JaNET

K Koushik – B24CS1035

Arnav Vivek – B24CS1014

Aditya Pandey - B24CS1087

Ishaan Ray Ghatak - B24CS1113

Social Network Analysis System (Detailed Report)

A) SCHEMA DESIGN

1. User Schema

The User schema models every individual in the social network.
It stores:

- Basic profile information (name, username, email, year, department, intro)
- List of friends (each with a stored weight representing friendship strength)
- Friend requests received
- Friend requests sent
  This schema forms the foundation of the user graph on which most algorithms operate.

2. Chat Schema

The Chat schema stores:

- Sender ID
- Receiver ID
- Message content
- Timestamp
  This schema is essential for implementing message history retrieval, dynamic friendship weight adjustment, and future NLP-based recommendation systems (future scope for project).

3. Post Schema

The Post schema contains:

- Post creator
- Caption/content
- Tags
- Likes and interactions
  This structure is later used to compute post similarity using Jaccard score for recommendation.

B) MAJOR ALGORITHMS IMPLEMENTED

1. Friend Recommendation System (Using Jaccard Similarity)

For each user, we identify potential new connections through their friends-of-friends. Recommendations are ranked using a **Jaccard similarity score**, computed as:

$$\text{Jaccard}(A, B) = \frac{|\ Friends(A) \cap Friends(B)\ |}{|\ Friends(A) \cup Friends(B)\ |}$$

This ensures highly relevant and meaningful suggestions rather than random user choices.

---

2. Shortest Path Visualizer (Bidirectional BFS)

To compute the shortest chain of connections between any two users, we implemented a **Bidirectional BFS**, which is significantly faster than standard BFS for large graphs. The result is visualized as a chain of people (nodes) and connections (edges), showing how two users are socially linked.

---

3. Strongest Underlying Connection Network (MSF using Prim's Algorithm)

We used a **Minimum Spanning Forest** built using a modified version of **Prim's algorithm** on disconnected social graphs.
Each tree in the forest represents a **strongest underlying connection backbone** within a community.

A dynamic visualizer was implemented using **Vis.js**, displaying:

- Nodes → Users
- Edges → Friendships
- Edge lengths & labels proportional to friendship weights

This acts like a "social network skeleton," revealing the fundamental structure of the graph. It helps to cleanly visualise the graph without too many unnecessary edges.

---

4. Dynamic Friendship Calculator

Friendship is modelled using edge weights.
Each time two users chat, the weight of their edge is decreased, meaning their social

closeness increases.
Over time, this builds a dynamic model of how relationships evolve.

---

5. Post Recommendation System (Jaccard Similarity on Tags)

Posts are recommended based on the overlap between:

- Tags the user engages with
- Tags associated with available posts

The Jaccard similarity ensures that posts aligned with a user's interests appear first.

C) OTHER IMPORTANT ALGORITHMIC IMPLEMENTATIONS

1. User Management Algorithms (users.js)

We implemented complete CRUD-like functionality:

- Add new user
- Add group of users (helper function for testing purposes)
- Get all users
- Get user by ID
- Delete all users (testing cleanup)

These endpoints are essential for populating and maintaining the graph.

---

2. Friendship System (friends.js)

We implemented:

- Send friend request
- Accept request
- Reject request
- Send mass requests (testing only)

These operations maintain bidirectional edges in the graph and update request lists safely.

---

3. Chat System (chats.js)

We implemented:

- Send a message
- Retrieve full chat history between two users

This directly integrates with the dynamic weight update algorithm for friendships.

---

4. Community Detection (community.js)

Using **Depth-First Search**, we detect communities (connected components) in the graph.
This feature allows analysis of structural clustering and social sub-groups within the network.

---

5. Visualization Algorithms (visualisers.js)

Two major visualizers were implemented:

- **MSF Visualizer** (Prim + Vis.js)
- **Shortest Path Visualizer** (Bidirectional BFS)

These tools help users see the structure of the network and better understand their social environment.