

# SocialNet Simulator

## Long Assignment Report

Rami Jay Ketanbhai  
Entry No: 2024CS10510

November 4, 2025

### Contents

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

# 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.