

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Програмний засіб керування часом та управління задачами

Здобувача вищої освіти першого (бакалаврського)
рівня,
4 курсу, групи КБ-19-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за
галузями)»


(підпис)

Коверзнева Андрія
Володимировича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Ясенова Ірина Сергіївна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

✓ Керівник освітньо-
професійної програми


(підпис)

Федотова - Півень Ірина
Миколаївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «26» травня 2023 № 13

Завідувач кафедри


(підпис)

Філоненко Михайло Миколайович,
к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене
звання)

Ірпінь – 2023

Реєстрація № 21

06.06.2023
(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

29.05.2023
(дата)



(підпис)

Результат захисту

5 (відмінно)
(оцінка)

14.06.2023
(дата)



(підпис)

Голова комісії:



(підпис)

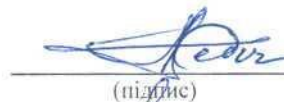
В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

О.В. Редич



(підпис)

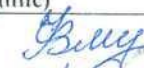
Є.В. Котух

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		14.06.2023 Франчук В.М.  Філоненко М.М.  Редич О.В.  Котух Є.В.  Суховерхівка Л.П. 
2. Дотримання вимог щодо оформлення	від 0 до 10	8		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником - 46 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підписи)
91	A	5 (відмінно)	14.06.2023	Франчук В.М.  Філоненко М.М.  Редич О.В.  Котух Є.В.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	3 (задовільно)
60-69	D	2 (незадовільно)
50-59	E	
35-49	FX	

Факультет фінансів та цифрових технологій

Кафедра комп'ютерних та інформаційних технологій і систем

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри

 Філоненко М.М.
«16» 11 2023

Завдання на кваліфікаційну роботу

здобувачу вищої освіти

Коверзнев Андрій Володимирович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Програмний засіб керування часом та управління задачами

Затверджена наказом ректора від «15» листопада 2022 № 1205 (зі змінами наказ від «04» квітня 2023 № 361)

2. Строк здачі закінченої роботи «26» травня 2023

3. Цільова установка Розробка програмного засобу керування часом та управління задачами.

4. Мета роботи Мета роботи полягає в розробці ефективного програмного засобу, який допоможе користувачам ефективно керувати своїм часом і управляти різноманітними завданнями

5. Об'єкт дослідження Об'єктом дослідження є процес управління часом та управління задачами. В рамках дослідження розглядаються різні підходи до цих процесів, а також способи їх оптимізації і поліпшення.

6. Предмет дослідження Предметом дослідження є програмний засіб, що дозволяє керувати часом та управляти задачами.

7. База дослідження Базою дослідження виступає літературні джерела, наукові публікації, а також існуючі програмні засоби керування часом та управління задачами для проведення порівняльного аналізу.

8. Вихідні дані до роботи Дослідження подібних розробок в сфері керування часом.

9. Перелік графічного матеріалу Рис.1. Імпортовані бібліотеки, Рис.2. Константи, Рис.3. Ініціалізація даних, Рис.4. Команда start, Рис.16. Виправлення помилки з неправильним форматом.

10. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом) 1. Аналіз теми дипломної роботи, 2. Етапи розробки програмного засобу керування часом та управління задачами, 3. Розробка програмного засобу керування часом та управління задачами.

11. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2022	15.11.2022
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	01.02.2023	01.02.2023
3	II розділ (аналіз експериментальної частини про методики досліджень)	03.04.2023	04.04.2023
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.05.2023	09.05.2023
5	Висновки	15.05.2022	15.05.2022
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.05.2023	26.05.2023
7	Отримання відгуку керівника	06.06.2023	06.06.2023
8	Отримання зовнішньої рецензії	08.06.2023	08.06.2023

8. Дата видачі завдання «16» листопада 2022

9. Керівник випускної кваліфікаційної роботи


 (підпис)

Ясенова
 (прізвище, іні)

10. Керівник освітньо-професійної програми


 (підпис)

Федотов
 (прізвище, іні)

11. Завдання до виконання отримав здобувач вищої освіти


 (підпис)

Коверзне
 (прізвище, іні)

АНОТАЦІЯ

Програмний засіб керування часом та управління задачами

Мета роботи полягає в розробці ефективного програмного засобу, який допоможе користувачам ефективно керувати своїм часом і управляти різноманітними завданнями.

Завданнями роботи є аналіз існуючих програмних рішень для керування часом, розробка та реалізація нового програмного засобу, тестування та оцінка його функціональності.

Предметом дослідження є програмний засіб, що дозволяє керувати часом та управляти задачами.

Об'єктом дослідження є процес управління часом та управління задачами. В рамках дослідження розглядаються різні підходи до цих процесів, а також способи їх оптимізації і поліпшення.

Базою дослідження виступає літературні джерела, наукові публікації, а також існуючі програмні засоби керування часом та управління задачами для проведення порівняльного аналізу.

За результатами дослідження сформульовані пропозиції щодо розробки нового програмного засобу, який враховує недоліки існуючих рішень і пропонує нові функціональні можливості.

Одержані результати можуть бути використані різними користувачами, такими як студенти, працівники офісів, менеджери та інші особи, які прагнуть покращити свою продуктивність, ефективно керувати своїм часом і успішно виконувати завдання.

Кваліфікаційна робота містить 57 сторінок, 0 таблиць, 16 рисунків, список використаних джерел 40 найменувань, додатків.

Ключові слова: Бот, Телеграм, програмний засіб, продукт, проєкт.

Рік виконання кваліфікаційної роботи 2023

Рік захисту роботи 2023

ANNOTATION

A software tool for time management and task management

The aim of the work is to develop an effective software tool that will help users to manage their time and manage various tasks.

The tasks are to analyze existing time management software solutions, develop and implement a new software tool, test and evaluate its functionality.

The subject of the study is a software tool that allows you to manage time and manage tasks.

The object of research is the process of time management and task management. The study considers various approaches to these processes, as well as ways to optimize and improve them.

The research is based on literature sources, scientific publications, as well as existing time management and task management software tools for comparative analysis.

Based on the results of the study, proposals for the development of a new software tool that takes into account the shortcomings of existing solutions and offers new functionality are formulated.

The results obtained can be used by various users, such as students, office workers, managers, and others who seek to improve their productivity, manage their time effectively, and successfully complete tasks.

The qualification work contains 57 pages, 0 tables, 16 figures, a list of 40 references, and appendices.

Keywords: Bot, Telegram, software, product, project.

Year of completion of the qualification work 2023

Year of thesis defense 2023

ЗМІСТ

Вступ	9
1. Аналіз теми дипломної роботи.....	12
1.1. Постановка проблеми.....	12
1.2. Цільова аудиторія продукту.....	13
1.3. Необхідність продукту.....	14
2. Етапи розробки програмного засобу керування часом та управління задачами.....	16
2.1. Аналіз подібних розробок.....	16
2.2. Опис мінімально життєздатного продукту.....	22
2.3. Вибір платформи.....	23
3. Розробка програмного засобу керування часом та управління й задачами.....	27
3.1. Розробка прототипу.....	27
3.2. Тестування продукту.....	40
3.3. Подальший розвиток проєкту.....	44
Висновок	47
Список використаних джерел.....	48
Додатки.....	50

ВСТУП

Актуальність теми:

У сучасному світі, де швидкість та обсяг інформації постійно зростають, ефективне управління часом та задачами стає все важливішим. Люди шукають способи оптимізувати свою продуктивність, забезпечити виконання поставлених цілей та ефективно розподілити свій час між різними завданнями. В цьому контексті програмний засіб керування часом та управління задачами є необхідним інструментом для досягнення успіху в особистому та професійному житті.

Стан і ступінь розробки проблеми:

Проблема керування часом та управління задачами існує давно, і люди використовують різні методи та стратегії для її вирішення. Традиційні паперові планери, щоденники та список завдань допомагали в структуруванні часу і задач. Однак, з появою комп'ютерів та розвитком інформаційних технологій, з'явилися програмні засоби, які надають нові можливості для керування часом та управління задачами. Проте, необхідно провести дослідження та розробити програмний засіб, який відповідає потребам сучасного користувача і забезпечує максимальну продуктивність.

Мета і завдання дослідження:

Метою даного дослідження є розробка програмного засобу, який надає ефективні інструменти для керування часом та управління задачами. Основними завданнями дослідження є аналіз потреб користувачів у сфері керування часом, вивчення існуючих програмних рішень, розробка нового програмного засобу та його тестування.

Об'єкт дослідження:

Об'єктом дослідження є процес керування часом та управління задачами у реальному часі, а також потреби та вимоги користувачів щодо програмних засобів у цій сфері.

Предмет дослідження:

Предметом дослідження є програмний засіб керування часом та управління задачами, який має на меті надати користувачам інструменти для ефективного планування, організації та виконання завдань, а також керування власним часом.

Методи дослідження:

Для досягнення поставлених цілей дослідження будуть використовуватись наступні методи:

- Аналіз літературних джерел та наукових публікацій з питань керування часом та управління задачами.
- Аналіз існуючих програмних рішень та їхніх функціональних можливостей.
- Розробка програмного засобу на основі зібраних даних.
- Тестування програмного засобу для перевірки його функціональності.

База досліджень:

В ході дослідження будуть використані літературні джерела, наукові публікації, а також існуючі програмні засоби керування часом та управління задачами для проведення порівняльного аналізу.

Науково-практична значимість одержаних результатів:

Результати даного дослідження матимуть важливу науково-практичну значимість. Розроблений програмний засіб надасть користувачам зручні та ефективні інструменти для керування часом та управління задачами, що дозволить підвищити продуктивність та досягти поставлених цілей. Отримані результати також можуть бути використані для подальших досліджень у сфері програмних засобів керування часом та управління задачами.

Особистий внесок здобувача:

Провів аналіз літературних джерел, зібрав та проаналізував дані відповідно до поставлених завдань, розробив програмний засіб, провів його тестування.

Апробація результатів роботи:

Результати роботи можуть бути опубліковані у наукових журналах для широкого доступу та використання у науковому та практичному середовищі.

