Документація до проекту "Server"

1. Загальний опис програми

Ця програма реалізує клієнт-серверну систему з використанням мови програмування C++ і бібліотеки Winsock. Сервер дозволяє підключати декількох клієнтів одночасно, обробляє їхні запити, пересилає повідомлення між клієнтами на основі унікальних ідентифікаторів, а також веде журнал усіх дій.

Основні можливості програми:

- Підтримка одночасного підключення кількох клієнтів.
- Пересилання повідомлень між клієнтами.
- Ведення журналу операцій.
- Надсилання клієнтам їх унікальних ідентифікаторів.
- Автоматичне оновлення списку доступних клієнтів.

2. Структура програми

2.1. Сервер

Основні функції сервера:

1. Прийом клієнтів і призначення унікальних ідентифікаторів.
2. Ведення списку активних клієнтів.
3. Логування дій сервера.
4. Пересилання повідомлень між клієнтами.

Ключові функції:

- `logMessage`: Запис повідомлень до файлу журналу.
- `broadcastClientList`: Надсилає всім клієнтам оновлений список доступних клієнтів у форматі:

```
CLIENTS: <ID1> <ID2> <ID3>
```

- `handleClient`: Обробляє повідомлення від конкретного клієнта.
- `main`: Ініціалізація сервера і основний цикл роботи.

```
void logMessage(const string& message) {
    lock_guard<mutex> lock(clientMutex);
    logFile << message << endl;
    cout << message << endl;
}
```

Лістинг 1: Код функції logMessage

2.2. Клієнт

Основні функції клієнта:

1. Надсилання повідомлень іншим клієнтам.
2. Отримання списку активних клієнтів.
3. Отримання власного унікального ідентифікатора.

Ключові функції:

- `receiveMessages`: Постійне отримання повідомлень від сервера:

```
Received from server: CLIENTS: <ID1> <ID2> <ID3>
Received from server: Hello from Client <ID>
```

- `main`: Ініціалізація клієнта, підключення до сервера і взаємодія з користувачем.

```
void receiveMessages(SOCKET clientSocket) {
    wchar_t buffer[1024];
    int bytesReceived;

    while (true) {
        bytesReceived = recv(clientSocket, (char*)buffer, sizeof(
            buffer), 0);
        if (bytesReceived > 0) {
            buffer[bytesReceived / sizeof(wchar_t)] = L'\0';
            wcout << L"Received from server: " << buffer << endl;
        } else if (bytesReceived == 0) {
            cout << "Connection closed by server." << endl;
            break;
        } else {
            cerr << "Error receiving message. Error code: " <<
                WSAGetLastError() << endl;
            break;
        }
    }
}
```

Лістинг 2: Функція receiveMessages для клієнта

3. Логіка роботи програми

3.1. Сервер

1. Сервер слухає підключення клієнтів і приймає їх.
2. Кожному клієнту присвоюється унікальний ідентифікатор.
3. Сервер пересилає повідомлення між клієнтами на основі їхніх ідентифікаторів.
4. Сервер оновлює список доступних клієнтів і розсилає його всім підключеним клієнтам.

3.2. Клієнт

1. Клієнт підключається до сервера і отримує свій ідентифікатор.
2. Клієнт може:
    - Надсилати повідомлення іншим клієнтам.
    - Отримувати повідомлення від інших клієнтів.
    - Переглядати список доступних клієнтів.

4. Інструкція з використання

Запуск сервера

Запустіть сервер за допомогою:

```
./server.exe
```

Сервер починає слухати підключення клієнтів.

