування включає перевірку окремих модулів або компонентів системи для виявлення помилок у їхній роботі [9];

b) інструменти: для модульного тестування використовувалися бібліотеки unittest та pytest, які дозволяють створювати та виконувати тестові сценарії для окремих функцій та класів.

Лістинг 3.3 – Модульне тестування

```
import unittest  # Імпорт бібліотеки unittest для створення тестів
from personal_planner import PersonalPlanner  # Імпорт класу PersonalPlanner з файлу personal_planner
class TestPersonalPlanner(unittest.TestCase):  # Створення класу для тестування, який наслідує unittest.TestCase
    def setUp(self):  # Метод, який виконується перед кожним тестом
        self.app = PersonalPlanner()  # Ініціалізація об'єкта класу PersonalPlanner
    def test_add_task(self):  # Тестовий метод для перевірки додавання завдання
        self.app.add_task("New Task")  # Додавання нового завдання
        self.assertIn("New Task", self.app.tasks)  # Перевірка, чи додане завдання є в списку завдань
    def test_edit_task(self):  # Тестовий метод для перевірки редагування завдання
        self.app.add_task("Task to Edit")  # Додавання завдання, яке буде редагуватися
        self.app.edit_task(0, "Edited Task")  # Редагування першого завдання у списку
        self.assertEqual(self.app.tasks[0], "Edited Task")  # Перевірка, чи завдання було успішно відредаговано
    def test_delete_task(self):  # Тестовий метод для перевірки видалення завдання
```

```
        self.app.add_task("Task to Delete")  # Додавання завдання, яке буде видалено
        self.app.delete_task(0)  # Видалення першого завдання у списку
        self.assertNotIn("Task to Delete", self.app.tasks)  # Перевірка, чи завдання було успішно видалено
    if __name__ == '__main__':  # Перевірка, чи запускається скрипт безпосередньо
        unittest.main()  # Запуск тестів
```

Кінець лістингу 3.3

Результати модульного тестування:

– тест на додавання завдання: «Успішно»;

– тест на редагування завдання: «Успішно»;

– тест на видалення завдання: «Успішно»;

– виявлені помилки: під час тестування виявлено, що при додаванні завдань з пустими назвами виникала помилка, яку було виправлено шляхом додавання перевірки на пустий рядок;

2) інтеграційне тестування лістинг 3.4:

a) опис: інтеграційне тестування включає перевірку взаємодії між різними модулями системи для виявлення проблем у їхній інтеграції;

b) інструменти: використовувалися ті ж інструменти, що і для модульного тестування, з додатковою перевіркою взаємодії між модулями.

Лістинг 3.4 – Інтеграційне тестування

```
    import unittest  # Імпорт бібліотеки unittest для створення тестів
    from personal_planner import PersonalPlanner  # Імпорт класу PersonalPlanner з файлу personal_planner
    class TestIntegration(unittest.TestCase):  # Створення класу для інтеграційного тестувння, який наслідує unittest.TestCase
        def setUp(self):  # Метод, який виконується перед кожним тестом
            self.app = PersonalPlanner()  # Ініціалізація об'єкта класу PersonalPlanner
```

```
    def test_task_flow(self):  # Тестовий метод для перевірки повного потоку роботи з завданням
        self.app.add_task("Integrated Task")  # Додавання нового завдання
        self.app.edit_task(0, "Edited Integrated Task")  # Редагування першого завдання у списку
        self.app.complete_task(0)  # Позначення першого завдання як виконаного
        self.assertIn("Edited Integrated Task", [task[0] for task in self.app.completed_tasks])  # Перевірка, чи відредаговане завдання є у списку виконаних завдань
if __name__ == '__main__':  # Перевірка, чи запускається скрипт безпосередньо
    unittest.main()  # Запуск тестів    unittest.main()
```

Кінець лістингу 3.4

Результати інтеграційного тестування:

– тест на повний цикл завдання (додавання, редагування, виконання): «Успішно»;

– виявлені помилки: при інтеграційному тестуванні було виявлено помилку при передачі індексів завдань між модулями. Помилку було виправлено шляхом коректного оброблення індексів;

3) функціональне тестування лістинг 3.5:

a) опис: тестування включає перевірку функціональності системи відповідно до вимог користувачів;

b) інструменти: використовувалися ручні методи тестування, а також автоматизовані тести за допомогою бібліотеки selenium.