1. Аналіз теми дипломної роботи

1.1. Постановка проблеми

На сьогоднішній день керування часом та задачами критично важливі аспекти нашого життя. Управління завданнями та ефективне планування часу може бути досить складним викликом, особливо для людей, які повинні одночасно виконувати багато завдань. У таких ситуаціях програмний засіб керування часом та управління задачами може бути дуже корисним. Цей інструмент допомагає ефективно керувати завданнями, тим самим сприяючи досягненню поставлених цілей.

Основна проблема, яку вирішує програмний засіб керування часом та управління задачами - це ефективне управління тими самими завданнями, також однією з проблем, з якою стикаються люди, є нездатність управляти кількома задачами одночасно. У таких ситуаціях програмний засіб керування часом та управління задачами надає централізовану платформу, на якій користувачі можуть планувати задачі, встановлювати їх пріоритетність та нагадування про необхідність завершення вчасно.

При роботі над кількома завданнями важко відстежувати прогрес і визначати, які завдання потребують негайної уваги. Програмний засіб дозволяє користувачам відстежувати прогрес і переглядати завдання.

Крім того, цей засіб розв'язує проблему пріоритетів. У більшості випадків люди схильні зосереджуватися на термінових завданнях, а не на важливих. Цей підхід може призвести до знехтування критичними завданнями, які потребують уваги. Програмний засіб керування часом та управління задачами допомагає користувачам надавати пріоритети завданням залежно від їх важливості та терміновості, забезпечуючи їх вчасне та ефективне виконання.

Отже, програмний засіб керування часом та управління задачами є необхідним інструментом для кожного, хто бажає покращити свою продуктивність та ефективно керувати своїм часом. Цей засіб допомагає

управляти багатьма задачами, відстежувати прогрес та задавати пріоритети для завдань, що допомагає досягати поставлених цілей. За допомогою цього засобу люди можуть працювати більш ефективно, що призводить до успіху як в особистому, так і в професійному житті.

1.2. Цільова аудиторія продукту

Програмний засіб керування часом та управління завданнями є корисним інструментом для осіб, які бажають підвищити свою продуктивність та ефективність. Цей програмний засіб надає централізовану платформу для управління багатьма завданнями, ефективного планування часу та надає можливість задавати пріоритети роботі на основі важливості та терміновості.

Цей інструмент корисний для людей, які мають багато відповідальностей та мають проблеми з ефективним управлінням часом. Це можуть бути студенти, які повинні балансувати між навчанням, завданнями та позаурочними заходами, або фахівці, які мають керувати кількома проєктами та дедлайнами одночасно. Крім того, люди, які працюють на відстані або мають гнучкий графік роботи, можуть відчувати складнощі з ефективним управлінням часом, тому програма для управління часом та завданнями є відмінним рішенням.

Крім того, підприємці та власники невеликих бізнесів також можуть скористатися цим інструментом. Зазвичай вони повинні виконувати декілька завдань одночасно, їх час часто розділяється між управлінням бізнесом, спілкуванням з клієнтами та виконанням різних завдань. З допомогою цього застосунку вони можуть пріоритезувати завдання, ефективно планувати час та відстежувати прогрес, щоб забезпечити вчасне виконання всіх завдань.

Окрім цього, люди, які мають проблеми з відкладанням справ та розсіюванням уваги, також можуть скористатися програмою для управління часом та завданнями. Цей інструмент допомагає розбити завдання на менші, керовані кроки, тим самим зменшуючи відчуття перевантаження та полегшуючи фокус на окремих завданнях.

Інструменти управління часом та задачами є корисними для будь-якої людини, яка бажає підвищити свою продуктивність, ефективно керувати своїм часом та досягати своїх цілей. Цей інструмент може бути корисним для всіх людей, особливо для студентів, підприємців та осіб, які мають проблеми з прокрастинацією та розсіяністю.

1.3. Необхідність продукту

Програмне забезпечення для управління часом та завданнями є корисним інструментом для людей, які хочуть покращити свою продуктивність та ефективність. Цей інструмент забезпечує централізовану платформу для управління багатьма завданнями, ефективного планування часу та пріоритезації роботи на основі важливості та терміновості.

В першу чергу, програмний інструмент управління часом та завданнями допомагає вам пріоритезувати завдання на основі їх важливості та терміновості. Це допомагає фокусуватись на найважливіших завданнях першими та забезпечує їх своєчасне виконання.

По-друге, програмний інструмент управління часом та завданнями допомагає вам залишатися організованими. З цим інструментом ви можете зберігати всі свої завдання в одному центральному місці, що дозволяє легше відстежувати прогрес та переконатися, що нічого не було пропущено. Це допомагає зменшити стрес та дозволяє фокусуватися на виконанні завдань, а не на тому, що ще потрібно зробити.

По-третє, програмний інструмент управління часом та завданнями може допомогти вам керувати своїм часом більш ефективно. Відстежуючи час, який ви витрачаєте на кожне завдання, ви можете визначити області, де ви витрачаєте час даремно та вносити необхідні корективи. Це може допомогти вам працювати більш ефективно та виконувати завдання за менший час.

Нарешті, інструмент допоможе вам досягнути своїх цілей. Встановлюючи цілі та відстежуючи прогрес, ви можете побачити, наскільки далеко ви зайшли і

що потрібно зробити, щоб досягти своїх цілей. Це допомагає зберегти мотивацію і фокусуватися на поточній задачі.

В підсумку, цей проєкт буде корисним для кожного, хто бажає ефективно управляти своїм часом, пріоритезувати завдання, бути організованим і досягати своїх цілей. За допомогою цього інструменту ви можете працювати більш ефективно, зменшувати стрес та досягати більшого успіху як в особистому, так і в професійному житті. Незалежно від того, чи ви студент, підприємець або звичайна людина. Інструмент управління часом та задачами може допомогти вам ефективно управляти своїм часом та реалізувати свій потенціал.

2. Етапи розробки програмного засобу керування часом та управління задачами

2.1. Аналіз подібних розробок

Існує безліч програмних засобів для керування часом та управління задачами, але ми розглянемо декілька з них:

1. Todoist - це онлайн-додаток для керування задачами, який дозволяє створювати списки завдань, планувати їх за датою та пріоритетом, додавати коментарі та використовувати інші корисні функції.

Плюси:

- *Простий та зрозумілий інтерфейс.* Todoist має дуже простий та зрозумілий інтерфейс, що дозволяє легко створювати, редагувати та організовувати завдання.
- *Можливість додавати завдання швидко та легко.* Todoist дозволяє додавати нові завдання швидко та легко з будь-якої сторінки веб-браузера, за допомогою віджетів, електронної пошти, чат-ботів та інших інтеграцій.
- *Можливість встановлювати терміни та пріоритети.* Todoist дозволяє встановлювати терміни та пріоритети для завдань, що допомагає організовувати свій час та працювати більш ефективно.
- *Інтеграції з іншими сервісами.* Todoist має інтеграції з багатьма іншими сервісами, такими як Google Calendar, Slack, Dropbox, Evernote та багато інших.

Мінуси:

- *Обмежена безкоштовна версія.* Todoist має обмежену безкоштовну версію, що не дозволяє використовувати всі функції, такі як нагадування, мітки, пріоритети та інші.

- *Вартість підписки.* Повна функціональність Todoist доступна лише з підпискою, що може бути вартістю, залежно від того, як часто ви плануєте використовувати додаток та які функції вам необхідні.

- *Відсутність інтеграції з деякими сервісами.* Todoist не має інтеграцій з деякими сервісами, що може бути незручним для користувачів, які хочуть використовувати Todoist разом зі своїми улюбленими сервісами.

Todoist є корисним інструментом для керування задачами та організації робочого часу. Додаток має безліч корисних функцій та інтеграцій, які допоможуть вам більш ефективно працювати. Однак, є обмеження, які варто враховувати при виборі Todoist або будь-якого іншого програмного засобу для керування задачами.

2. Trello - це інтерактивна дошка, яка дозволяє створювати картки з завданнями та перетягувати їх між колонками, щоб відслідковувати прогрес роботи.

Плюси:

- *Простий та зрозумілий інтерфейс.* Trello має дуже простий та зрозумілий інтерфейс, що дозволяє легко створювати, редагувати та організовувати завдання.

- *Можливість створювати дошки та списки.* Trello дозволяє створювати дошки та списки, які допомагають організовувати проекти та завдання візуально.

- *Можливість додавати завдання швидко та легко.* Trello дозволяє додавати нові завдання швидко та легко з будь-якої сторінки веб-

браузера, за допомогою віджетів, електронної пошти, чат-ботів та інших інтеграцій.

- *Інтеграції з іншими сервісами.* Trello має інтеграції з багатьма іншими сервісами, такими як Google Drive, Dropbox, Slack, GitHub та багато інших.

Мінуси:

- *Обмежена безкоштовна версія.* Trello має обмежену безкоштовну версію, що не дозволяє використовувати всі функції, такі як нагадування, мітки, пріоритети та інші.
- *Вартість підписки.* Повна функціональність Trello доступна лише з підпискою, що може бути вартістю, залежно від того, як часто ви плануєте використовувати додаток та які функції вам необхідні.
- *Обмежені можливості пошуку та фільтрації.* Trello має обмежені можливості пошуку та фільтрації, що може бути незручним для користувачів з великою кількістю завдань та проектів.

Trello є корисним інструментом для керування проектами та задачами, особливо для командної роботи. Однак, для деяких користувачів може бути недостатньою обмежена безкоштовна версія та обмежені можливості пошуку та фільтрації. Також, вартість підписки на повну функціональність Trello може бути високою для деяких користувачів. Тому, як і в інших програмних засобах, важливо ретельно вивчити функціонал та оцінити його відповідність вашим потребам перед тим, як вибрати Trello або інший інструмент для керування задачами.

3. Asana - це інструмент для керування проектами, який дозволяє створювати задачі, призначати їх відповідальним особам, встановлювати терміни та відстежувати прогрес.

Плюси:

- *Зручний та легкий інтерфейс.* Asana має дуже зручний та легкий інтерфейс, що дозволяє легко створювати, редагувати та організовувати завдання.
- *Можливість створювати проекти та завдання.* Asana дозволяє створювати проекти та завдання, які допомагають організовувати роботу візуально та детально.
- *Можливість додавати коментарі та вкладення.* Asana дозволяє додавати коментарі та вкладення до завдань, що дозволяє легко спілкуватися та обговорювати завдання з іншими користувачами.
- *Інтеграції з іншими сервісами.* Asana має інтеграції з багатьма іншими сервісами, такими як Google Drive, Dropbox, Slack, GitHub та багато інших.

