Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

Social Media Friend Recommendation

Course Title: Algorithms Lab
Course Code: CSE 208
Section: D9

Students Details

Name	ID
Ashab Uddin	232002274

Submission Date: 29/04/2025
Course Teacher's Name: Farjana Akter Jui

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

1 Introduction **3**

1.1 Overview . . . 3

1.2 Motivation . . . 3

1.3 Problem Definition . . . 3

1.3.1 Problem Statement . . . 3

1.3.2 Complex Engineering Problem . . . 3

1.4 Design Goals/Objectives . . . 5

1.5 Application . . . 5

2 Implementation of the Project **6**

2.1 Introduction . . . 6

2.2 Project Details . . . 6

2.3 Project Implementation . . . 6

2.3.1 Objective . . . 6

2.3.2 Features . . . 6

2.3.3 Implementation . . . 7

2.3.4 Testing . . . 7

2.3.5 Development . . . 8

2.4 Implementation . . . 8

2.4.1 Implementation Workflow . . . 8

3 Performance Evaluation **10**

3.1 Simulation Procedure . . . 10

3.1.1 Results Analysis and Testing . . . 10

3.1.2 Results Overall Discussion . . . 11

3.1.3 Complex Engineering Problem Discussion . . . 11

4 Conclusion **12**

4.1 Conclusion 12
4.1.1 Discussion 12
4.1.2 Limitations 12
4.1.3 Scope of Future Work 13

Chapter 1

Introduction

1.1 Overview

The Social Media Friend Recommendation System is a Java-based system that allows users to log in, manage friendships, and receive friend recommendations using breadth-first search (BFS) and depth-first search (DFS). It is built using object-oriented programming with efficient data structures like lists, queues, and sets.

1.2 Motivation

In modern social networks, discovering new people and building meaningful connections can be challenging. This system is designed to simulate a scalable friend suggestion mechanism using graph traversal algorithms, enabling users to expand their social network intelligently.

1.3 Problem Definition

1.3.1 Problem Statement

1. Users on social media platforms often struggle to discover potential friends they might know. Manual discovery is inefficient, especially in large networks. This system automates friend discovery based on mutual connections.

1.3.2 Complex Engineering Problem

Developing an efficient recommendation system requires handling user authentication, dynamic friend networks, and real-time suggestion calculations without redundancy or loops. It also requires memory optimization and algorithm correctness.

Table 1.1: Summary of the attributes touched by the Recommendation System

Name of the P Attributess	**Explain how to address**
P1: Security	Basic security through username-based login. You've enhanced the system by adding a password-based login. This means users now need both a username and password to access their account, improving account security compared to the earlier version with just a username.
P2: Scalability	The system uses dynamic structures like ArrayList, Queue, and Set, which can handle growing numbers of users and connections. However, since there's no database and it's running in-memory, it's more suitable for small to medium-scale usage.
P3: User Experience	Simple console-based UI; functional and easy to navigate.
P4: Integration with Existing Systems	This project runs independently and doesn't connect with external platforms like Facebook, Google login, or cloud databases. It works well on its own, but integration would make it more powerful and realistic.
P5: Regulatory Compliance	Basic system; no specific compliance with data protection laws.

1.4 Design Goals/Objectives

- Implement user login and session handling.
- Allow users to send and accept friend requests.
- Maintain lists of users, friends, and requests.
- Suggest potential friends using BFS and DFS.

1.5 Application

This system can be integrated into any social networking platform to automate friend recommendations. Improver interaction and engagement by providing relevant connection suggestions and is an excellent educational project to understand graph algorithms.

- **Social Media Platforms:** Can be integrated into social networks to recommend potential friends based on mutual connections.
- **Online Communities:** Used by online forums or groups to suggest new members to connect with based on common interests or friends.
- **Educational Platforms:** Helps students connect with peers based on shared courses or academic interests.
- **Workplace Networks:** Can be used in enterprise social networks to suggest coworkers based on departments, projects, or mutual colleagues.
- **Gaming Platforms:** Suggests friends for multiplayer games based on current friends and gaming interests.
- **Event-Based Networking:** Useful in event apps to recommend people with similar interests attending the same event.

This system can be customized and scaled for larger social platforms by adding additional features like interests, location-based suggestions, or privacy controls.

Chapter 2

Implementation of the Project

2.1 Introduction

The project is developed in Java using object-oriented programming. It features menu-driven interaction and utilizes built-in data structures from the Java Collections Frame work to manage user accounts and relationships.

2.2 Project Details

The system supports user login, sending/accepting friend requests, and recommending friends. Each user has a name, list of friends, and pending requests. Recommendations are generated by traversing user connections.