Лістинг 3.5 – Функціональне тестування

```
from selenium import webdriver  # Імпорт бібліотеки Selenium для автоматизації браузера
import unittest  # Імпорт бібліотеки unittest для створення тестів
class TestFunctional(unittest.TestCase):  # Створення класу для функціонального тестування, який наслідує unittest.TestCase
    def setUp(self):  # Метод, який виконується перед кожним тестом
        self.driver = webdriver.Chrome()  # Ініціалізація веб-драйвера для браузера Chrome
        self.driver.get("http://localhost:8000")  # Перехід на локальну версію додатка
    def test_add_task(self):  # Тестовий метод для перевірки додавання завдання
        driver = self.driver  # Отримання драйвера
        task_input = driver.find_element_by_id("task_input")  # Знаходження елемента введення завдання за його ID
        task_input.send_keys("Functional Test Task")  # Введення тексту завдання
        add_button = driver.find_element_by_id("add_button")  # Знаходження кнопки додавання завдання за її ID
        add_button.click()  # Натискання на кнопку додавання завдання
        self.assertIn("Functional Test Task", driver.page_source)  # Перевірка, чи додане завдання присутнє в джерелі сторінки
    def tearDown(self):  # Метод, який виконується після кожного тесту
        self.driver.quit()  # Закриття браузера
if __name__ == "__main__":  # Перевірка, чи запускається скрипт безпосередньо
    unittest.main()  # Запуск тестів    unittest.main()
```

Кінець лістингу 3.5

Результати функціонального тестування:

– тест на додавання завдання: «Успішно»;

– виявлені помилки: при функціональному тестуванні було виявлено, що інтерфейс іноді не оновлювався після додавання завдання. Помилку було виправлено шляхом додавання виклику методу оновлення інтерфейсу після кожної зміни.

Налагодження:

1) інструменти налагодження:

– для налагодження системи використовувалися стандартні інструменти налагодження, що входять до складу інтегрованих середовищ розробки (IDE), таких як PyCharm та Visual Studio Code;

– використання інструментів налагодження дозволило виявити та виправити логічні помилки в коді, а також оптимізувати його роботу;

2) процес налагодження:

– виявлення помилок: під час тестування були виявлені різні помилки, включаючи логічні помилки та проблеми з інтеграцією модулів;

– виправлення помилок: всі виявлені помилки були виправлені за допомогою інструментів налагодження, що дозволило забезпечити стабільну роботу системи;

– оптимізація: після виправлення помилок була проведена оптимізація коду для покращення продуктивності системи.

Проведення тестування та налагодження дозволило забезпечити високу якість та стабільність роботи інформаційно-комп'ютерної системи. Використання різних методів тестування дозволило виявити та виправити помилки на ранніх етапах розробки, що значно покращило продуктивність та надійність системи. Виявлені під час тестування помилки були успішно виправлені, що дозволило підвищити стабільність та функціональність системи.

3.3 Візуалізація роботи додатка

В даному розділі представлено візуалізацію роботи персонального планувальника, розробленого для організації особистого часу користувачів. Додаток забезпечує інтуїтивно зрозумілий інтерфейс, який дозволяє легко додавати, редагувати, видаляти та переглядати завдання, а також налаштовувати нагадування про важливі події.