Мінуси:

- *Обмежена безкоштовна версія.* Asana має обмежену безкоштовну версію, що не дозволяє використовувати всі функції, такі як нагадування, мітки, пріоритети та інші.
- *Вартість підписки.* Повна функціональність Asana доступна лише з підпискою, що може бути вартістю, залежно від того, як часто ви плануєте використовувати додаток та які функції вам необхідні.
- *Обмежені можливості пошуку та фільтрації.* Asana має обмежені можливості пошуку та фільтрації, що може бути незручним для користувачів з великою кількістю завдань та проектів.

Загалом, Asana є дуже корисним інструментом для керування проектами та задачами, зручним та простим у використанні. Однак, як і будь-який інший програмний засіб, він має свої обмеження та недоліки, які необхідно враховувати при його використанні. Якщо вам потрібен інструмент для

керування проектами та задачами, рекомендуємо ретельно дослідити можливості Asana та інших аналогів, щоб знайти той, який найбільш відповідає вашим потребам та бюджету.

4. Google Calendar - це календарний додаток, який дозволяє планувати зустрічі та події, встановлювати нагадування та додавати інші корисні деталі.

Плюси:

- *Легкий у використанні.* Google Calendar має дуже простий та легкий інтерфейс, що дозволяє легко створювати та організовувати події та зустрічі.
- *Можливість додавати гостей та додаткові календарі.* Google Calendar дозволяє додавати гостей та додаткові календарі, що дозволяє легко планувати зустрічі та події з колегами, друзями та родичами.
- *Можливість налаштовувати нагадування.* Google Calendar дозволяє налаштовувати нагадування про події та зустрічі, що дозволяє не пропустити важливі дати та часи.
- *Інтеграції з іншими сервісами.* Google Calendar має інтеграції з іншими сервісами, такими як Gmail, Google Meet, Microsoft Outlook та багато інших, що дозволяє зручно та швидко планувати зустрічі та події.

Мінуси:

- *Не дуже ефективний для керування завданнями.* Google Calendar не є найкращим інструментом для керування завданнями, оскільки його функціонал обмежений у цьому плані.

- *Обмежена можливість налаштування сповіщень.* Google Calendar має обмежені можливості налаштування сповіщень про події та зустрічі, що може бути незручним для деяких користувачів.

- *Залежність від Інтернету.* Google Calendar потребує підключення до Інтернету, щоб працювати, що може бути проблемою у випадку відсутності зв'язку.

Загалом, Google Calendar є дуже зручним та простим у використанні інструмент для планування подій та зустрічей, який має багато корисних функцій та інтеграцій з іншими сервісами. Проте, якщо вам потрібен програмний засіб для керування завданнями, можливо, варто розглянути альтернативи, які мають більш розвинуті функції для цієї мети. Також, якщо вам потрібна можливість працювати офлайн, Google Calendar може не підійти, тому варто звернути увагу на цей аспект перед тим, як вибирати програмний засіб для планування та керування своїм часом та завданнями.

5. RescueTime - це програма, яка дозволяє відстежувати, яким чином ви витрачаєте час на комп'ютері, та надавати рекомендації щодо оптимізації робочого часу.

Плюси:

- *Моніторинг діяльності.* RescueTime дозволяє автоматично відстежувати, як ви витрачаєте свій час на комп'ютері або смартфоні.

- *Аналіз продуктивності.* RescueTime надає детальну статистику про ваше використання часу, що може допомогти вам зрозуміти, на що ви витрачаєте більше часу і як змінити свої звички, щоб бути більш продуктивним.

- *Блокування відволікаючих сайтів.* RescueTime може блокувати доступ до відволікаючих сайтів, таких як соціальні мережі або

інтернет-магазини, щоб допомогти вам сконцентруватися на важливих завданнях.

Мінуси:

- *Обмеження безкоштовної версії.* RescueTime безкоштовна версія обмежена в можливостях, вона надає більш повну функціональність в платній версії.
- *Недостатній контроль над відслідковуванням.* RescueTime відстежує вашу активність автоматично, і це може бути небажано для деяких користувачів, які хочуть більше контролювати свою приватність.
- *Обмеження на мобільних пристроях.* RescueTime має обмеження на мобільних пристроях, таких як обмеження на аналіз додатків на iOS-пристроях, або обмеження на відстеження використання на Android-пристроях, які працюють в фоновому режимі.

У загальному, RescueTime - це корисний інструмент для відслідковування та аналізу використання часу, який може бути особливо корисним для людей, які працюють з комп'ютером або мобільним пристроєм. Однак, як і з будь-яким програмним засобом, важливо бути обережним і ретельно вибирати параметри відслідковування, щоб він не заважав вашій роботі та приватному життю.

2.2. Опис мінімально життєздатного продукту

Мінімально життєздатний продукт (MVP) для програмного засобу керування часом та управління задачами повинен включати основні функції, які допоможуть користувачам ефективніше управляти своїм часом.

Основні функції засобу керування часом мають включати в себе створення та планування задач з невеликим описом, можливість позначати завдання як виконані, а також у програмі повинен бути мінімальний інтерфейс.[40]

Для створення мінімально життєздатного продукту необхідно створити прототип з вказаними функціями, який може бути протестований користувачами. Зручність використання програми можна забезпечити шляхом додавання додаткових функцій, таких як оновлення опису завдань, нагадування про закінчення терміну виконання завдань, можливість надавати завданням пріоритети, а також можливі інтеграції з календарем та іншими програмами. Однак, для мінімально життєздатної версії, необхідно обмежитися основними функціями та забезпечити їх безперебійну роботу.[34]

У підсумку, мінімально життєздатний продукт для програми керування часом та управління задачами повинен включати базові функції, такі як:

- Можливість створення завдань з описом;
- Можливість перегляду списку завдань;
- Можливість редагувати існуючі завдання;
- Можливість встановити таймер для завдання;
- Можливість скасувати таймер;
- Можливість видалення завдань.

Це базові функції, які повинні працювати у MVP, але в подальшому проект може бути і повинен додавати більший функціонал для розвитку своїх можливостей.

ROADMAP:

Для розробки програмного засобу керування часом та управління задачами з урахуванням підходу MVP виглядає наступним чином:

Створення прототипу з мінімальною функціональністю (MVP)

- Розробити можливість створення та планування завдань з описом
- Розробити можливість перегляду списку завдань з невеликим описом
- Додавання можливості редагування опису завдань
- Розробити простий інтерфейс користувача для створення та перегляду завдань

- Провести тестування прототипу
- Аналіз фідбеку від користувачів та спеціалістів
- Виправлення помилок та недоліків в прототипі

Оновлення функціоналу програмного засобу

- Розробити можливість позначення завдань як виконані
- Додавання можливості встановлення терміну виконання завдань
- Розробити можливість нагадування про терміни виконання завдань
- Додавання можливості встановлення пріоритету завдань
- Провести тестування функціоналу зі спеціалістами та користувачами
- Аналіз фідбеку від користувачів та спеціалістів
- Виправлення помилок та недоліків в прототипі

2.3. Вибір платформи

Управління часом і завданнями відіграють ключову роль у налагодженні ефективного робочого процесу та досягненні високої продуктивності. При створенні програмного забезпечення для цих цілей важливо обрати платформу, яка забезпечить ефективну реалізацію функціональності та задовольнить потреби користувачів. Нижче перелічено декілька можливих варіантів програмного забезпечення для управління часом і завданнями, а також їх переваги та недоліки.[31]

Веб-додатки:

Плюси:

- Доступність з будь-якого пристрою з Інтернет-підключенням.
- Широкий охоплення аудиторії користувачів.
- Простота впровадження оновлень та розширення функціональності.

Мінуси:

- Залежність від Інтернет-з'єднання.

- Обмежені можливості взаємодії з операційною системою користувача.
- Потенційні проблеми з безпекою.

Мобільні додатки:

Плюси:

- Висока мобільність та доступність.
- Інтеграція з функціональністю мобільних пристроїв (наприклад, календар, повідомлення).
- Можливість використання специфічних функцій пристрою, таких як сповіщення.

Мінуси:

- Необхідність розробки окремих версій для різних платформ (iOS, Android).
- Процес схвалення додатків в магазинах додатків може бути тривалим.
- Залежність від оновлень операційних систем мобільних пристроїв.

Телеграм бот:

Плюси:

- Велика база користувачів, які вже використовують Телеграм.
- Інтуїтивний та зручний інтерфейс для взаємодії з користувачем.
- Можливість надсилання повідомлень, сповіщень та нагадувань через платформу миттєвого повідомлення.
- Інтеграція з іншими функціями Телеграм, такими як календар, списки завдань і групові чати.

Мінуси:

- Обмежена можливість використання функціональності, порівняно з веб-додатками або мобільними додатками.
- Залежність від інтернет-з'єднання.
- Необхідність розробки та підтримки бота для Телеграм окремо від інших платформ.

З усіх цих варіантів було обрано розробку додатка в Телеграм боті.[10]

Масовість аудиторії: Телеграм є однією з найпопулярніших платформ миттєвих повідомлень, з мільярдами користувачів по всьому світу. Розробка додатку для Телеграму дозволить охопити широку аудиторію та забезпечити доступність користувачам. Можливо аудиторія телеграму буде менша за мобільні додатки але в телеграмі є вбудована інтеграція зі сповіщеннями, тобто Телеграм надає можливість надсилати сповіщення та повідомлення користувачам через платформу, без використання додаткових інструментів. Це дозволяє створити ефективну систему нагадувань про завдання та важливі події, що сприяє поліпшенню управління часом та збільшенню продуктивності.

Також інтерфейс Телеграму зрозумілий та зручний для більшості користувачів. Розробка додатку у вигляді Телеграм бота дозволить забезпечити простоту взаємодії з програмним засобом. Користувачі зможуть з легкістю створювати та керувати завданнями, встановлювати нагадування та отримувати інформацію про свій часовий розклад без необхідності встановлювати додаток окремо.[24]

Розробка та підтримка Телеграм бота дозволяє швидко вносити зміни до функціональності, вдосконалювати та оновлювати додаток. При цьому користувачі автоматично отримують оновлення, не потребуючи додаткових дій з їх боку.