2.3 Project Implementation

2.3.1 Objective

The objective of this project is to design and implement a Social Media Friend Recommendation System that allows users to log in, manage their friends, and receive friend recommendations based on mutual connections. The system uses Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms to recommend friends, enhancing the user experience on social platforms by suggesting potential friends with minimal input.

2.3.2 Features

The system provides the following features:

- **User Authentication:** Users can log in with a username and password to access their accounts.

- **Friend Management:** Users can send, accept, or reject friend requests, as well as view their current friends.
- **Friend Recommendations (BFS):** The system suggests friends based on mutual connections using the BFS algorithm.
- **Friend Recommendations (DFS):** The system also offers friend suggestions using the DFS algorithm, exploring deeper connections.
- **Account Creation:** New users can register with a username and create an account.
- **Logout Option:** Users can log out of their accounts for security and session management.

2.3.3 Implementation

The Social Media Friend Recommendation System is implemented in Java using object-oriented programming principles. The key components of the implementation are:

- **User Class:** Represents a user with attributes like ‘name‘, ‘friends‘, and ‘requests‘, and contains methods for managing friend relationships.
- **Login System:** Users authenticate using their username and password, ensuring secure access to their accounts.
- **Friendship Management:** Allows users to send and accept friend requests, as well as display their friend list.
- **Recommendation Algorithms:** The system uses BFS and DFS to suggest friends based on mutual connections and degrees of separation.
- **Data Structures:** Uses data structures like ‘ArrayList‘, ‘HashSet‘, and ‘Queue‘ to efficiently store and process users, friends, and requests.

The system operates in a command-line interface, where users interact with the program through a menu system that allows them to perform various actions.

2.3.4 Testing

Testing was performed at each stage of development to ensure the system operates correctly:

- **Unit Testing:** Individual methods and functions were tested to verify their correctness, such as adding a user, sending a request, and recommending friends.
- **Integration Testing:** Ensured that the login, friend request, and recommendation functionalities work together as expected.

- **Edge Case Testing:** Tested cases like sending requests to oneself, accepting already accepted requests, and handling empty or null data.
- **Manual Testing:** Performed by interacting with the system to ensure the user interface behaves as expected and the system is responsive.

2.3.5 Development

The development process of the Social Media Friend Recommendation System followed these key stages:

- **Requirement Analysis:** Determining the necessary features, such as user authentication, friend management, and friend recommendation.
- **Design:** Architected the system with a user-centered approach, ensuring clear functionality and an easy-to-use menu system.
- **Implementation:** Developed the system using Java, focusing on modularity and reusable code through object-oriented principles.
- **Testing and Debugging:** Thorough testing was conducted to identify bugs and optimize the system's performance.
- **Deployment:** The system is deployed in a local environment and can be tested by users to ensure full functionality.

2.4 Implementation

2.4.1 Implementation Workflow

The implementation of the Social Media Friend Recommendation System follows a structured and modular approach. The system is designed using Java and utilizes object-oriented principles for efficient user and friend management. Below is the step-by-step workflow of the implementation:

1. **User Registration and Login:**
    - At the beginning, users are presented with a main menu where they can either log in, register a new account, or exit the program.
    - For registration, the system asks for a unique username (and optionally a password, if implemented) and stores the new user in the user list.
    - If the user selects the login option, the system checks whether the entered username exists in the system. Upon successful verification, the user is granted access to the main user dashboard.
2. **Friendship Management:**

- Once logged in, users can manage their social network by sending or accepting friend requests.
- The `sendRequest()` method allows the user to send a friend request to another user by entering their username.
- The `acceptRequests()` method enables users to view and accept all pending friend requests, automatically updating both users' friend lists upon acceptance.

3. **Friend Recommendation (BFS and DFS):**

   - The system suggests potential friends using two classical graph traversal algorithms:
   - **Breadth-First Search (BFS):** This algorithm explores all direct friends and their connections in a level-wise manner. It is effective in finding friends-of-friends.
   - **Depth-First Search (DFS):** This algorithm dives deeper into the network and helps in exploring further relationships that are not directly connected.
   - These algorithms help the system identify users who are not yet friends with the current user but have mutual friends, making them suitable recommendations.

4. **Friend Request and Recommendation Flow:**

   - After receiving recommendations, the system displays a list of usernames that can be added as friends.
   - The user can choose to send requests to any of the recommended profiles.
   - This process strengthens the network graph and enhances future recommendation accuracy.

5. **Data Structure Utilization:**

   - The system uses `ArrayList` to maintain the list of users and their friends.
   - `HashSet` is used to avoid duplicate entries during recommendation processing.
   - `Queue` (for BFS) and recursion (for DFS) are used to implement the traversal logic efficiently.

