SocialNet Simulator
README Documentation

1. Overview

This project implements a backend simulator for a social networking platform using C++. The simulator provides a command-line interface where users can register, make friends, post content, view posts, and explore their social connections.

The system uses two main data structures:

- A **Graph** to represent users and friendships.
- An **AVL Tree** to store and order user posts by creation time.
- All usernames are case-insensitive.

2. File Structure

The project contains the following files:

- `User.hpp` – user data structure
- `AVL_Trees.hpp` – AVL tree for post management
- `SocialNet_Simulator.hpp` – Graph implementation
- `main.cpp` – command-line interface and command parser
- `run.sh` – compile and run script
- `README.pdf`

3. Core Architecture

3.1 User Class (`User.hpp`)

Each user object stores:

- `string username` – stored in lowercase for uniformity.
- `unordered_set<User*> friends` – provides $O(1)$ friend lookups.
- `Node* root` – root of an AVL Tree storing posts.
- `int number_of_posts` – total post count.

3.2 AVL Tree Node (`AVL_Trees.hpp`)

Each post is represented as a node containing:

- `val` – the post content (string)
- `created_timestamp` – timestamp when post was added
- `height, left, right` – AVL tree balancing and structure fields

AVL rotations ensure that the tree is balances for (logn) lookups and insertion.

3.3 Graph Class (`SocialNet_Simulator.hpp`)

The `Graph` class integrates users and friendships, manages posts, and handles all operations through the command interface. It uses an `unordered_map<string, User*>` for efficient user access.

4. Command Interface (`main.cpp`)

The program reads commands line-by-line from `stdin` using a loop. Each command is parsed using a `stringstream`.

The interface provides clear error messages for all invalid or incomplete inputs:

- Missing username → `"Please Enter Username!"`
- Missing friend's username → `"Please Enter Friend's Username!"`
- Missing post → `"Please Enter Post Content!"`
- Invalid command → `"PLEASE ENTER A VALID COMMAND!"`

The command loop exits when the `EXIT` command is entered.

4. Functionality and Error Handling

Below is a detailed description of each function's purpose and its printed messages for invalid inputs.

- **ADD_USER ⟨username⟩**
  Adds a new user to the network.
  On success: Prints `"User Added!"`
  If username is empty: Prints `"Please Enter Username!"`

- **ADD_FRIEND ⟨username1⟩⟨username2⟩**
  Creates a bidirectional friendship between `username1` and `username2`.
  On success: Prints `"⟨username1⟩and ⟨username2⟩are now friends!"`
  If either user not found: Prints `"Username NOT Found!"`

- **LIST_FRIENDS ⟨username⟩**
  Lists all friends of the user in alphabetical order.
  On success: Prints each friend on a new line.
  If user not found: Prints `"Username NOT Found!"`
  If user has no friends: Prints `"No Friends!"`

- **SUGGEST_FRIENDS ⟨username⟩⟨N⟩**
  Suggests up to N users who are "friends of a friend" but not already friends. Results are ranked by number of mutual friends (descending), with ties broken alphabetically.
  If user not found: Prints `"Username NOT Found!"`
  If N<0: Prints `"Please Enter a Valid Number!"`
  If no suggestions: Prints `"No Available Suggestions!"`

- **DEGREES_OF_SEPARATION ⟨username1⟩⟨username2⟩**
  Uses BFS to compute the shortest friendship path between two users.
  On success: Prints the degree (0 for self, 1 for direct friends, etc.)
  If disconnected: Prints `"-1"`
  If user not found: Prints `"Username NOT Found!"`

- **ADD_POST ⟨username⟩⟨post_content⟩**
  Adds a new post to the user's AVL tree, timestamped with the current time.
  On success: Prints `"Post Added!"`
  If user not found: Prints `"Username NOT Found!"`

- **OUTPUT_POSTS ⟨username⟩⟨N⟩**
  Displays the user's latest N posts.
  If N = -1: Prints all posts.
  If N<0: Prints `"Please Enter a Valid Number!"`
  If user not found: Prints `"Username NOT Found!"`
  If no posts: Prints `"No posts!"`

5. Compilation and Execution

5.1 Manual Compilation

```
g++ -std=c++17 main.cpp -o socialnet
./socialnet
```

5.2 Automated Script (`run.sh`)

A shell script is provided to automate compilation and execution.

Usage:

1. Make executable: `chmod +x run.sh`
2. Run: `./run.sh`

8. Design Highlights

- **Efficiency:** $O(1)$ lookups, $O(\log N)$ post insertion, $O(V + E)$ BFS.
- **Clean Error Messages:** Every invalid or missing input is handled gracefully.
- **Modular Design:** Independent, reusable classes for clarity and maintainability.