Також Телеграм має високі стандарти безпеки та шифрування, що робить його надійною платформою для зберігання та обробки конфіденційної інформації. Це особливо важливо при управлінні задачами та розкритті конфіденційних даних.

Загалом, розробка програмного засобу для керування часом та управління задачами у Телеграмі (тобто створення Телеграм бота) має численні переваги, включаючи масовість аудиторії, зручний інтерфейс, інтеграцію зі сповіщеннями, гнучкість оновлень та високу безпеку. Враховуючи ці переваги, Телеграм може бути кращим варіантом платформи для реалізації програмного засобу.

3. Розробка програмного засобу керування часом та управління задачами

3.1. Розробка прототипу

Для початку написання коду треба підготуватись і створити токен для подальшої взаємодії з серверами Telegram.[35] Токен - це унікальний ідентифікатор, який дає можливість програмному забезпеченню доступ до API Telegram. Отже, щоб створити токен в Telegram, потрібно дотриматися наступних кроків:

- 1). Відкриваємо месенджер Telegram та знаходимо бота @BotFather.
- 2). Запускаємо чат з @BotFather та вводимо команду /newbot, щоб створити нового бота.
- 3). Вводимо назву свого бота та натискаємо "Enter".
- 4). Далі в @BotFather вводимо коротку мітку для нашого бота. Ця мітка повинна закінчуватися на "bot", наприклад, "mytestbot".
- 5). Після цього @BotFather надішле нам повідомлення з токеном нашого бота.
- 6). Скопіюємо токен та збережемо його у безпечному місці. Токен має приблизно такий вигляд: 1234567890:AbCDeFGhIJKLmnOPQrStUvWxYZ.

Важливо пам'ятати, що токен - це конфіденційна інформація, яку не слід передавати нікому іншому. Також варто дотримуватися найкращих практик забезпечення безпеки, щоб захистити наш токен від несанкціонованого доступу.

Після створення токена, заходимо в середу розробки (в даній роботі використовується середовище розробки IntelliJ IDEA), при запуску проєкту створюємо основний файл - «main.py», та два додаткових – «tasks.json» та «timers.json». [32]

В файл tasks.json запишемо тільки квадратні дужки - «[]», цей файл буде використовуватись як база даних для завдань. В файл timers.json запишемо

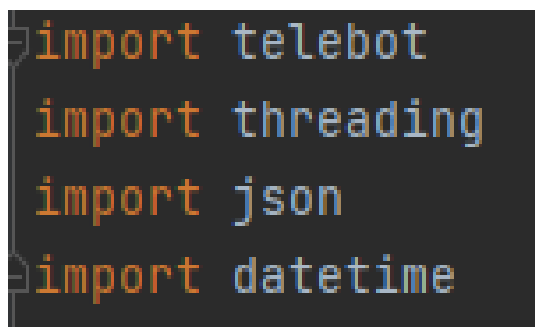
тільки фігурні душки - «{}», цей файл буде використовуватись як база даних для створених таймерів. В основний файл main.py імпортемо всі необхідні бібліотеки(Рис.1.), а саме:

«import telebot» - імпортує бібліотеку Telebot, яка дозволяє розробникам створювати ботів для месенджера Telegram.

«import threading» - імпортує вбудований модуль threading для роботи з потоками виконання.

«import json» - імпортує модуль json, який дозволяє працювати з форматом даних JSON.

«import datetime» - імпортує вбудований модуль datetime для роботи з датою та часом.



```
import telebot
import threading
import json
import datetime
```

Рис.1. Імпортовані бібліотеки

Бібліотека – це набір написаних кодом функцій, процедур, класів та інших складових, які можуть бути використані розробниками для розв'язання конкретних задач. Бібліотеки зазвичай містять функціональність, яка була написана заздалегідь та може бути використана у багатьох програмах, що дозволяє економити час розробки та зменшувати кількість помилок. Бібліотеки можуть бути стандартними, які входять в склад розроблюваної мови програмування та зазвичай не потребують окремого встановлення, або зовнішніми, які потрібно встановлювати окремо для використання. Також існують спеціалізовані бібліотеки, які мають написані певні функції для роботи з конкретними технологіями, протоколами чи програмними продуктами.

Використання бібліотек може значно спростити та прискорити процес розробки програмного продукту. В більшості випадків розробники використовують бібліотеки, щоб скоротити час розробки та підвищити якість коду.[7]

Далі потрібно вписати константи. Константи TOKEN, TASKS_FILE, і TIMERS_FILE використовуються в коді для зберігання API токена, шляху до файлу з завданнями та шляху до файлу з таймерами відповідно. Значення константи TOKEN є токеном Telegram API, необхідним для здійснення з'єднання з Telegram API та запуску бота. Константи TASKS_FILE та TIMERS_FILE містять шлях до файлів, в які бот зберігає завдання користувачів та таймери відповідно. Ці файли використовуються для зберігання даних між запусками бота (Рис.2.).



```
# Константи
TOKEN = '...'
TASKS_FILE = 'tasks.json'
TIMERS_FILE = 'timers.json'
```

Рис.2. Константи

Константи - це значення, які не можуть змінюватись під час виконання програми. Константи використовуються для зберігання значень, які є важливими для програми, і вони можуть бути доступні для використання в різних частинах програми. Константи використовуються для забезпечення безпеки та стабільності програми, оскільки вони не можуть бути змінені під час виконання програми. Вони також допомагають уникнути помилок, які можуть виникнути внаслідок неправильного використання змінних.

Далі ми ініціалізуємо дані (Рис.3.). Ініціалізацією називають створення з'єднання між нашим програмним кодом та Telegram API для забезпечення можливості взаємодії з ботом в Telegram. Це означає налаштування з'єднання з Telegram серверами, отримання доступу до бот - інтерфейсу та призначення

токену. Після успішної ініціалізації, бот може взаємодіяти з користувачами Telegram та виконувати функції, які ми йому надамо.

```
# Ініціалізуємо бота
bot = telebot.TeleBot(TOKEN)

# Ініціалізуємо дані
tasks = []
timers = {}
```

Рис.3. Ініціалізація даних

Після підготовки всіх бібліотек, даних і файлів можемо починати розробляти функціонал. Почнемо з команди start (Рис.4.). Коли користувач напише команду боту (/start), виконується функція start(). Функція отримує параметр message, який містить інформацію про повідомлення, яке було надіслане боту, далі програма отримує ідентифікатор чату з параметру message, і відправляє користувачу повідомлення привітання з інструкцією.

```
@bot.message_handler(commands=['start'])
def start(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Ласкаво просимо до бота-тайм-менеджера! '
                              '\nВикористайте /help для перегляду списку доступних команд.')
```

Рис.4. Команда start

Після привітання, в програмі реалізована команда «/help» (Рис.5.). Ця команда виконується, коли користувач вперше починає взаємодіяти з ботом і потрібна для того, щоб користувач розумів який функціонал доступний в програмі. Також дане меню може допомогти швидко використати ту функцію, яка потрібна користувачу, для цього потрібно натиснути на відповідну команду в меню.

```
@bot.message_handler(commands=['help'])
def help(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, '/new_task      - Створити нове завдання.\n'
                               '/list_tasks   - Список всіх завдань.\n'
                               '/edit_task    - Редагувати існуюче завдання.\n'
                               '/set_timer    - Встановити таймер для завдання.\n'
                               '/cancel_timer - Скасувати таймер.\n'
                               '/delete_task  - Видалити існуюче завдання.\n')
```

Рис.5. Команда help

Далі починаємо розробляти основні функції програмного засобу, а саме «/new_task» (Рис.6.). Ця частина коду визначає функцію new_task, яка викликається при отриманні команди /new_task. При виклику цієї команди, бот починає процес створення нового завдання.

- 1). Функція new_task відправляє повідомлення користувачу з проханням ввести назву завдання;
- 2). Далі після отримання відповіді з назвою завдання, викликається функція new_task_name;
- 3). Функція new_task_name отримує назву завдання і відправляє повідомлення користувачу з проханням ввести опис завдання;
- 4). Після отримання відповіді з описом завдання, викликається функція new_task_description;
- 5). Функція new_task_description отримує опис завдання, генерує унікальний ідентифікатор для завдання, отримує ідентифікатор користувача, створює словник, що представляє завдання, і додає його до списку tasks;
- 6). Після збереження завдання, бот відправляє повідомлення користувачу про успішне створення завдання, а також повідомлення з деталями нового завдання.

Також в цій частині коду використовуються інші допоміжні функції, такі як save_tasks і get_task_message, які виконують збереження завдань у файлі та форматування повідомлення про завдання відповідно.

```

@bot.message_handler(commands=['new_task'])
def new_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть назву завдання:')
    bot.register_next_step_handler(message, new_task_name)

def new_task_name(message):
    chat_id = message.chat.id
    task_name = message.text
    bot.send_message(chat_id, 'Введіть опис завдання:')
    bot.register_next_step_handler(message, new_task_description, task_name)

def new_task_description(message, task_name):
    chat_id = message.chat.id
    task_description = message.text
    task_id = len(tasks) + 1
    user_id = message.from_user.id
    task = {'user_id': user_id, 'task_id': task_id, 'task_name': task_name, 'task_description': task_description, 'status': 'Виконується'}
    tasks.append(task)
    save_tasks()
    bot.send_message(chat_id, 'Завдання успішно створено!')
    bot.send_message(chat_id, get_task_message(task))

```

Рис.6. Команда new_task

Після створення завдання, ми можемо використати команду «/list_tasks» (Рис.7.). Ця команда відображає список усіх завдань, які створив користувач, який відправив запит. Спочатку визначається ID користувача, який надіслав повідомлення, а далі програма фільтрує усі завдання за цим ID. Якщо користувач не має жодного завдання, бот повідомляє про це. В іншому випадку бот відправляє повідомлення з описом кожного завдання. Для створення тексту повідомлення використовується функція get_task_message, яка бере завдання як вхідний параметр і повертає рядок з його даними.

```

@bot.message_handler(commands=['list_tasks'])
def list_tasks(message):
    chat_id = message.chat.id
    user_id = message.from_user.id
    user_tasks = [task for task in tasks if task['user_id'] == user_id]
    if len(user_tasks) == 0:
        bot.send_message(chat_id, 'Завдань не знайдено.')
    else:
        for task in user_tasks:
            bot.send_message(chat_id, get_task_message(task))

```

Рис.7. Команда list_tasks

Після перегляду всіх існуючих завдань, користувач може відредагувати свої завдання, для цього треба використати команду «/edit_task» (Рис.8.). Ця функція обробляє команду /edit_task і дозволяє користувачу редагувати існуюче завдання.

