System Requirements Specification Index

For

Social Network Graph Analysis System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Social Network Graph Analysis
System Requirements Specification

1 PROJECT ABSTRACT

NetConnect Analytics requires a specialized set operations system to analyze social network connections, community structures, and information flow patterns. The system will identify friend circles, analyze mutual connection networks, detect community clusters, and track information propagation through social graphs. This tool will enable researchers to identify influential nodes, understand connection patterns, and predict information spread through complex social networks.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. System needs to store and analyze different types of network data (users, connections, communities, interaction metrics) 2. System must support filtering connections by relationship type, interaction frequency, or community membership 3. Console should handle different set operations like Basic set operations (friend lists, mutual connections), Set comprehensions (for filtering specific user patterns), Set methods (add, remove, intersection, union), Set relationship tests (analyzing network overlaps), Set-based network metrics calculation

3 CONSTRAINTS

3.1 INPUT REQUIREMENTS

1. User Sets:

- Must be stored as sets of strings representing user IDs
- Example: `network_users = {"user1", "user2", "user3", "user4", "user5"}`

2. Connection Sets:
   - Must be stored as sets of connections between users
   - Example: `user1_connections = {"user2", "user3", "user4"}`
3. Community Sets:
   - Must be stored as sets representing community memberships
   - Example: `tech_community = {"user1", "user3", "user5", "user7"}`
4. Predefined Network Sets:
   - Must use these exact predefined sets:
     - `network_a = {"user1", "user2", "user3", "user4", "user5", "user6", "user7"}`
     - `network_b = {"user5", "user6", "user7", "user8", "user9", "user10"}`
     - `tech_group = {"user1", "user3", "user5", "user8", "user10"}`
     - `gaming_group = {"user2", "user4", "user6", "user8", "user9"}`
     - `arts_group = {"user3", "user5", "user7", "user10"}`
5. Connection Maps:
   - Must use these exact connection mappings:
     - `user1_connections = {"user2", "user3", "user5"}`
     - `user2_connections = {"user1", "user4", "user6"}`
     - `user3_connections = {"user1", "user5", "user7"}`
     - `user4_connections = {"user2", "user6"}`
     - `user5_connections = {"user1", "user3", "user7", "user8"}`
6. New Network Sets:
   - Must use these exact predefined items:
     - `new_users = {"user11", "user12", "user13"}`
     - `influencers = {"user3", "user5", "user8", "user11"}`

3.2 OPERATIONS CONSTRAINTS

1. Set Creation:
   - Must use proper set creation syntax

- Example: `{"user1", "user2", "user3"}` or `set(["user1", "user2", "user3"])`

2. Set Access:
   - Must use proper set membership testing
   - Example: `if user in user_set:` or `for user in user_set:`
3. Set Operations:
   - Must use standard set operations
   - Example: Union: `A | B` or `A.union(B)`, Intersection: `A & B` or `A.intersection(B)`
4. Set Comprehensions:
   - Must use set comprehensions where appropriate
   - Example: `{user for user in network_a if user in tech_group}`
5. Set Methods:
   - Must use appropriate set methods
   - Example: `user_set.add(new_user)`, `user_set.remove(user)`, `user_set.update(new_users)`
6. Set Relationships:
   - Must test set relationships correctly
   - Example: `network_a.issuperset(tech_group)`, `user1_connections.isdisjoint(user4_connections)`
7. Set-Based Analytics:
   - Must use sets for network analysis
   - Example: `len(user1_connections & user2_connections)` to count mutual friends
8. Error Handling:
   - Must handle set operation errors appropriately
   - Example: Using `.discard()` instead of `.remove()` to avoid KeyError
9. Immutability:
   - Must respect set operation principles
   - Example: Creating new sets for results rather than modifying inputs
10. Set Theory Applications:
    - Must apply set theory to network problems
    - Example: Using symmetric difference to find exclusive connections

3.3 OUTPUT CONSTRAINTS

1. Display Format:
    - Show network analysis results with clear formatting
    - Format large user sets with appropriate separators
    - Each analysis result must be displayed on a new line
2. Output Format:
    - Must show in this order:
        - Show "== SOCIAL NETWORK GRAPH ANALYSIS =="
        - Show "Total Users: {count}"
        - Show "Available Networks: {networks}"
        - Show "Current Network Data:"
        - Show sets with format: "{network_name}: {users}"
        - Show "Analysis Result:" when displaying operation results

4. TEMPLATE CODE STRUCTURE:

1. Data Management Functions:

- `initialize_data()` - creates the initial network sets and connection data

2. Dictionary Operation Functions:

- `find_mutual_connections(user_a_connections, user_b_connections)` - finds mutual connections between users
- `find_exclusive_connections(user_a_connections, user_b_connections)` - finds connections unique to each user
- `find_all_connections(connections_list)` - merges all connections from a list
- `find_common_groups(user, group_dict)` - finds all groups a user belongs to
- `is_direct_connection(user_a, user_b, connections_dict)` - checks if users are directly connected
- `is_second_degree_connection(user_a, user_b, connections_dict)` - checks for connections through a mutual friend
- `calculate_connection_strength(user_a, user_b, interaction_data)` - calculates connection strength
- `find_community_overlaps(group_a, group_b)` - finds users in multiple communities
- `identify_bridge_users(communities_list)` - identifies users that connect multiple communities
- `calculate_network_density(users, connections_dict)` - calculates network density
- `find_isolated_users(users, connections_dict)` - finds users with no connections
- `recommend_connections(user, all_users, connections_dict)` - recommends new connections

3. Display Functions:

- `format_users_for_display(group_name, users)` - formats users for display
- `display_analysis_result(operation, set_a, set_b, result)` - displays analysis results
- `display_data(data, data_type)` - displays sets or other data types

4. Program Control Functions:

- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:
   - Option 1: View Network Data
   - Option 2: Analyze Connections
   - Option 3: Find Community Patterns
   - Option 4: Calculate Network Metrics
   - Option 5: Generate Recommendations
   - Option 0: Exit
4. Perform operations on the network data
5. View results after each operation
6. Exit program when finished