Social Networking Platform

By

Vansh Srivastava (RA2311003011846)

Shubhangi Sharma (RA2311003011809)

Under the guidance of

Dr. Shiny Irene D

Assistant Professor, Department of Computing Technologies

21CSC204J – Design and Analysis of Algorithms

Faculty of Engineering and Technology

School of Computing

SRM Institute of Science and Technology Kattankulathur

March 2025

Summary

Social Networking Platform, simulates the core functionality of a basic social media system using graph-based data structures. It allows users to connect, build friendships, and explore their network using Breadth-First Search (BFS) traversal. It plays a vital role in fostering relationships, enabling communication, and promoting collaboration.

The system efficiently manages user friendships by representing the network as a graph, where users act as nodes and friendships as edges. BFS traversal is employed to explore connections level by level, making it ideal for discovering friends, suggesting connections, and identifying mutual friends. The platform also provides features for adding new users, creating friendships, and fetching common connections between two users.

Built using Flask for the backend and HTML, CSS, and JavaScript for the frontend, the project uses JSON for data storage, ensuring a lightweight and efficient system. This simulation mirrors real-world social media platforms, offering insights into how friend suggestions, network traversal, and mutual connections function.

The project demonstrates the practical application of graph traversal algorithms in social networking systems, making it a valuable learning tool for understanding data structures, algorithms, and network analysis. Furthermore, it showcases the efficiency of BFS in exploring and managing social connections, making it a scalable and adaptable model for larger platforms.

Algorithm

Social Networking Platform employs **Breadth-First Search (BFS)** as the primary algorithmic technique to traverse and explore the social graph. BFS is a graph traversal algorithm that explores nodes level by level, making it particularly effective for discovering immediate and indirect connections in a social network.

In this project:

- Users are treated as nodes in the graph.
- Friendships are represented by edges connecting the nodes.
- BFS is used to traverse the graph starting from a specified user, exploring all directly connected friends (level 1 connections) before moving to the next level (friends of friends).

Steps Required to Solve the Problem

Step 1: Initialize the graph

Step 2: Operations in the Platform:

- **Add User:** Insert a new user into the graph with an empty friends list.
- **Add Friend:** Establish a bidirectional friendship between two users.
- **Get Mutual Friends:** Find the intersection of two users' friend lists.
- **Get Friends:** Perform BFS but return only the direct friends of a specified user.

Step 2: BFS Implementation

Pseudo Code:

```
BFS(graph, start_user):
  Initialize an empty queue
  Initialize an empty set for visited nodes
  Initialize an empty list for result
  Enqueue(start_user)
  while queue is not empty:
      current = Dequeue()
      if current is not visited:
          Add current to visited
          Append current to result list
          for neighbor in graph[current]:
              if neighbor is not visited:
                  Enqueue(neighbor)
  return result
```

Step 4: BFS for Finding Friends of a Specific User

- Starts from the given user.
- Retrieves their direct friends from the graph.
- Returns the list of direct friends.

Pseudo Code

1. Add User

```
FUNCTION add_user(username)
    IF username is EMPTY:
        PRINT "Invalid username"
        RETURN
    IF username EXISTS in graph:
        PRINT "User already exists"
        RETURN
    graph[username] = []   → Create empty friends list
    SAVE graph to graph_data.json
    PRINT "User added successfully!"
END FUNCTION
```

2. Add Friend

```
FUNCTION add_friend(user1, user2)
    IF user1 == user2:
        PRINT "Cannot add yourself as a friend"
        RETURN
    IF user1 NOT in graph OR user2 NOT in graph:
        PRINT "User not found"
        RETURN
    IF user2 NOT in graph[user1]:
        APPEND user2 to graph[user1]
    IF user1 NOT in graph[user2]:
        APPEND user1 to graph[user2]
    SAVE graph to graph_data.json
    PRINT "Friendship added successfully!"
END FUNCTION
```

3. Get Mutual Friends

```
FUNCTION get_mutual_friends(user1, user2)
    IF user1 NOT in graph OR user2 NOT in graph:
        PRINT "User not found"
        RETURN
    mutual_friends = []
    FOR friend IN graph[user1]:
        IF friend IN graph[user2]:
            APPEND friend to mutual_friends
    PRINT "Mutual Friends:", mutual_friends
END FUNCTION
```

4. Get Friends

```
FUNCTION bfs_friends(username)
    IF username NOT in graph:
        PRINT "User not found"
        RETURN
    friends = []
    FOR each friend IN graph[username]:
        APPEND friend to friends
    PRINT "Direct Friends:", friends
END FUNCTION
```

Time Complexity

Let:

- V = Number of vertices (users)
- E = Number of edges (friendships)

Base case:

$$T(1) = O(1)$$

- For a single node, BFS takes O(1) time to traverse it.