При отриманні команди, спочатку користувачу надсилається повідомлення з проханням ввести ID завдання, яке він хоче відредагувати. Після отримання ID завдання, викликається функція edit_task_id.

У функції edit_task_id перевіряється, чи існує завдання з введеним ID. Якщо такого завдання немає, бот надсилає повідомлення про відсутність завдання. У протилежному випадку, береться індекс завдання за допомогою функції get_task_index і зберігається в змінну task_index. Потім береться саме завдання за допомогою tasks[task_index] і зберігається в змінну task.

Далі бот надсилає повідомлення з проханням ввести нову назву завдання, після чого викликається функція edit_task_name зі змінною task як аргументом.

У функції edit_task_name бот отримує нову назву завдання з повідомлення користувача і змінює поле task['task_name'] на нове значення. Після цього викликається функція save_tasks для збереження змін до файлу. Потім бот надсилає повідомлення з проханням ввести новий опис завдання і реєструє функцію edit_task_description для обробки наступного повідомлення.

У функції edit_task_description бот отримує новий опис завдання з повідомлення користувача і змінює поле task['task_description'] на нове значення. Після цього знову викликається функція save_tasks для збереження змін до файлу. Нарешті, бот надсилає повідомлення з оновленими даними завдання за допомогою функції get_task_message.

```

@bot.message_handler(commands=['edit_task'])
def edit_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, edit_task_id)

def edit_task_id(message):
    chat_id = message.chat.id
    task_id = int(message.text)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        task = tasks[task_index]
        bot.send_message(chat_id, 'Введіть нову назву завдання:')
        bot.register_next_step_handler(message, edit_task_name, task)

def edit_task_name(message, task):
    chat_id = message.chat.id
    task_name = message.text
    task['task_name'] = task_name
    save_tasks()
    bot.send_message(chat_id, 'Введіть новий опис завдання:')
    bot.register_next_step_handler(message, edit_task_description, task)

def edit_task_description(message, task):
    chat_id = message.chat.id
    task_description = message.text
    task['task_description'] = task_description
    save_tasks()
    bot.send_message(chat_id, get_task_message(task))

```

Рис.8. Команда edit_task

Далі реалізуємо можливість видалення завдань – «/delete_task» (Рис.9.). При отриманні команди, користувачу надсилається повідомлення з проханням ввести ID завдання, яке він хоче видалити. Після отримання ID завдання, викликається функція delete_task_id.

У функції delete_task_id перевіряється, чи існує завдання з введеним ID. Якщо такого завдання немає, бот надсилає повідомлення про відсутність завдання. У протилежному випадку, береться індекс завдання за допомогою

функції `get_task_index` і зберігається в змінну `task_index`. Потім виконується видалення завдання зі списку завдань за допомогою методу `pop(task_index)` і зберігається зміна до файлу за допомогою функції `save_tasks`. Нарешті, бот надсилає повідомлення про успішне видалення завдання.

```
@bot.message_handler(commands=['delete_task'])
def delete_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, delete_task_id)

def delete_task_id(message):
    chat_id = message.chat.id
    task_id = int(message.text)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        tasks.pop(task_index)
        save_tasks()
        bot.send_message(chat_id, 'Завдання успішно видалено!')
```

Рис.9. Команда `delete_task`

Для нагадування використовується команда «`/set_timer`» (Рис.10.). Ця частина коду обробляє команду і дозволяє користувачу встановити таймер для конкретного завдання.

При отриманні команди, користувачу надсилається повідомлення з проханням ввести ID завдання, для якого він хоче встановити таймер. Після отримання ID завдання, викликається функція `set_timer_id`.

У функції `set_timer_id` перевіряє чи існує завдання з введеним ID. Якщо такого завдання немає, бот надсилає повідомлення про відсутність завдання. У протилежному випадку, береться індекс завдання за допомогою функції

`get_task_index` і зберігається в змінну `task_index`. Потім береться саме завдання зі списку завдань за допомогою `tasks[task_index]` і зберігається в змінну `task`.

Далі бот надсилає повідомлення з проханням ввести час у хвиликах для таймера. Коли отримується відповідь з часом, викликається функція `set_timer_time`.

У функції `set_timer_time` спочатку отриманий час перетворюється у кінцевий час шляхом додавання його до поточного часу (`datetime.datetime.now()`) за допомогою `timedelta`. Отриманий кінцевий час зберігається в змінну `end_time`.

Потім встановлюються дані таймера в словнику `timers`, де ключем є ID завдання, а значенням є словник з `end_time` (кінцевий час таймера), `chat_id` (ідентифікатор чату) та `message_id` (ідентифікатор повідомлення, яке бот буде оновлювати). Збереження таймерів до файлу виконується за допомогою функції `save_timers`. Далі, бот надсилає повідомлення про успішне встановлення таймера. Після встановлення таймера запускається функція `timer_callback`, яка виконується через певний час, встановлений користувачем. Функція отримує `chat_id` та `timer_id`, використовуючи які вона знаходить індекс завдання за допомогою функції `get_task_index`. Якщо індекс не дорівнює `None`, то береться саме завдання зі списку завдань за допомогою `tasks[task_index]` і надсилається повідомлення про те, що таймер завершився. Після цього видаляється дані таймера зі словника `timers` та зберігається його оновлена версія до файлу за допомогою функції `save_timers`. [1]

```

=====Команда /set_timer
@bot.message_handler(commands=['set_timer'])
def set_timer(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, set_timer_id)

def set_timer_id(message):
    chat_id = message.chat.id
    task_id = message.text
    if not task_id.isdigit():
        bot.send_message(chat_id, 'Ви вказали невірний формат.')
        return
    task_id = int(task_id)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        task = tasks[task_index]
        if task_id in timers:
            bot.send_message(chat_id, 'На це завдання вже встановлений таймер.')
        else:
            bot.send_message(chat_id, 'Введіть час у хвиликах:')
            bot.register_next_step_handler(message, set_timer_time, task)

def set_timer_time(message, task):
    chat_id = message.chat.id
    time = int(message.text)
    end_time = datetime.datetime.now() + datetime.timedelta(minutes=time)
    timers[task['task_id']] = {'end_time': end_time.timestamp(), 'chat_id': chat_id, 'message_id': None}
    save_timers()
    bot.send_message(chat_id, 'Таймер встановлено успішно!')

    timer_id = task['task_id']
    timer_interval = time * 60
    timer = threading.Timer(timer_interval, timer_callback, args=[chat_id, timer_id])
    timer.start()

def timer_callback(chat_id, timer_id):
    task_index = get_task_index(timer_id)
    if task_index is not None:
        task = tasks[task_index]
        bot.send_message(chat_id, f'Таймер для завдання "{task["task_name"]}" закінчився!')
        del timers[timer_id]
        save_timers()

```

Рис.10. Команда set_timer

Якщо користувач виконав завдання і хоче видалити таймер на завданні, яке виконував то тут допоможе команда «/cancel_timer» (Рис.11.).

При отриманні команди /cancel_timer, користувачу надсилається повідомлення з проханням ввести ID завдання, для якого він хоче скасувати таймер. Після отримання ID завдання, викликається функція cancel_timer_id.

У функції cancel_timer_id спочатку перевіряється, чи існує таймер для завдання з введеним ID. Якщо такого таймера немає, бот надсилає повідомлення про відсутність таймера. У протилежному випадку, береться саме таймер зі словника timers за допомогою task_id як ключа. Якщо значення

`message_id` в таймері не є `None`, бот видаляє повідомлення з цим ідентифікатором за допомогою функції `bot.delete_message`. Потім таймер видаляється зі словника `timers` за допомогою `task_id` як ключа. Збереження таймерів до файлу виконується за допомогою функції `save_timers`. Нарешті, бот надсилає повідомлення про успішне скасування таймера.

```
@bot.message_handler(commands=['cancel_timer'])
def cancel_timer(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, cancel_timer_id)

def cancel_timer_id(message):
    chat_id = message.chat.id
    task_id = int(message.text)
    if task_id not in timers:
        bot.send_message(chat_id, 'Таймер не знайдено.')
    else:
        timer = timers[task_id]
        if timer['message_id'] is not None:
            bot.delete_message(chat_id, timer['message_id'])
        del timers[task_id]
        save_timers()
    bot.send_message(chat_id, 'Таймер успішно скасовано!')
```

Рис.11. Команда `cancel_timer`

У кінці код містить допоміжні функції та змінні, які виконують певні операції для збереження та завантаження задач і таймерів в програмі (Рис.12.).

`get_task_index(task_id)` - ця функція приймає ідентифікатор задачі (`task_id`) та повертає індекс цієї задачі в списку задач (`tasks`), якщо вона є в списку. Якщо задача не знайдена, повертається `None`.

`get_task_message(task)` - ця функція приймає словник з даними задачі (`task`) та повертає рядок з інформацією про задачу, включаючи її ідентифікатор, тему, опис та статус.

`save_tasks()` - ця функція зберігає список задач в файлі з назвою `TASKS_FILE`. Перед збереженням, для кожної задачі в списку додається поле `"user_id"`, що містить ідентифікатор користувача, який створив цю задачу.

`load_tasks()` - ця функція завантажує список задач з файлу з назвою `tasks.json` та повертає його. Якщо в файлі є поле `"user_id"` для кожної задачі, це поле перетворюється назад на ціле число.

`save_timers()` - ця функція зберігає словник таймерів (`timers`) в файлі з назвою `TIMERS_FILE`.

`load_timers()` - ця функція завантажує словник таймерів з файлу з назвою `TIMERS_FILE` та зберігає його в змінну `timers`. Якщо файл не знайдено, створюється порожній словник.

