Міністерство освіти і науки
України Національний
технічний університет

«Дніпровська політехніка

ЗВІТ

Про виконання Практичної роботи №4-5

З дисципліни «Аналіз програмного забезпечення»

Виконав

Студент гр. 124-22-1

Возіянов Денис Олександрович

Перевірив

Доцент ФІТ

Мінєєв Олександр Сергійович

Дніпро

2025

Скрам-завдання для одного гравця (Python)

Назва Спринту: "Персональний помічник для відстеження часу"

Тривалість Спринту: 3 дні (або 3 робочих сесії)

Мета: Створити просту консольну програму на Python, яка допомагає відстежувати, на що ви витрачаєте свій час.

Беклог Продукту (Product Backlog)

Ось список завдань, упорядкований за пріоритетом (від найважливішого до менш важливого). Моє завдання — виконати якомога більше з верхньої частини списку за цей спринт.

Обов'язкові (Must Have)

[US-001] Як користувач, я хочу вводити назву завдання, яким займаюся, щоб програма почала відлік часу.

[US-002] Як користувач, я хочу натискати Enter (або вводити команду "stop"), щоб зупинити відлік часу для поточного завдання.

[US-003] Як користувач, я хочу, щоб програма зберігала назву завдання та витрачений на нього час (у секундах/хвилинах).

Важливі (Should Have):

4. [US-004] Як користувач, я хочу бачити список усіх завдань та сумарний витрачений на них час у кінці робочого дня.

Бажані (Could Have):

5. [US-005] Як користувач, я хочу, щоб програма показувала мені статистику (наприклад, найпопулярніші завдання) після завершення роботи.

План Спринту (Sprint Planning)

Моя мета на цей спринт: Реалізувати всі завдання з категорії "Обов'язкові (Must Have)" ([US-001], [US-002], [US-003]).

Технічні нотатки:

Використовую модуль time для відстеження часу (time.time()).

Для зберігання даних використовую словник або список.

Весь взаємодія - через консоль (input(), print()).

Програмний код

```
import time
import json
import os

# Файл для зберігання даних
DATA_FILE = "time_tracker_data.json"

def load_data():
    """Завантажує дані з файлу, якщо він існує."""
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, 'r') as f:
            return json.load(f)
    return {}

def save_data(data):
    """Зберігає дані у файл."""
    with open(DATA_FILE, 'w') as f:
        json.dump(data, f)

def start_tracking():
    """Основна функція для відстеження часу."""
    print("=== Персональний трекер часу ===")
    print("Команди: 'stop' - завершити роботу, 'history' - переглянути
історію")

    # Завантаження попередніх даних
    time_data = load_data()

    while True:
        # Запит на назву завдання (User Story #1)
```

```
        task_name = input("\nВведіть назву завдання (або 'stop'/'history'):
").strip()

        if task_name.lower() == 'stop':
            print("Дякую за використання трекера! Дані збережено.")
            break
        elif task_name.lower() == 'history':
            show_history(time_data)
            continue
        elif not task_name:
            print("Будь ласка, введіть назву завдання.")
            continue

        # Запам'ятовування часу початку (User Story #1)
        start_time = time.time()
        print(f"Таймер запущено для завдання: '{task_name}'")
        print("Натисніть Enter, щоб зупинити таймер...")

        # Очікування команди зупинки (User Story #2)
        input()

        # Розрахунок витраченного часу (User Story #2)
        end_time = time.time()
        elapsed_time = end_time - start_time
        elapsed_minutes = round(elapsed_time / 60, 2)

        print(f"Час виконання: {elapsed_minutes} хвилин")

        # Зберігання даних (User Story #3)
        if task_name in time_data:
            time_data[task_name] += elapsed_minutes
        else:
            time_data[task_name] = elapsed_minutes

        save_data(time_data)
        print(f"Дані для '{task_name}' оновлено!")

def show_history(data):
    """Показує історію витраченого часу (User Story #4)."""
    if not data:
        print("Історія відстеження порожня.")
        return

    print("\n--- Історія витраченого часу ---")
    total_time = 0
    for task, minutes in data.items():
```

```
            print(f"- {task}: {minutes} хв")
            total_time += minutes

        print(f"Загальний час: {round(total_time, 2)} хв")

        # Проста статистика (User Story #5)
        if data:
            most_time_task = max(data, key=data.get)
            print(f"Найбільше часу витрачено на: '{most_time_task}'")

# Запуск програми
if __name__ == "__main__":
    try:
        start_tracking()
    except KeyboardInterrupt:
        print("\n\nРоботу перервано. Дані збережено.")
    except Exception as e:
        print(f"Сталася помилка: {e}")
```

Тестування(лише приклад)

```
=== Персональний трекер часу ===
Команди: 'stop' - завершити роботу, 'history' - переглянути історію

Введіть назву завдання (або 'stop'/'history'): Тестування мого трекера часу
Таймер запущено для завдання: 'Тестування мого трекера часу'
Натисніть Enter, щоб зупинити таймер...

Час виконання: 1.29 хвилин
Дані для 'Тестування мого трекера часу' оновлено!

Введіть назву завдання (або 'stop'/'history'): stop
Дякую за використання трекера! Дані збережено.

=== Персональний трекер часу ===
Команди: 'stop' - завершити роботу, 'history' - переглянути історію

Введіть назву завдання (або 'stop'/'history'): history

--- Історія витраченого часу ---
- Тестування мого трекера часу: 1.29 хв
Загальний час: 1.29 хв
Найбільше часу витрачено на: 'Тестування мого трекера часу'

Введіть назву завдання (або 'stop'/'history'): stop
Дякую за використання трекера! Дані збережено.
```