6. **Logout and Session Handling:**

   - The user has the option to log out at any time.
   - On logout, the system resets the `currentUser` to `null` and returns to the main menu for new login or registration actions.

Chapter 3

Performance Evaluation

3.1 Simulation Procedure

The application was run in a local Java IDE, simulating multiple user interactions. Users were created, connected, and tested for friend suggestions via BFS and DFS.

3.1.1 Results Analysis and Testing

To ensure the reliability and correctness of the Social Media Friend Recommendation System, extensive testing was performed. The functionalities were tested with various edge cases and user scenarios. These included:

- **Sending Friend Request to Self:** The system correctly prevents users from sending friend requests to themselves by checking usernames before processing the request.
- **Accepting All Pending Requests:** The system accurately displays all pending friend requests and successfully updates both users' friend lists upon acceptance, ensuring mutual friendship integrity.
- **Suggesting Mutual Friends Across Multiple Layers:** Both BFS and DFS algorithms were tested for friend recommendation. BFS effectively provided broader recommendations through level-wise traversal, while DFS explored deeper and more distant relationships within the graph.
- **Duplicate Friend Prevention:** The system prevented duplicate requests or friend entries using Java `Set` structures and conditional checks during the traversal and acceptance process.
- **Login and Session Handling:** The login/logout mechanisms were verified to ensure only the authenticated user could perform social actions like sending/accepting requests or viewing friends.

3.1.2 Results Overall Discussion

The testing phase demonstrated that the system works as intended. The BFS algorithm proved to be more effective for users looking for broad and immediate mutual friend suggestions, especially in denser networks. In contrast, the DFS algorithm was capable of revealing hidden friend connections that may span across several nodes, making it more useful for deeper social exploration.

The results confirmed that both algorithms provided accurate, non-redundant recommendations and maintained the data integrity of the network.

3.1.3 Complex Engineering Problem Discussion

The core complexity of this system lies in handling dynamic social networks as graphs and ensuring efficient traversal without introducing logical errors like cycles or redundant links. Implementing BFS and DFS in a way that scales with user size required the careful selection of data structures such as:

- `ArrayList` for maintaining user and friend lists.
- `HashSet` for tracking visited nodes and eliminating duplicates.
- `Queue` and recursive functions for implementing BFS and DFS respectively.

Using object-oriented principles in Java allowed modularity and re-usability, making the code base easy to maintain and extend. The program structure ensured that each feature — from login to recommendation — interacted smoothly without compromising performance or security.

Chapter 4

Conclusion

4.1 Conclusion

4.1.1 Discussion

The Social Media Friend Recommendation System developed in this project successfully simulates the core functionality of a basic social networking platform. It incorporates essential features such as user login, user registration, sending and accepting friend requests, and suggesting potential friends using two powerful graph traversal algorithms: Breadth-First Search (BFS) and Depth-First Search (DFS).

This implementation showcases how data structures such as lists, sets, and queues, combined with object-oriented principles in Java, can be used to model and manage dynamic user networks effectively. The system demonstrates how graph-based algorithms can be applied in real-world applications like social media platforms for friend discovery.

4.1.2 Limitations

Despite its functional strengths, the current system has some limitations:

- **Lack of Authentication:** The system currently does not support password-based authentication, which limits its security and restricts its deployment in real environments.
- **Command-Line Interface:** The program runs entirely in the console, which is not user-friendly or visually appealing for end-users.
- **Scalability Concerns:** The system is designed for small-scale use and may not perform optimally when handling a large number of users or relationships.

4.1.3 Scope of Future Work

To improve and extend the current system, several future enhancements can be considered:

- **Add Password Authentication:** Implementing secure login with password verification and encryption mechanisms will improve the system's safety and realism.
- **Develop a Graphical User Interface (GUI):** A GUI using JavaFX or Swing will make the application more intuitive and user-friendly.
- **Optimize Algorithm Performance:** Modifying the BFS and DFS algorithms to handle large-scale social networks more efficiently will enhance performance and scalability.
- **Introduce User Profiles and Activity Logs:** Adding more features like user bios, profile pictures, and interaction logs could bring the simulation closer to modern social media systems.
- **Implement Real-Time Notifications:** Users can be notified of friend requests or suggestions dynamically, improving interactivity.

References

1. Oracle Java Documentation: https://docs.oracle.com/en/java/
   Official documentation for the Java programming language used in the implementation of this project.

2. GeeksforGeeks – Graph Algorithms: https://www.geeksforgeeks.org/graph-data-structu
   Resource for understanding and implementing graph traversal algorithms like BFS and DFS.

3. Java Collections Framework – TutorialsPoint: https://www.tutorialspoint.com/java/java_collections.htm
   Helpful reference for understanding Java data structures used in this project such as ArrayList, HashSet, and Queue.