`bot.polling(none_stop=True)` - це запуск програми, завдяки цьому коду бот буде працювати допоки адміністратор не вимкне його.

```
def get_task_index(task_id):
    for i in range(len(tasks)):
        if tasks[i]['task_id'] == task_id:
            return i
    return None

def get_task_message(task):
    task_message = f"ID: {task['task_id']}\nТема: {task['task_name']}\nОпис: {task['task_description']}\nСтатус: {task['status']}"
    return task_message

def save_tasks():
    with open(TASKS_FILE, 'w') as f:
        tasks_to_save = [dict(task, user_id=str(task['user_id'])) for task in tasks]
        json.dump(tasks_to_save, f, indent=4)

def load_tasks():
    with open('tasks.json', 'r') as f:
        tasks = json.load(f)
        for task in tasks:
            if 'user_id' in task:
                task['user_id'] = int(task['user_id'])
        return tasks

def save_timers():
    with open(TIMERS_FILE, 'w') as f:
        json.dump(timers, f)

def load_timers():
    global timers
    try:
        with open(TIMERS_FILE, 'r') as f:
            timers = json.load(f)
    except FileNotFoundError:
        timers = {}

load_tasks()
load_timers()

bot.polling(none_stop=True)
```

Рис.12. Функції та змінні

3.2. Тестування продукту

На даному етапі розробки буде проводитись тестування продукту, це є важливим етапом в процесі розробки, що допомагає впевнитися в якості та надійності розробки перед її випуском на ринок. У цьому підрозділі ми зосередимося на проведенні тестування продукту та виявленні потенційних проблем і недоліків. [37]

Для того щоб почати тестування продукту, потрібно створити тестові сценарії, у яких будуть описані детальні сценарії, які описують послідовність кроків для перевірки окремих функцій чи взаємодій системи. В системі є декілька команд, з якими працюють користувачі і для всіх команд є очікуваний результат:

- Команда «/start»;

Очікуваний результат: Команда активується самостійно коли користувач тільки починає роботу з ботом. Бот відправляє повідомлення "Ласкаво просимо до бота-тайм-менеджера! Використайте /help для перегляду списку доступних команд."

- Команда «/help»;

Очікуваний результат: Бот відправляє повідомлення

«/new_task - Створити нове завдання.

/list_tasks - Список всіх завдань.

/edit_task - Редагувати існуюче завдання.

/set_timer - Встановити таймер для завдання.

/cancel_timer - Скасувати таймер.

/delete_task - Видалити існуюче завдання.»

- Команда «/new_task»;

Очікуваний результат: Бот запитує користувача про назву завдання, після отримання назви від користувача, бот запитує користувача про опис завдання,

після отримання опису від користувача, бот відправляє повідомлення про успішне створення завдання і виводить інформацію про створене завдання.

- Команда «/list_tasks»;

Очікуваний результат: Бот відправляє повідомлення зі списком всіх завдань, який зробив саме користувач.

- Команда «/edit_task»;

Очікуваний результат: Бот запитує користувача про ID завдання, яке потрібно редагувати, після отримання ID від користувача, бот починає пошук, якщо завдання з таким ID відсутнє, бот пише що такого завдання немає, якщо завдання існує, бот запитує користувача про нову назву завдання. Далі після зміни назви, бот запитує користувача про новий опис завдання, коли користувач закінчує опис, бот зберігає нові дані про завдання і надсилає повідомлення з оновленою інформацією про завдання.

- Команда «/set_timer».

Очікуваний результат: Бот запитує користувача про ID завдання, для якого потрібно встановити таймер, після отримання ID від користувача, бот починає пошук, якщо завдання з таким ID відсутнє, бот пише що такого завдання немає, якщо завдання існує, бот запитує час у хвилинах, після отримання часу, бот встановлює таймер на завданню число хвилин для завдання з вказаним ID і відправляє повідомлення про успішне встановлення таймера.

- Команда «/cancel_timer».

Бот запитує користувача про ID завдання, для якого потрібно скасувати таймер, після отримання ID від користувача, бот починає пошук, якщо завдання з таким ID відсутнє, бот пише що такого завдання немає, якщо завдання існує, бот скасовує таймер для відповідного завдання і відправляє повідомлення про успішне скасування таймера.

- Команда «/delete_task»;

Очікуваний результат: Бот запитує користувача про ID завдання, яке потрібно видалити, після отримання ID від користувача, бот починає пошук, якщо завдання з таким ID відсутнє, бот пише що такого завдання немає, якщо

завдання існує, бот видаляє завдання з вказаним ID і відправляє повідомлення про успішне видалення завдання.

Після виявлення очікуваних результатів від команд, можна починати проводити тестування програми. Провівши тестування всіх команд, було знайдено декілька багів, які не були передбачені у системі.

Якщо зробити два завдання, перше видалити і зробити ще одне, два останніх завдання будуть мати один і той самий ID (Рис.13.). Це погано тим, що при ID для всіх завдань встановлюється унікальне і якщо в системі буде два однакових ID, то система буде працювати некоректно.

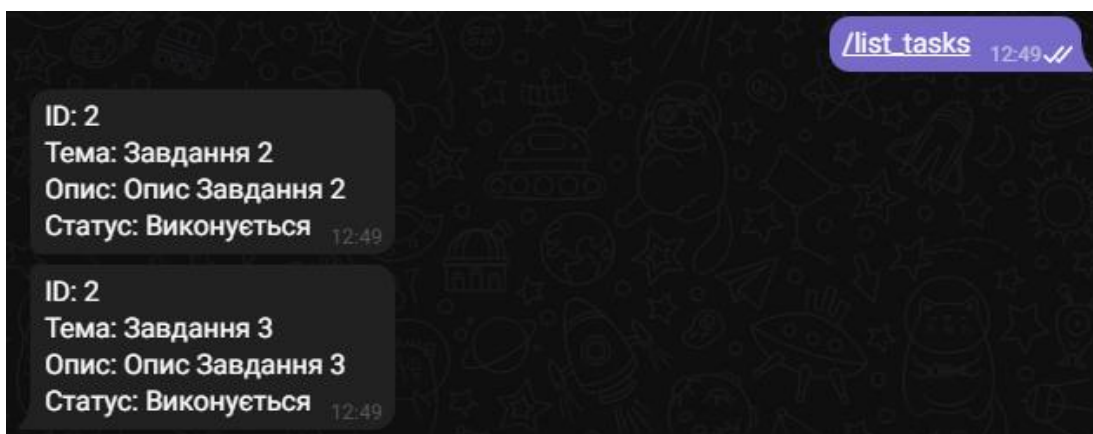


Рис.13. Однакове ID в різних завданнях

Для вирішення цієї проблеми було оновлено частину коду, тепер якщо зробити ті самі маніпуляції, то завдання отримає нову ID (Рис.14.). Якщо всі завдання видалити і створити нове, то ID почнеться з одиниці.

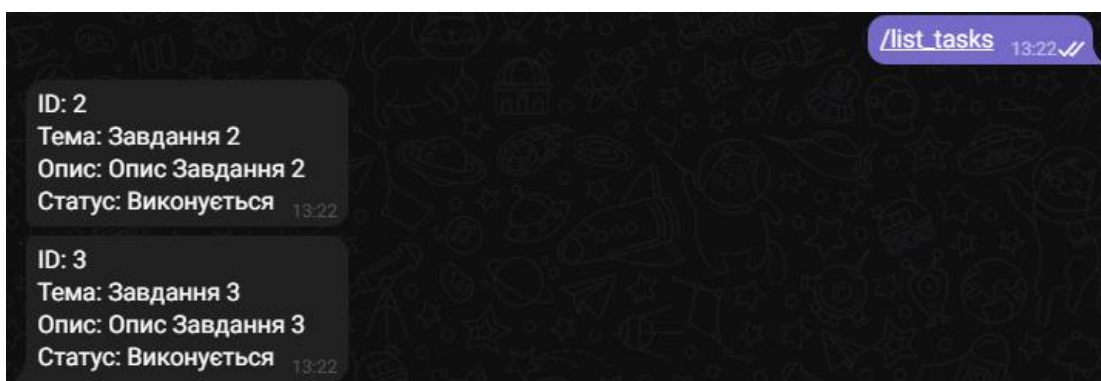


Рис.14. Виправлення помилки з унікальністю ID

Також в програмі було знайдена проблема з вказанням невірного формату. При вказанні стрінгового значення в пункт ID, програма просто вимикалась і потрібно було вмикати її вручну (Рис.15.).

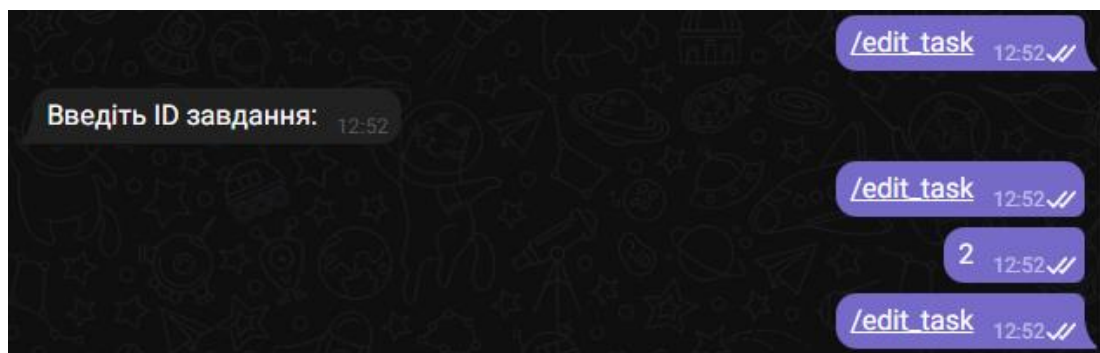


Рис.15. Проблема з вводом неправильного формату

Для вирішення цієї проблеми, в коді було додано декілька строк, які перевіряли правильність формату, тепер всі команди, які просять ввести ID завдання, спочатку перевіряють правильність формату, а вже потім починається основна функція (Рис.16.).

