**Multi-User Chat Application with File Handling**

**Authors:**

1. Numan Maroof (F202340126)
2. Usman Tariq (F2023408116)

**Abstract:**

This project implements a **Multi-User Chat Application** using C++ programming language. The application allows users to engage in private and group chats while storing chat history in text files. The project utilizes essential data structures such as **Linked Lists**, **Queues**, and **Vectors** to manage contacts, messages, and group members. The application supports features like adding/removing contacts, creating/joining groups, and saving chat history to files. The implementation demonstrates the practical use of data structures in real-world applications.

**Keywords:** Chat Application, Linked List, Queue, File Handling, C++.

**Introduction:**

The project aims to create a **Multi-User Chat Application** that allows users to communicate in private and group chats. The application stores chat history in text files, ensuring data persistence. The project is implemented in C++ and uses data structures like **Linked Lists** for managing contacts and groups, **Queues** for storing messages, and **Vectors** for managing group members. The application provides a user-friendly menu-driven interface for interacting with the system.

**Problems:**

1. Some problems were faced dealing with file handling. Especially while trying to load data on startup, but all problem were tackled.

**Features:**

The application includes the following features:

1. **Private Chats:**
   * Add new contacts.
   * Remove existing contacts.
   * Send and receive messages in private chats.
   * Store private chat history in text files.
2. **Group Chats:**
   * Create new groups.
   * Add or remove members from groups.
   * Send and receive messages in group chats.
   * Store group chat history in text files.
3. **File Handling:**
   * Automatically create and update text files for private and group chats.
   * Load chat history from files when the application starts.

**Implementation Details:**

**Data Structures Used**

1. **Linked List:**
   * Used to manage contacts and groups.
   * Each contact or group is represented as a node in the linked list.
2. **Queue:**
   * Used to store messages for each contact or group.
   * Messages are stored in a First-In-First-Out (FIFO) manner.
3. **Vector:**
   * Used to store group members in group chats.

**Classes and Functions:**

1. **node Class:**
   * Represents a node in the message queue.
   * Contains a string for the message and a pointer to the next node.
2. **queuemsgs Class:**
   * Implements a queue using a linked list.
   * Functions:
     + insert(): Adds a message to the queue.
     + dele(): Deletes all messages from the queue.
     + display(): Displays all messages in the queue.
     + storem(): Stores messages in a text file.
3. **node Class:**
   * Represents a contact in the private chat list.
   * Contains a queue for messages, contact name, and a pointer to the next contact.
4. **Privatechats Class:**
   * Manages the list of private contacts.
   * Functions:
     + add(): Adds a new contact.
     + del(): Deletes a contact.
     + display(): Displays all contacts.
     + enter(): Adds or displays messages for a contact.
     + store(): Stores chat history in a file.
5. **nodec Class:**
   * Represents a group in the group chat list.
   * Contains a queue for messages, a vector for group members, and a pointer to the next group.
6. **groupchat Class:**
   * Manages the list of groups.
   * Functions:
     + create(): Creates a new group.
     + exit(): Deletes a group.
     + enter(): Adds or displays messages for a group.
     + add(): Adds a member to a group.
     + remove(): Removes a member from a group.
     + traverse(): Collects messages from group members.
     + store(): Stores group chat history in a file.

**File Handling:**

* The application creates two subfolders: **Privatechats** and **Groupchats**.
* Private chat histories are stored in text files named <FirstName><LastName>.txt.
* Group chat histories are stored in text files named <GroupName>.txt.
* The application loads chat history from these files when it starts.

**Algorithms and Techniques:**

1. **Linked List Traversal:**
   * Used to search, add, and delete contacts or groups.
2. **Queue Operations:**
   * Messages are inserted at the end of the queue and displayed in FIFO order.
3. **File I/O:**
   * Chat histories are read from and written to text files using ifstream and ofstream.

**Outcomes:**

* The application successfully implements private and group chat functionalities.
* Chat histories are stored and retrieved from text files.
* The use of data structures ensures efficient management of contacts, groups, and messages.

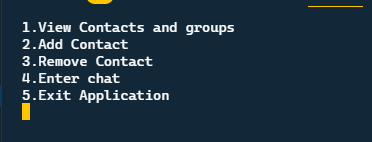
**Conclusion:**

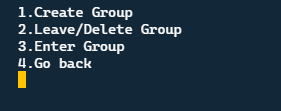
The Multi-User Chat Application demonstrates the effective use of data structures like Linked Lists, Queues, and Vectors in a real-world application. The project highlights the importance of file handling for data persistence and provides a user-friendly interface for managing chats. Future enhancements could include encryption for secure messaging and a graphical user interface (GUI).

**References:**

1. C++ Documentation: <https://en.cppreference.com/>
2. Data Structures and Algorithms in C++, 2nd Edition by Michael T. Goodrich.
3. File Handling in C++: <https://www.geeksforgeeks.org/file-handling-c-classes/>
4. **Screenshots**

**Menus:**







**Chat box:**

