

SocialNet Simulator

Long Assignment Report

Rami Jay Ketanbhai
Entry No: 2024CS10510

November 4, 2025

Contents

1	Introduction	2
2	Files Used	2
3	How to Run	2
3.1	Linux / macOS : compile.sh	2
3.2	Windows : run.ps1	2
4	Commands	2
5	Data Structures Used	3
6	Error Handling	3

1 Introduction

This project implements a simplified *SocialNet Simulator* inspired by social media platforms. Using self-implemented **graphs**, **AVL trees** and other data structures.

2 Files Used

The code base is organised into the following files:

- **main.cpp** – Main files. Handles inputs and feeds to the socialnet system.
- **AVL.hpp** – Defines the **AVLTree** class and involves commands for posts.
- **input.txt** - Allows user to give several inputs at once.
- **user.hpp** – Used to maintain users in system and hold their post trees.
- **graph.hpp** – Custom graph implementation using hashmap and set to maintain friends and other operations.
- **compile.sh** – Bash script to compile and run on Linux/macOS.
- **run.ps1** – PowerShell script to compile and run on Windows.

3 How to Run

Compilation and execution can be automated with the scripts provided. It also takes user input to choose between automated input through `input.txt` or interactive input.

3.1 Linux / macOS : `compile.sh`

```
chmod +x compile.sh  
./compile.sh
```

The first line is used to set bash file to make executable files. The script prompts the user to choose between running with ‘`input.txt`’ or interactive input.

3.2 Windows : `run.ps1`

```
.\run.ps1
```

Before first use, enable script execution:

```
Set-ExecutionPolicy -Scope CurrentUser RemoteSigned
```

The script asks the same question : use the input file or type commands interactively.

4 Commands

Each command is parsed in `main.cpp` using `istringstream`. Key commands include:

ADD_USER

`ADD_USER <username>` Creates a new user on social net with no initialised posts or friends.

ADD_FRIENDS

ADD_FRIENDS <user1> <user2> Sets user1 and user2 as friends of one another with an edge between them in graph implementation.

ADD_POSTS

ADD_POSTS <username> <content> Used to add posts for a given user. Stores content as well as timestamp of post creation.

LIST_FRIENDS

LIST_FRIENDS <username> Lists all users that are friends of a given user that is all adjacent edges in graph.

SUGGEST_FRIENDS

SUGGEST_FRIENDS <username> [N] Suggest top N friends of friends but not already friend of a given user based on their count of mutual friends and then in alphabetical order.

DEGREES_OF_SEPARATION

DEGREES_OF_SEPARATION <user1> <user2> Returns length of shortest path between two users.

OUTPUT_POSTS

OUTPUT_POSTS <username> [N] Output newest N posts from given user.

EXIT

I have defined this command in order to end the program.

5 Data Structures Used

- **AVL Tree** (`AVLTree.hpp`) – Each node stores posts, left, right pointers and creation timestamp. Used to get newest files efficiently.
- **Graph** (`graph.hpp`) – Used to maintain user friend list and for friends suggestions and degree of separation between two friends using BFS.
- **HashMap** (`graph.hpp`) – Used to access users from their usernames and their friends set in $O(1)$ time.
- **Set** (`graph.hpp`) – Used to maintain friends but only unique times (doesn't count multiple occurrences).

6 Error Handling

Error handling is incorporated in various modules to ensure robust system behavior during user interactions and input processing:

- When attempting to suggest friends, if the user does not exist in the system, an error message **User doesn't exist** is printed and an empty list is returned to prevent invalid operations.
- Doesn't check for case-sensitivity in users case and gives an error if we write names "a" and "A" as **User Already Exists** error.

- During post retrieval, if the requested number of posts exceeds the available count or is negative (set to -1), all posts are returned, preventing runtime errors.
- AVL tree insertion and rotations handle null pointers gracefully to maintain balanced trees, avoiding crashes during tree operations.
- Input parsing and command execution in the graph and user modules check for invalid or missing user names, printing clear error messages such as **Invalid User**. **Please add user.** when a user is not found.
- When outputting posts or suggesting friends, the system prints informative messages such as **No Suggestions** or **User doesn't exist** to indicate cases with no results or invalid requests.
- Compilation scripts and runtime execution provide status messages upon success or failure, including prompts for running with input files or interactively, guiding the user through proper usage.