Social Media Network Analyzer: Project Documentation

Executive Summary

The Social Media Network Analyzer is a full-stack system for analyzing and visualizing social media graphs. It integrates an efficient C++ backend implementing advanced graph algorithms, a highly interactive web frontend, and a Python-based dataset generator. The system enables exploration of synthetic social networks, analyzing community structures, ranking influencers, recommending friendships, and tracking network evolution over time. This document covers system architecture, major algorithms, implementation details, API specification, project setup, and future directions, providing a comprehensive technical reference for developers and researchers.

Project Overview

The project features:

- **Backend:** C++ HTTP server with high-performance graph algorithms and a RESTful API.
- **Frontend:** Web interface for real-time network visualization and interactive exploration, leveraging vis-network.
- **Dataset Generator:** Python module for creating synthetic user graphs with temporal snapshots and regional clustering.

Key Features

- Graph visualization and exploration
- Interactive community detection (Label Propagation, Greedy Modularity)
- PageRank-based influencer ranking
- Friend recommendation based on multiple real-world factors
- Shortest path and mutual friends analysis
- Trie-based user search and network statistics
- Temporal support with multiple date-based datasets

System Architecture

The architecture consists of:

- **Frontend Layer**: Built on HTML/CSS/JavaScript, providing search, feature controls, graph canvas, and modals for analysis results.
- **Backend Layer**: C++ API server with multithreaded request handling and graph caching per dataset date. Key classes include `SocialGraph`, `SimpleHTTPServer`, and unified interface via `GraphAlgorithms`.

- **Dataset Layer**: Python scripts generate initial graphs, apply temporal evolution, and export data as JSON/CSV for backend ingestion.

Backend Implementation

Core Graph Data Structures

- **Node** stores user data, relationship sets (friends, followers, following), and features such as location and interests.
- **Edge** captures relationships with additional metadata (type, distance, interaction).
- **SocialGraph** manages all nodes and edges, optimized for O(1)/O(n) lookup and batch updates.

Unified Algorithm Interface (`Algorithm.hpp`)

- Aggregates all major algorithm modules: community detection, PageRank, influencer ranker, friend recommender, path calculator, mutual friends analyzer, centrality analyzer, trie-based search, friendship score calculator.

HTTP API Server (`apiserver.cpp`)

- RESTful API with endpoints for core features and cross-platform socket implementation.
- Supports date-based caching for temporal network snapshots and multi-threaded request routing.

Major Algorithms

Community Detection

- **Label Propagation**: Iteratively propagates dominant labels in the network for fast clustering.
- **Greedy Modularity**: Agglomerative optimization of modularity for dense community identification.

PageRank & Influencer Ranking

- **PageRank**: Evaluates node importance using iterative formula with configurable damping.
- **Influencer Ranker**: Combines normalized followers, fans, and PageRank score for multi-factor influencer assessment.

Friend Recommendation

- Factors: mutual friends, interests overlap, geographic distance, community similarity.
- Provides ranked suggestions with detailed reasoning for each recommendation.

Shortest Path

- **Bidirectional BFS** for efficient path finding and batch queries.

Mutual Friends & Centrality Metrics

- Fast set intersection for mutual friends, Jaccard similarity calculation.
- Calculation of degree, closeness, and clustering coefficients for user and global network analysis.

Trie-Based User Search

- Prefix-based fast lookups using Trie for real-time user filtering and autocomplete.

Friendship Score

- Multi-factor score combining direct friendship status, path distance, mutuals, interests, and proximity, with human-readable explanations.

Frontend Functionality

File Structure

- `index.html`: Layout organizes top toolbar, search bar, graph canvas, node details, feature panels, and modals.
- `main.js`: Core logic for graph rendering, event handling, and state management.
- `api.js`: REST API client handling backend interaction and data fetching.
- `features.js`: Feature controls, analytics, display logic for results and highlights.
- `search.js`: Debounced autocomplete and prefix-based node search.
- `utils.js`, `styles.css`: Utility logic and aesthetic styling (color scheme, animations, responsive design).

Interactive Features

- Search users, explore nodes and edges, view influencer leaderboards, detect communities, analyze paths and recommendations, manage dates and visualization states.
- Real-time highlighting and dynamic legend updates, user-centered UX.

Dataset Generation and Temporal Analysis

- **Python module** creates **synthetic graphs**, with interest and region clustering, managed friend/fan relationships, daily evolution of connections, message counts, and support for viral user dynamics.
- Flexible configuration for node count, regions, evolution days, connection probabilities, and export formats.

RESTful API Reference

- **GET /api/dates:** List available dataset dates
- **GET /api/graph?date=YYYY-MM-DD:** Retrieve graph snapshot for date
- **GET /api/node/{id}:** Detailed user node info
- **GET /api/search?q=prefix:** Fast user search for autocomplete
- **POST /api/mutual-friends:** Analyze mutual friends between two users
- **GET /api/influencer-leaderboard:** Retrieve influencer ranking
- **GET /api/communities:** Identify network communities
- **GET /api/path:** Find shortest path between users, friendship score
- **GET /api/recommendations:** Friend recommendations with scores and explanations

Setup, Installation, and Usage

Prerequisites

- **Backend:** C++17 compiler (GCC/Clang/MSVC), nlohmann::json header
- **Frontend:** Modern browser, local HTTP server optional
- **Dataset Generation:** Python 3.7+, NetworkX

Installation

- Backend: Compile and launch the C++ server with dataset files
- Frontend: Open `index.html` directly or via Python HTTP server
- Dataset: Setup dependencies, configure parameters, run generator, export JSON/CSV files

Usage Guide

- Generate dataset with Python script
- Launch backend server and frontend interface
- Explore graph interactively: search, highlight, analyze features
- Use feature buttons for advanced analytics (mutual friends, paths, recommendations)
- Change dates for temporal analysis

Results, Statistics, and Future Enhancements

- ~5000 lines of code across backend, frontend, and dataset modules
- Covers 9 advanced graph algorithms, 9 REST endpoints
- Quantitative statistics and UX analytics available via feature panels

Planned Improvements

- Real-time updates (WebSocket integration)
- Graph export, advanced filtering, saved searches, historical trends, optimized performance for large graphs
- User authentication and personalized analytics dashboard

Conclusion

The Social Media Network Analyzer demonstrates state-of-the-art graph algorithms, scalable backend engineering, and interactive frontend visualization. Its modular architecture supports maintainability and extensibility for future research and product enhancements. This documentation provides a rigorous technical overview for users and contributors.