Інтерфейс користувача складається з декількох основних елементів:

– поле введення завдання: дозволяє користувачам вводити текст завдання [10];

– поле вибору дати: забезпечує можливість вибору дати виконання завдання за допомогою календаря;

– поле вибору часу: дозволяє вказати точний час виконання завдання;

– кнопки управління завданнями: дозволяють додавати нові завдання, редагувати існуючі, видаляти вибрані завдання, зберігати завдання у файл та завантажувати їх з файлу;

– список завдань: відображає всі додані завдання із зазначенням дати та часу їх виконання;

– перемикач теми: дозволяє змінювати тему інтерфейсу між світлою та темною.

Основні функціональні можливості:

1) додавання завдання. Користувач може ввести текст завдання у відповідне поле, вибрати дату та час виконання, і натиснути кнопку «Додати завдання». Завдання буде додано до списку завдань та збережено у файлі;

2) редагування завдання. Для редагування завдання необхідно вибрати його у списку завдань, після чого його дані з’являться у відповідних полях введення. Після внесення змін користувач може натиснути кнопку «Редагувати завдання», щоб оновити інформацію про завдання;

3) видалення завдання. Користувач може вибрати завдання у списку та натиснути кнопку «Видалити вибране завдання», щоб видалити його зі списку та з файлу;

4) збереження та завантаження завдань. Кнопки «Зберегти завдання» та «Завантажити завдання» дозволяють відповідно зберігати поточний список завдань у файл та завантажувати його з файлу;

5) нагадування про завдання. Додаток регулярно перевіряє поточний час та порівнює його з часом виконання завдань. Якщо час виконання завдання співпадає з поточним часом, додаток відтворює звукове нагадування та відображає повідомлення;

6) перемикання теми. Користувач може перемикати тему інтерфейсу між світлою та темною за допомогою відповідного перемикача.

Нижче наведено приклади основних вікон та елементів інтерфейсу користувача: головне вікно додатка рисунок 3.1 (світла тема); головне вікно додатка рисунок 3.2 (темна тема); додавання завдання рисунок 3.3; редагування завдання рисунок 3.4; список завдань рисунок 3.5; нагадування про завдання рисунок 3.6.

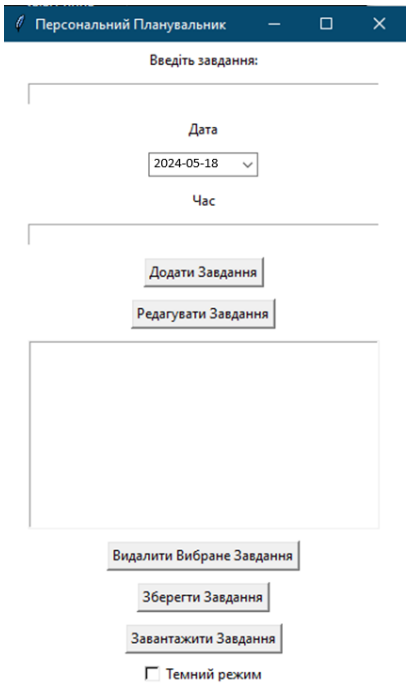

Рисунок 3.1 – Головне вікно додатка (світла тема)