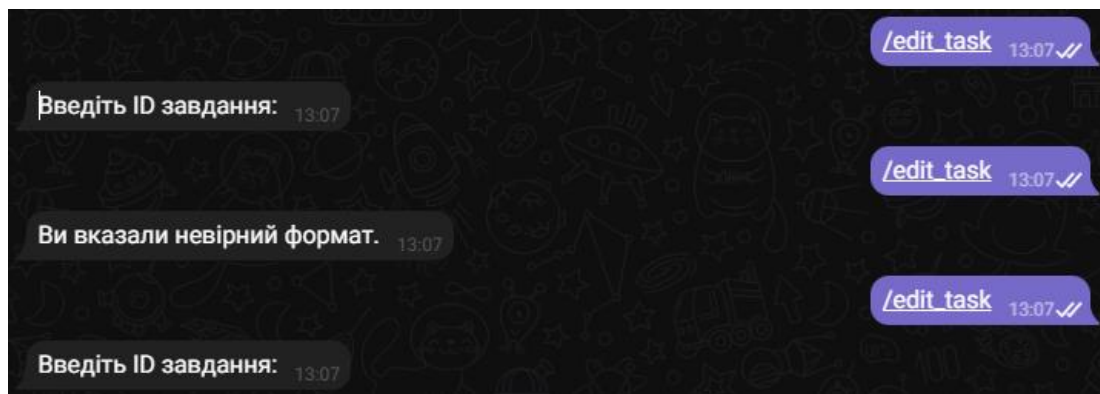


Рис.16. Виправлення помилки з неправильним форматом

Під час проведення тестування були виявлені деякі дефекти програми, але вони успішно виправлені. Тепер програма працює більш надійно і безперебійно, забезпечуючи коректну роботу з завданнями та таймерами.

3.3. Подальший розвиток проєкту

У програмному засобі йде зосередження на наданні користувачам зручних та потужних інструментів для керування їх завданнями. Були враховані потреби та вимоги користувачів, і планується розширення функціоналу програмного засобу, додаючи нові можливості, які допоможуть краще планувати, виконувати та відстежувати свої завдання.

Можливість позначення завдань як виконані:

Ця нова функція в програмному засобі дозволить користувачеві легко відслідковувати завершені завдання і зберігати порядок у списку. При створенні нового завдання користувач зможе позначити його як невиконане за замовчуванням. Після завершення завдання користувач може відмітити його як виконане. Це може бути реалізовано шляхом додавання прапорця або кнопки "Виконано" до кожного завдання у програмі. Після позначення завдання як виконаного, його стан зміниться, щоб вказувати, що завдання виконане. Таким чином, користувач може швидко відокремити завершені завдання від невиконаних у списку. Він також може використовувати фільтри для перегляду лише виконаних або лише невиконаних завдань, залежно від своїх потреб.

Можливості встановлення терміну виконання завдань:

Функція надасть користувачу можливість встановити термін, до якого потрібно завершити кожне завдання. При створенні нового завдання користувач зможе вказати дату або час, коли завдання має бути завершено. Це може бути реалізовано шляхом додавання поля "Термін виконання" до кожного завдання у програмі. Користувач може вибрати дату з календаря або ввести її вручну. Коли термін виконання наближається або минає, програма може надсилати нагадування користувачеві, щоб він не забув про завдання, які повинні бути завершені. Таке нагадування може відбуватися у вигляді сповіщень.

Можливість нагадування про терміни виконання завдань:

Ця функція покращить ефективність та організацію користувача, дозволяючи йому встановлювати нагадування про терміни виконання завдань.

При встановленні терміну виконання для завдання користувач також може встановити час нагадування, коли система повідомить його про наближення терміну виконання. Наприклад, користувач може вибрати, щоб отримувати сповіщення за годину або день до терміну виконання завдання. Це дозволить користувачеві планувати свій час та уникати прострочення завдань. При настанні встановленого часу нагадування програма може надіслати повідомлення або сповіщення користувачеві з інформацією про невиконані завдання та їх терміни виконання.

Можливість встановлення пріоритету завдань:

Ця можливість дозволить користувачеві встановити пріоритет для кожного завдання. При створенні нового завдання користувач може вибрати рівень пріоритету, наприклад, високий, середній або низький, або використовувати іншу систему позначень, яка підходить йому. Це може бути реалізовано шляхом додавання поля "Пріоритет" до кожного завдання. Таким чином, користувач зможе легко визначити, які завдання мають високий пріоритет і потребують негайної уваги, а які можуть бути відкладені на потім. Пріоритетні завдання можуть бути виділені в списку або відображатися спеціальним чином, щоб користувач міг швидко знайти їх та працювати над ними.

Кожна з цих функцій відкриває нові можливості для користувачів програмного засобу, поліпшує організацію їх завдань та сприяє більш ефективному виконанню роботи. Завдяки позначенню завдань як виконаних, користувач може легко відслідковувати свій прогрес та зберігати порядок у списку завдань. Встановлення терміну виконання дозволяє планувати завдання заздалегідь та виконувати їх вчасно. Нагадування про терміни виконання допомагають уникнути забуття про завдання та своєчасно нагадують про них. Встановлення пріоритету завдань дозволяє визначити найважливіші завдання та працювати над ними в першу чергу. Загалом, ці нові можливості сприятимуть покращенню організації та продуктивності користувачів програмного засобу,

дозволяючи їм ефективно планувати, виконувати та контролювати свої завдання.

Висновок

Дипломна робота присвячена розробці програмного засобу керування часом та управління задачами. В ході дослідження теми було проведено аналіз постановки проблеми, визначено цільову аудиторію продукту та виявлено необхідність його розробки.

Перший розділ дослідження зосередився на постановці проблеми. Було виявлено, що багато людей стикаються з проблемою неефективного керування часом та управлінням задачами. Це може призводити до стресу, втрати продуктивності та неспроможності досягнути поставлених цілей. Тому розробка програмного засобу, який допоможе ефективно керувати часом та задачами, є актуальною та необхідною.

Другий розділ дослідження присвячений етапам розробки програмного засобу. Проведено аналіз подібних розробок, що дозволило виявити основні характеристики та функціонал, які повинні бути в продукті. Далі було описано мінімально життєздатний продукт, який включає базовий набір функцій для керування часом та управління задачами. Також був зроблений вибір платформи для реалізації програмного засобу.

Третій розділ дослідження описує процес розробки програмного засобу. Був розроблений прототип продукту, який включає основні функції для керування часом та управління задачами. Після цього було проведено тестування продукту, щоб переконатися у його коректності та надійності. Далі передбачається подальший розвиток проєкту, додавання нового функціоналу та вдосконалення існуючих рішень.

У цілому, результати дослідження показують, що розробка програмного засобу керування часом та управління задачами має великий потенціал для вирішення проблеми неефективного використання часу та недостатньої організації робочих завдань. Розроблений прототип продукту відповідає основним вимогам та може бути використаний як основа для подальшого вдосконалення та розвитку.

Список використаних джерел

- 1.Асинхронне програмування у Python. URL:
https://www.youtube.com/watch?v=w9x2BDlkhMg&ab_channel=CodeUA
(дата звернення: 20.04.2023).
- 2.Бази даних. URL: https://www.youtube.com/@Igor_Bazulkin/playlists
(дата звернення: 02.05.2023).
- 3.Бази даних. <https://www.youtube.com/@DystOsvita/playlists> (дата
звернення: 02.05.2023).
- 4.Вивчення Python з нуля. URL:
<https://www.youtube.com/@uacoding/playlists> (дата звернення: 12.03.2023).
- 5.Відправлення повідомлень через Telegram. URL: [https://baza-
znaniy.appointer.ua/docs/zagalny-nastroiki-akkaunta/vidpravlennya-povidomlen-
cherez-telegram/](https://baza-znaniy.appointer.ua/docs/zagalny-nastroiki-akkaunta/vidpravlennya-povidomlen-cherez-telegram/) (дата звернення: 24.04.2023).
- 6.Власні звуки сповіщень. URL: [https://telegram.org/blog/notifications-
bots/uk](https://telegram.org/blog/notifications-bots/uk) (дата звернення: 25.05.2023).
- 7.Знайомство з Python. <https://www.youtube.com/@user-hi5if2gr6h/playlists>
(дата звернення: 15.03.2023).
- 8.Інтеграція ChatGPT у проєкт. URL:
https://www.youtube.com/watch?v=uFOwaRjh1EE&ab_channel=CodeUA
(дата звернення: 25.05.2023).
- 9.Мова програмування Python. URL:
https://www.youtube.com/@Igor_Bazulkin/playlists
(дата звернення: 11.03.2023).
- 10.Можливості, команди та функціонал. BotFather. URL:
https://gerabot.com/article/botfather_mozhливosti_ta_funkcional
(дата звернення: 01.04.2023).
- 11.ООП Python з нуля. URL: <https://www.youtube.com/@uacoding/playlists>
(дата звернення: 01.05.2023).
- 12.Особливості IntelliJ IDEA. URL:
https://de.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення: 01.03.2023).

13. Основи програмування у Python. URL: <https://www.youtube.com/@DystOsvita/playlists>
(дата звернення: 10.03.2023).
14. Перший телеграм бот. URL: <https://dou.ua/forums/topic/42413/>
(дата звернення: 22.04.2023).
15. Приклади чат-ботів. URL: <https://sendpulse.ua/features/chatbot/chatbot-examples> (дата звернення: 30.04.2023).
16. Підключення Telegram чат-бота. URL: <https://sendpulse.com/ua/knowledge-base/chatbot/telegram/create-telegram-chatbot> (дата звернення: 30.04.2023).
17. Програмування на Python. URL: <https://www.youtube.com/@ProgerSchool/playlists> (дата звернення: 10.03.2023).
18. Створення бота з прийому лідів в Telegram. URL: https://www.youtube.com/watch?v=wSecRwu5fAA&ab_channel=HillelITSchool (дата звернення: 22.04.2023).
19. СТВОРЕННЯ ЧАТ-БОТУ ДЛЯ ФАКУЛЬТЕТУ ІНФОРМАТИКИ. URL: <https://ekmair.ukma.edu.ua/server/api/core/bitstreams/15cb5163-93de-406f-9cb5-c6c50cf431aa/content> (дата звернення: 20.04.2023).
20. Створюємо Телеграм Бота. Найкращі поради. URL: <https://www.telegramadviser.com/uk/how-to-create-telegram-bot/>
(дата звернення: 20.04.2023).
21. Телеграм API. URL: <https://apix-drive.com/ua/telegram-api> (дата звернення: 19.03.2023).
22. Уроки Python. URL: <https://www.youtube.com/@ProgerSchool/playlists>
(дата звернення: 15.03.2023).
23. Чат-бот 7AAC. URL: <https://sud.ua/ru/news/sud-info/155487-u-7aas-zyavivsyia-telegram-bot-yak-nim-koristuvatisya> (дата звернення: 07.04.2023).
24. Що таке чат-бот в Telegram. URL: <https://marchenko.marketing/school-take-chat-bot-ta-yak-stvoriti-bota-v-telegram/> (дата звернення: 03.04.2023).

