1. Оценка по количеству строк кода (SLOC)

Ты прислал несколько файлов кода на C# и Python. Начнем с определения приблизительного количества строк кода в этих файлах.

Оценка количества строк кода:

- Файл с клиентской частью на C# имеет около 1089 строк написанных вручную без шаблонов.
- Файл с серверной частью на Python включает:

    1. GPT-класс для генерации поздравлений (23 строк).
    2. Класс для взаимодействия с базой данных (42строк).
    3. Класс для работы с Telegram (23 строк).
    4. Класс для работы с Json (35 строк).
    5. Основная логика приложения (35 строк). Таким образом, серверная часть занимает около 158 строк кода.

Итого: **1247 строк кода (SLOC)** для всей системы.

---

2. Оценка по функциональным точкам (Function Points)

Метод функциональных точек оценивает размер ПО на основе количества входов, выходов, запросов, файлов и интерфейсов, а затем взвешивает их по сложности.

1. **Количество внешних входов**:

    1. Входные данные для генерации поздравлений через OpenAI.

    Оценка: 1 входа (низкая сложность).

2. **Количество внешних выходов**:

    1. Генерация поздравлений на основе интересов пользователя.
    2. Отправка сообщений пользователю через Telegram.

    Оценка: 2 выхода (средняя сложность).

3. **Количество пользовательских запросов**:
    - Запрос к базе данных для (добавления/обновления/удаления) друга.

    Оценка: 1 пользовательских запроса (низкая сложность).

4. **Внутренние логические файлы (ILF)**:
    - Таблица с пользователями, их днями рождения и интересами в базе данных.

    Оценка: 1 внутренний логический файл (низкая сложность).

5. **Внешние интерфейсные файлы (EIF)**:
    - Работа с OpenAI API для генерации текста поздравлений.
    - Работа с Telegram API для отправки сообщений.

    Оценка: 2 внешних интерфейсных файла (средняя сложность).

3. Расчет функциональных точек

Теперь рассчитаем **функциональные точки** для каждого элемента. Взвешиваем по сложности (низкая, средняя, высокая) на основе стандарта IFPUG:

Элемент	Количество	Сложность	FP (функциональные точки)
Внешние входы (EI)	1	Низкая	1 × 3 = 3
Внешние выходы (EO)	2	Средняя	2 × 5 = 10
Пользовательские запросы (EQ)	1	Низкая	1 × 3 = 3
Внутренние логические файлы (ILF)	1	Низкая	1 × 7 = 7
Внешние интерфейсные файлы (EIF)	2	Средняя	2 × 7 = 14

Итого: 37 функциональных точек (*UFP*).

4. Оценка трудозатрат

CFP:

1. Требуется ли резервное копирование данных --------0
2. Требуется обмен данными--------5
3. Используются распределенные вычисления--------0
4. Важна ли производительность--------2
5. Программа выполняется на сильно загруженном оборудовании--------5
6. Требуется ли оперативный ввод данных--------0
7. Используется много форм для ввода данных--------3
8. Поля базы данных обновляются оперативно--------5
9. Ввод, вывод, запросы являются сложными--------0
10. Внутренние вычисления сложны--------0
11. Код предназначен для повторного использования--------5
12. Требуется преобразование данных и установка программы--------0
13. Требуется много установок в различных организациях--------0
14. Требуется поддерживать возможность настройки и простоту использования--------5

Итого: 30 функциональных точек (*CFP*).

$$VAF=0.65+(0.01\times\Sigma CFP) =0.95$$

$$AFP=UFP\times VAF = \underline{\mathbf{0.95*37 =35.15}}$$

Заключение:

- **Размер кода (SLOC)**: 1089 строк.
- **Функциональные точки (FP)**: 37.
- AFP= **35.15**