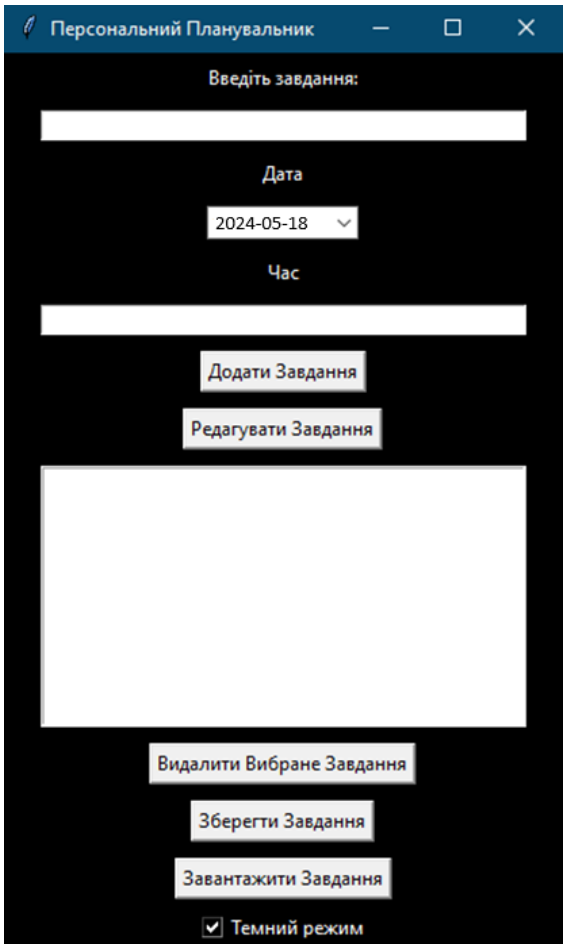

Рисунок 3.2 – Головне вікно додатка (темна тема)