Recurrence relation:

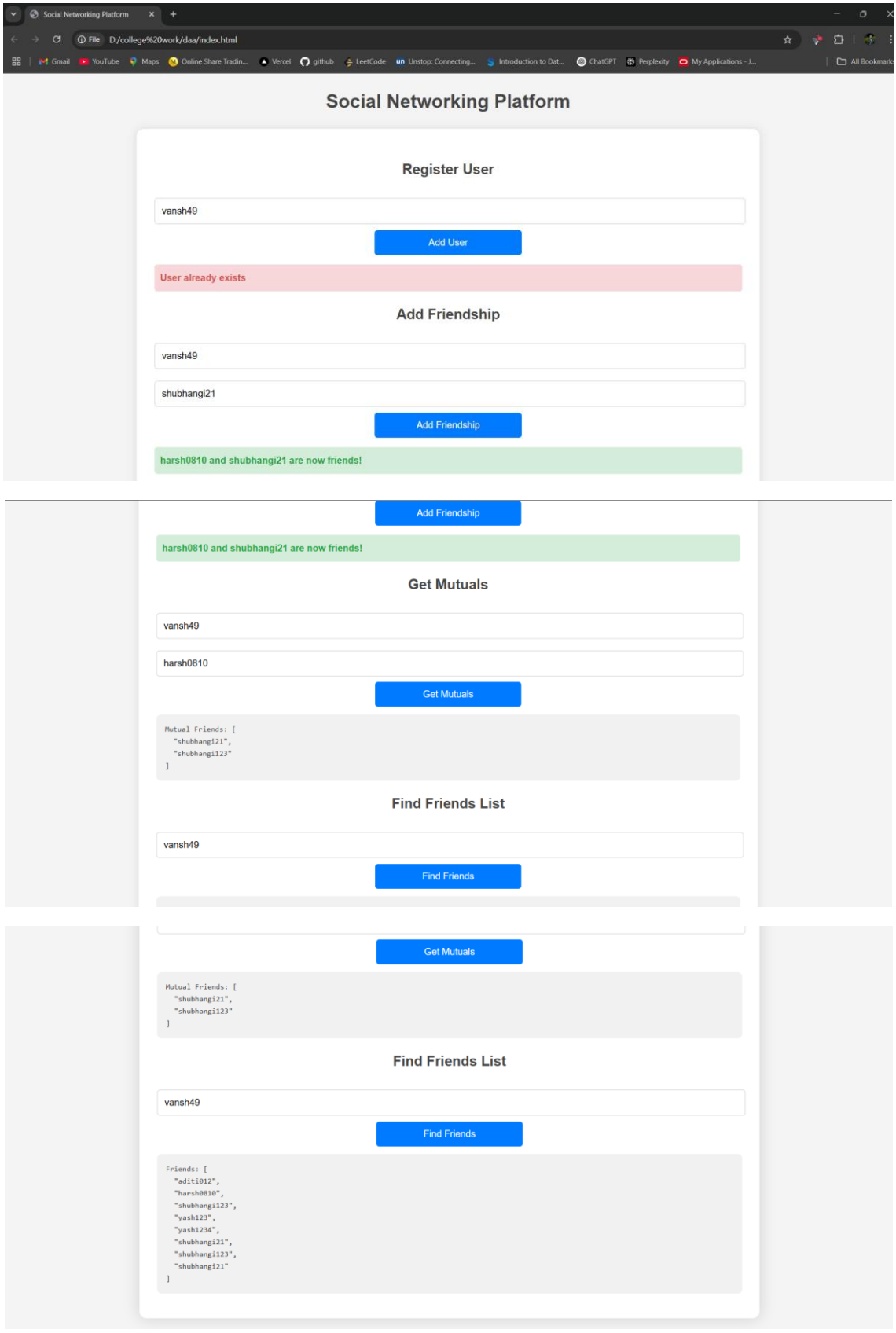
$$T(V) = T(V-1) + O(E/V)$$

- Each recursive step covers the remaining nodes $V - 1$.
- It processes E/V edges at each level (average edges per node).

Final complexity:

$$T(V, E) = O(V + E)$$

Output


Social Networking Platform
File D:/college%20work/daa/index.html
Gmail YouTube Maps Online Share Tradin... Vercel github LeetCode Unstop: Connecting... Introduction to Dat... ChatGPT Perplexity My Applications - J... All Bookmarks
Social Networking Platform
Register User
vansh49
Add User
User already exists
Add Friendship
vansh49
shubhangi21
Add Friendship
harsh0810 and shubhangi21 are now friends!
Add Friendship
harsh0810 and shubhangi21 are now friends!
Get Mutuals
vansh49
harsh0810
Get Mutuals
Mutual Friends: [
"shubhangi21",
"shubhangi123"
]
Find Friends List
vansh49
Find Friends
Get Mutuals
Mutual Friends: [
"shubhangi21",
"shubhangi123"
]
Find Friends List
vansh49
Find Friends
Friends: [
"aditi012",
"harsh0810",
"shubhangi123",
"yash123",
"yash1234",
"shubhangi21",
"shubhangi123",
"shubhangi21"
]

Conclusion

The **Social Networking Platform** project successfully demonstrates the design and implementation of a graph-based social networking platform. The system enables users to register, establish friendships, and discover mutual connections efficiently. The backend, developed using Flask (Python), provides a RESTful API to handle user data, friendship creation, and graph traversal operations. The frontend interface, built with HTML, CSS, and JavaScript, offers an intuitive and user-friendly experience, allowing seamless interaction with the platform.

By leveraging the Breadth-First Search (BFS) algorithm, the platform efficiently identifies and retrieves friend lists and mutual connections, showcasing the effectiveness of graph traversal techniques in social networking applications.

The project highlights the practical application of graph theory and web technologies in building scalable and interactive platforms. It not only meets the intended objectives but also provides a solid foundation for further enhancements and real-world deployment.