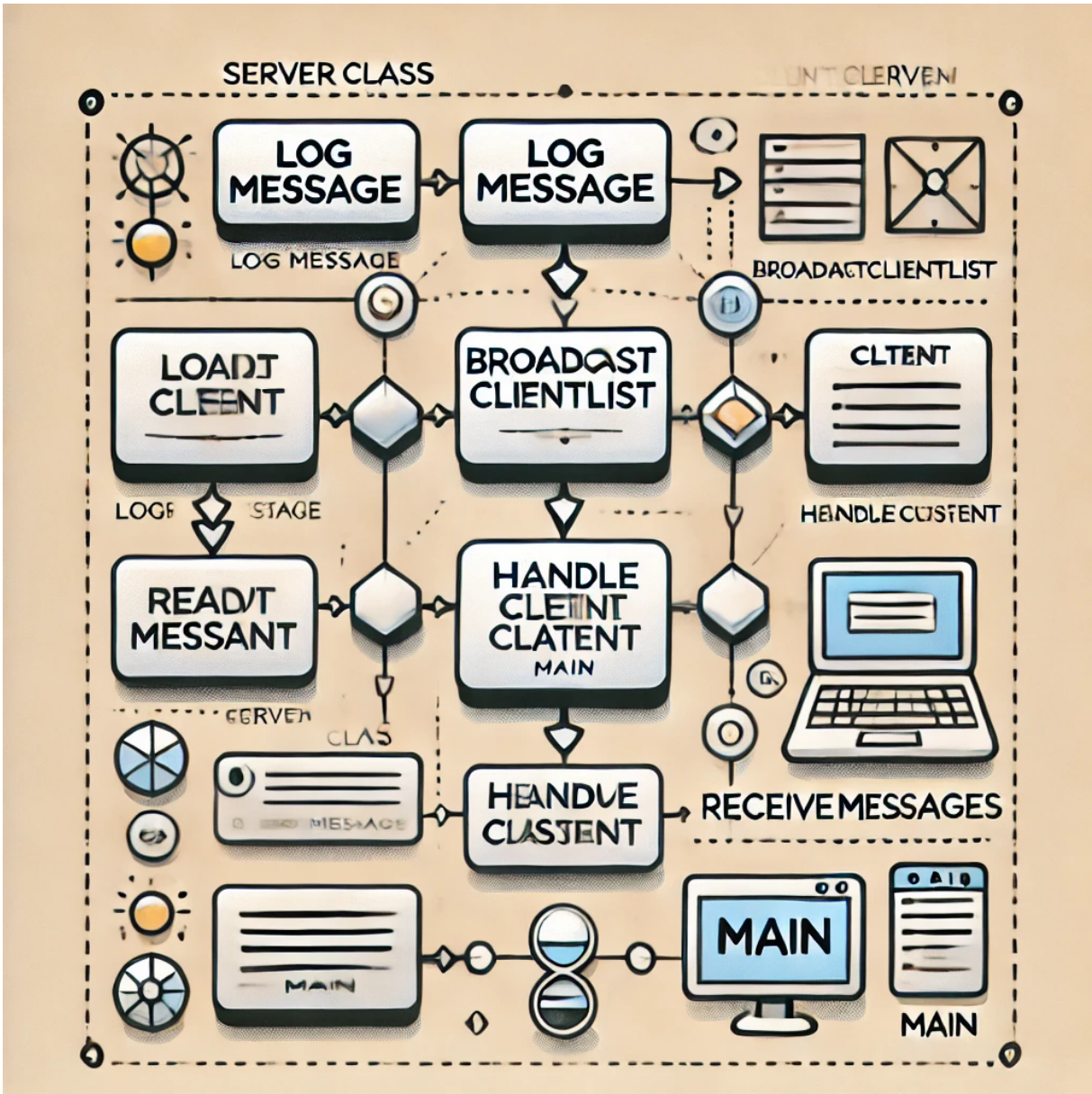
Запуск клієнта

Запустіть клієнт за допомогою:

```
./client.exe
```

Клієнт підключається до сервера і отримує унікальний ідентифікатор.

5. UML-діаграма



Опис UML-діаграми:

- **Клас Server:**
  - Атрибути:
    - `clients: map<int, SOCKET>` — Зберігає список підключених клієнтів.
    - `clientIDCounter: int` — Лічильник для унікальних ID клієнтів.
    - `logFile: ofstream` — Файл для логування.
  - Методи:
    - `logMessage(string)` — Записує повідомлення в журнал.
    - `broadcastClientList()` — Розсилає список клієнтів.
    - `handleClient(int, SOCKET)` — Обробляє клієнтські запити.
- **Клас Client:**
  - Атрибути:

    * `clientSocket: SOCKET` — Сокет для з'єднання з сервером.
  - Методи:
    * `receiveMessages(SOCKET)` — Отримує повідомлення від сервера.
    * `main()` — Головна функція для взаємодії з сервером.

Короткий опис:

- Сервер приймає клієнтів, зберігає їх у мапі та пересилає повідомлення між ними.
- Клієнт підключається до сервера, отримує свій ID і може взаємодіяти з іншими клієнтами через сервер.

6. Висновки

Проект "Server"демонструє реалізацію базової клієнт-серверної системи. У майбутньому можна додати:

- Шифрування повідомлень.
- Аутентифікацію клієнтів.
- Підтримку кількох серверів.