25. Як створити бота в Телеграм. URL:

https://www.youtube.com/watch?v=fqQMbXvzbNw&ab_channel=MarchenkoMarketing (дата звернення: 04.04.2023).

26. Як створити телеграм-бота за допомогою Python. URL:

<https://www.freecodecamp.org/ukrainian/news/yak-stvoryty-telehram-bota-za-dopomohoyu-python/> (дата звернення: 04.04.2023).

27. Як створити Telegram чат-бот. URL: <https://sendpulse.com/ru/knowledge-base/chatbot/telegram/create-telegram-chatbot> (дата звернення: 04.04.2023).

28. Bot menu button. URL: <https://core.telegram.org/api/bots/menu>

(дата звернення: 15.04.2023).

29. Bots. URL: <https://core.telegram.org/api/bots>

(дата звернення: 25.03.2023).

30. IntelliJ IDEA. URL: <https://www.jetbrains.com/idea/>

(дата звернення: 01.03.2023).

31. IntelliJ IDEA. URL: <https://www.youtube.com/@RaghavPal/playlists>

(дата звернення: 01.03.2023).

32. IntelliJ IDEA: Debugger Upskill. URL:

https://www.youtube.com/watch?v=ZbAVoypAlbE&ab_channel=IntelliJIDEAbyJetBrains (дата звернення: 01.03.2023).

33. Python. URL: https://www.youtube.com/@Lazy_IT/playlists

(дата звернення: 06.03.2023).

34. Python telegram bot. URL: <https://docs.python-telegram-bot.org/en/stable/> (дата звернення: 05.03.2023).

35. Telegram APIs. URL: <https://core.telegram.org/api>

(дата звернення: 19.03.2023).

36. Telegram FAQ. URL: <https://telegram.org/faq>

(дата звернення: 19.03.2023).

37. Test your bot. URL: <https://learn.microsoft.com/en-us/power-virtual-agents/authoring-test-bot?tabs=preview> (дата звернення: 23.05.2023).

38.The Chatbot Testing Checklist. URL: <https://chatbotslife.com/the-chatbot-testing-checklist-tools-techniques-and-metrics-to-include-in-your-testing-strategy-3478a74eb215> (дата звернення: 20.05.2023).

39.Thought box. URL: <https://www.webio.com/guide/chatbot-testing-checklist> (дата звернення: 08.05.2023).

40.Five steps to make chatbot. URL: <https://www.getjenny.com/chatbot-project-5-step-checklist> (дата звернення: 01.03.2023).

Додатки

```
import telebot
import threading
import json
import datetime

# Константи
TOKEN = 'TOKEN HERE'
TASKS_FILE = 'tasks.json'
TIMERS_FILE = 'timers.json'

# Ініціалізуємо бота
bot = telebot.TeleBot(TOKEN)

# Ініціалізуємо дані
tasks = []
timers = {}

#=====Команда /start
@bot.message_handler(commands=['start'])
def start(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Ласкаво просимо до бота-тайм-менеджера!
\nВикористайте /help для перегляду списку доступних команд.')

#=====Команда /help
@bot.message_handler(commands=['help'])
def help(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, '/new_task      - Створити нове завдання.\n'
                               '/list_tasks   - Список всіх завдань.\n'
                               '/edit_task    - Редагувати існуюче завдання.\n'
                               '/set_timer    - Встановити таймер для
завдання.\n'
                               '/cancel_timer - Скасувати таймер.\n'
                               '/delete_task  - Видалити існуюче завдання.\n')

#=====Команда /new_task
@bot.message_handler(commands=['new_task'])
def new_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть назву завдання:')
    bot.register_next_step_handler(message, new_task_name)

def new_task_name(message):
    chat_id = message.chat.id
    task_name = message.text
    bot.send_message(chat_id, 'Введіть опис завдання:')
    bot.register_next_step_handler(message, new_task_description, task_name)

def new_task_description(message, task_name):
    chat_id = message.chat.id
    task_description = message.text

    if len(tasks) > 0:
        last_task = tasks[-1]
        last_task_id = last_task['task_id']
        new task id = last task id + 1
```

```

else:
    new_task_id = 1

user_id = message.from_user.id
task = {
    'user_id': user_id,
    'task_id': new_task_id,
    'task_name': task_name,
    'task_description': task_description,
    'status': 'Виконується'
}
tasks.append(task)
save_tasks()
bot.send_message(chat_id, 'Завдання успішно створено!')
bot.send_message(chat_id, get_task_message(task))

#=====
=====Команда /list_tasks
@bot.message_handler(commands=['list_tasks'])
def list_tasks(message):
    chat_id = message.chat.id
    user_id = message.from_user.id
    user_tasks = [task for task in tasks if task['user_id'] == user_id]
    if len(user_tasks) == 0:
        bot.send_message(chat_id, 'Завдань не знайдено.')
    else:
        for task in user_tasks:
            bot.send_message(chat_id, get_task_message(task))

#=====
=====Команда /edit_task
@bot.message_handler(commands=['edit_task'])
def edit_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, edit_task_id)

def edit_task_id(message):
    chat_id = message.chat.id
    task_id = message.text
    if not task_id.isdigit():
        bot.send_message(chat_id, 'Ви вказали невірний формат.')
        return
    task_id = int(task_id)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        task = tasks[task_index]
        bot.send_message(chat_id, 'Введіть нову назву завдання:')
        bot.register_next_step_handler(message, edit_task_name, task)

def edit_task_name(message, task):
    chat_id = message.chat.id
    task_name = message.text
    task['task_name'] = task_name
    save_tasks()
    bot.send_message(chat_id, 'Введіть новий опис завдання:')
    bot.register_next_step_handler(message, edit_task_description, task)

def edit_task_description(message, task):
    chat_id = message.chat.id
    task_description = message.text

```

```

task['task_description'] = task_description
save_tasks()
bot.send_message(chat_id, get_task_message(task))

#=====
=====Команда /delete_task
@bot.message_handler(commands=['delete_task'])
def delete_task(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, delete_task_id)

def delete_task_id(message):
    chat_id = message.chat.id
    task_id = message.text
    if not task_id.isdigit():
        bot.send_message(chat_id, 'Ви вказали невірний формат.')
        return
    chat_id = message.chat.id
    task_id = int(message.text)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        tasks.pop(task_index)
        save_tasks()
        bot.send_message(chat_id, 'Завдання успішно видалено!')

#=====
=====Команда /set_timer
@bot.message_handler(commands=['set_timer'])
def set_timer(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, set_timer_id)

def set_timer_id(message):
    chat_id = message.chat.id
    task_id = message.text
    if not task_id.isdigit():
        bot.send_message(chat_id, 'Ви вказали невірний формат.')
        return
    task_id = int(task_id)
    task_index = get_task_index(task_id)
    if task_index is None:
        bot.send_message(chat_id, 'Завдання не знайдено.')
    else:
        task = tasks[task_index]
        if task_id in timers:
            bot.send_message(chat_id, 'На це завдання вже встановлений таймер.')
        else:
            bot.send_message(chat_id, 'Введіть час у хвилинах:')
            bot.register_next_step_handler(message, set_timer_time, task)

def set_timer_time(message, task):
    chat_id = message.chat.id
    time = int(message.text)
    end_time = datetime.datetime.now() + datetime.timedelta(minutes=time)
    timers[task['task_id']] = {'end_time': end_time.timestamp(), 'chat_id':
chat_id, 'message_id': None}
    save_timers()
    bot.send_message(chat_id, 'Таймер встановлено успішно!')

```

```

        timer_id = task['task_id']
        timer_interval = time * 60
        timer = threading.Timer(timer_interval, timer_callback, args=[chat_id,
timer_id])
        timer.start()

def timer_callback(chat_id, timer_id):
    task_index = get_task_index(timer_id)
    if task_index is not None:
        task = tasks[task_index]
        bot.send_message(chat_id, f'Таймер для завдання "{task["task_name"]}"
закінчився!')
        del timers[timer_id]
        save_timers()

#=====
=====Команда /cancel_timer
@bot.message_handler(commands=['cancel_timer'])
def cancel_timer(message):
    chat_id = message.chat.id
    bot.send_message(chat_id, 'Введіть ID завдання:')
    bot.register_next_step_handler(message, cancel_timer_id)

def cancel_timer_id(message):
    chat_id = message.chat.id
    task_id = message.text
    if not task_id.isdigit():
        bot.send_message(chat_id, 'Ви вказали невірний формат.')
        return
    chat_id = message.chat.id
    task_id = int(message.text)
    if task_id not in timers:
        bot.send_message(chat_id, 'Таймер не знайдено.')
    else:
        timer = timers[task_id]
        if timer['message_id'] is not None:
            bot.delete_message(chat_id, timer['message_id'])
        del timers[task_id]
        save_timers()
        bot.send_message(chat_id, 'Таймер успішно скасовано!')

def get_task_index(task_id):
    for i in range(len(tasks)):
        if tasks[i]['task_id'] == task_id:
            return i
    return None

def get_task_message(task):
    task_message = f"ID: {task['task_id']}\nТема: {task['task_name']}\nОпис:
{task['task_description']}\nСтатус: {task['status']}"
    return task_message

def save_tasks():
    with open(TASKS_FILE, 'w') as f:
        tasks_to_save = [dict(task, user_id=str(task['user_id'])) for task in
tasks]
        json.dump(tasks_to_save, f, indent=4)

def load_tasks():
    with open('tasks.json', 'r') as f:
        tasks = json.load(f)
    for task in tasks:
        if 'user_id' in task:

```

```
        task['user_id'] = int(task['user_id'])
    return tasks

def save_timers():
    with open(TIMERS_FILE, 'w') as f:
        json.dump(timers, f)

def load_timers():
    global timers
    try:
        with open(TIMERS_FILE, 'r') as f:
            timers = json.load(f)
    except FileNotFoundError:
        timers = {}

load_tasks()
load_timers()

bot.polling(none_stop=True)
```