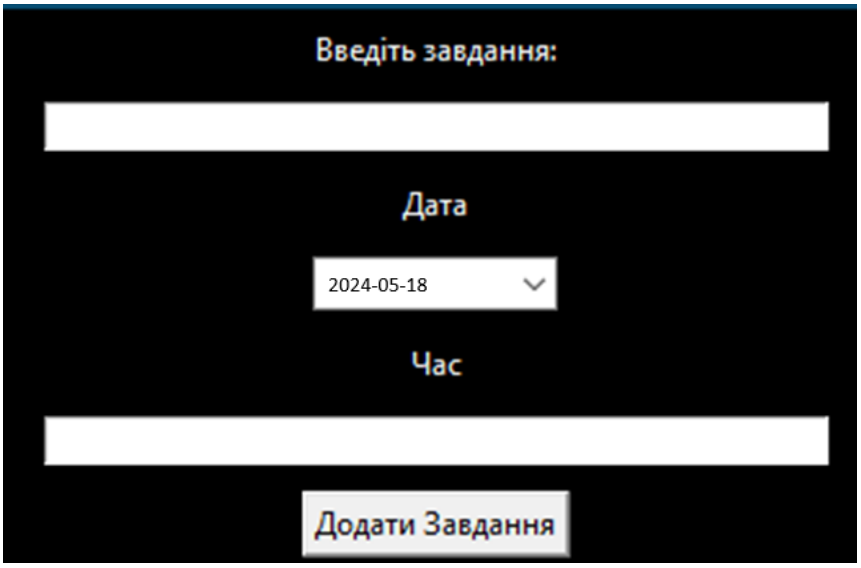

Рисунок 3.3 – Додавання завдання

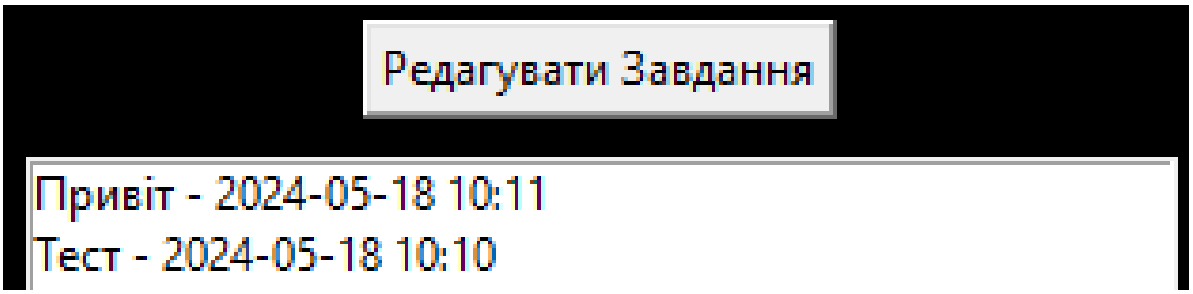

Рисунок 3.4 – Редагування завдання

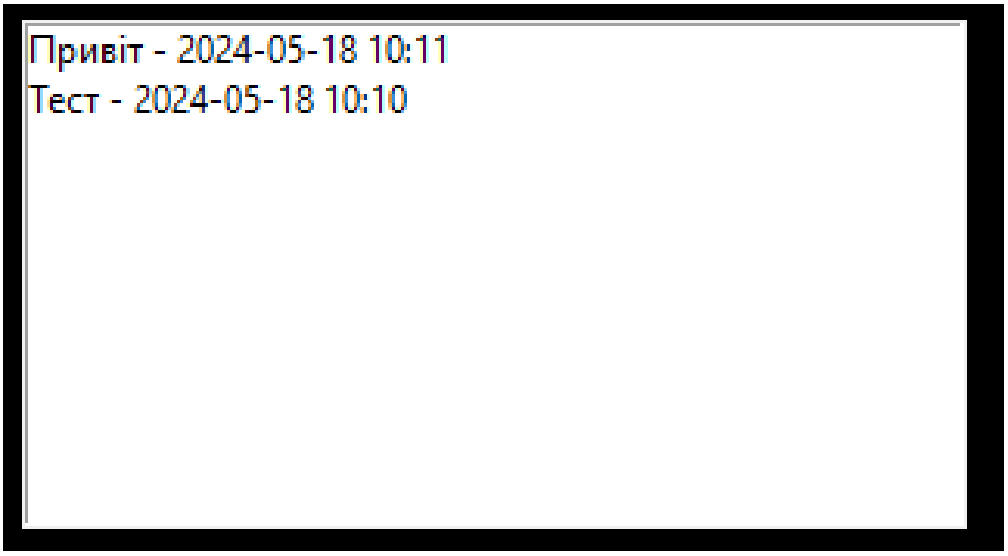

Рисунок 3.5 – Список завдань

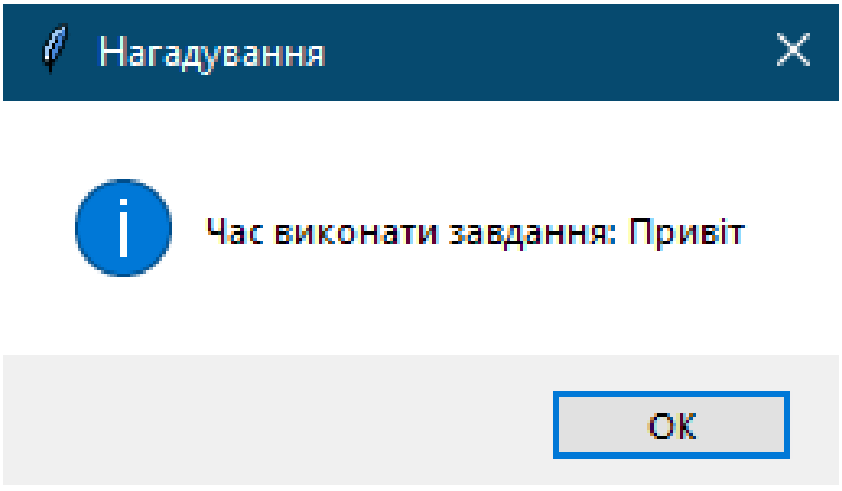

Рисунок 3.6 – Нагадування про завдання

Ці приклади наочно демонструють основні можливості додатка та його інтерфейс, що забезпечує зручність та ефективність у плануванні особистого часу.

Персональний планувальник для управління часом, розроблений у рамках даної роботи, надає користувачам зручні та ефективні інструменти для організації завдань. Інтуїтивно зрозумілий інтерфейс та багатий функціонал роблять цей додаток корисним у повсякденному житті, дозволяючи з легкістю керувати своїм часом та завданнями.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи був створений додаток для особистого планування часу, який забезпечує зручне управління завданнями, нагадуваннями та ефективну організацію часу користувачів.

Для досягнення поставленої мети були виконані наступні завдання: аналіз існуючих рішень у сфері планування особистого часу: проведено огляд сучасних додатків, таких як Todoist, Trello, Microsoft To Do та Google Keep; визначено основні функціональні можливості, включаючи створення завдань, встановлення нагадувань, інтеграцію з календарем, синхронізацію між пристроями, можливість роботи офлайн та наявність мобільної версії; виявлено сильні сторони конкурентних продуктів, такі як інтуїтивно зрозумілий інтерфейс, широкі можливості кастомізації, багатофункціональність та висока надійність; виявлено слабкі сторони конкурентних продуктів, включаючи відсутність деяких специфічних функцій, обмежену підтримку різних платформ, відсутність локалізації для деяких мов та високу вартість преміум-версій.

Визначення вимог до функціональності та дизайну додатку: сформульовано вимоги до функціональності додатку, включаючи управління завданнями, встановлення нагадувань, збереження та завантаження даних, можливість синхронізації з календарями та інтеграції з іншими сервісами; описано архітектурні підходи та структуру даних для забезпечення стабільної роботи додатку.

Вибір технологій та інструментів для розробки додатку: обрано Python, як основну мову програмування завдяки її простоті та широким можливостям; використано бібліотеки Tkinter для створення графічного інтерфейсу та Pygame для звукових сповіщень, що забезпечило кросплатформенність додатку; для зберігання даних використано SQLite, що дозволяє зберігати інформацію локально та забезпечує швидкий доступ до даних.

Розробка прототипу додатку та його тестування: створено прототип користувальницького інтерфейсу, який дозволив отримати зворотний зв’язок від потенційних користувачів та внести необхідні корективи; виконано тестування прототипу для виявлення та виправлення помилок.

Реалізація функціональних можливостей додатку: реалізовано основні функції додатку, включаючи додавання, редагування та видалення завдань, встановлення нагадувань, збереження та завантаження завдань; забезпечено можливість перемикання теми інтерфейсу між світлою та темною.

Тестування та налагодження програмного забезпечення – проведено модульне, інтеграційне та функціональне тестування додатку; виявлені під час тестування помилки були успішно виправлені, що забезпечило стабільність та надійність системи.

Написання технічної та користувацької документації – підготовлено детальну технічну документацію, яка описує архітектуру додатку, його функціональні можливості, інструкції з установки та використання, а також керівництво для подальшого розвитку та підтримки системи.

У підсумку, поставлені цілі досягнуті шляхом створення додатку для особистого планування часу, який забезпечує зручне управління завданнями, підвищує продуктивність і задоволеність користувачів, дозволяючи ефективно планувати та виконувати завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Попов О. В., Гнатюк Л. В. Місце процесу управління проектами в стратегії розвитку підприємств. Журнал «Управління проектами». 2021. С. 88- 90.

2. Todoist. URL: https://todoist.com/ (дата звернення: 06.04.2024).

3. Trello. URL: https://trello.com/uk (дата звернення: 10.05.2024).

4. Microsoft To Do. URL: https://to-do.office.com/tasks (дата звернення: 11.05.2024).

5. Google Keep. URL: https://keep.google.com (дата звернення: 16.05.2024).

6. Covey, Stephen R. "The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change." Simon and Schuster, 2020. P. 45-48.

7. Allen, David. "Getting Things Done: The Art of Stress-Free Productivity." Penguin Books, 2021. P. 98-101.

7. SQLite. URL: https://sqlite.org (дата звернення: 07.05.2024).

9. Goleman, Daniel. "Emotional Intelligence: Why It Can Matter More Than IQ." Bantam Books, 2021. P. 67-70.

10. Covey, Stephen R. "The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change." Simon and Schuster, 2020. P. 45-48.