Social Network Intelligence Engine — Final Project Draft

Project Name: Social Network Intelligence Engine (SNIE)

Version: Final Draft

Date: November 2025

1. Project Overview

The Social Network Intelligence Engine (SNIE) is an analytical system designed to understand, interpret, and optimize connectivity within a social or professional network. It models entities (individuals, businesses, researchers) as nodes and their relationships as weighted edges, allowing the system to detect communities, map influence, recommend bridges, and optimize global network structure.

SNIE integrates graph theory, relationship semantics, and domain expertise mapping into a unified hybrid model. It leverages both offline structural intelligence (Betweenness Centrality) and real-time local analytics (Weighted Dijkstra + Semantic Scoring) to generate context-aware recommendations.

2. Goals & Objectives

- Detect natural communities and isolated clusters.

- Identify influential individuals using graph centrality measures.

- Recommend local and cross-community connections.

- Suggest global bridges between two distinct networks for maximum connectivity improvement.

- Use semantic edge weights to represent the type and strength of relationships.

- Visualize the network, communities, and recommended bridges.

3. Domain & Tie Semantics

Node Attributes

Each node represents a user or entity and carries semantic information:

id: Unique integer identifier

domain: Area of work or specialization

strength: Skill/expertise level (0–10 scale)

Domain Mapping

1 = Finance

2 = Health

3 = AI / ML

4 = Academia

Edge Weight Mapping

1 = LinkedIn / Acquaintance (Weak tie)

2 = Colleague / Business Partner (Moderate tie)

3 = Friend (Strong tie)

4 = Close Friend / Confidant (Very strong tie)

Weighted edges are treated inversely for distance calculations: Effective Distance = 1 / Weight

4. High-Level Architecture

1. Offline Structural Analysis: Detect communities, compute Betweenness Centrality, and cache normalized scores.

2. Online Recommendation: Run Weighted Dijkstra for the queried node and generate recommendations using semantic scoring.

3. Global Optimization: Suggest bridges between networks using BC × BC heuristic.

4. Visualization: Show colored communities, node influence, and highlighted bridge recommendations.

5. Algorithms & Complexity

Community Detection — DFS / Union-Find — O(V + E)

Shortest Path — Weighted Dijkstra — O((V + E) log V)

Betweenness Centrality — Brandes Algorithm — O(V × E)

Bridge Pairing — BC × BC Heuristic — O(k^2)

6. Scoring Formulas

Local Recommendation Score = α*DomainSim + β*Proximity + γ*StrengthFactor + δ*NormalizedBC

Bridge Recommendation Score = α'*(1 - DomainSim) + β'*Proximity + γ'*StrengthBalance + δ'*NormalizedBC

Global Bridge Score = BC_A(i) × BC_B(j) × DomainComplementarity × StrengthBalance

7. Data Model & I/O

Nodes CSV: id,domain(1-4),strength(0-10)

Edges CSV: u,v,weight(1-4)

Exports: communities.json, centrality.csv, recommendations.json, bridges.json

8. Offline–Online Pipeline

Offline: Compute communities + BC (cached).

Online: Dijkstra + scoring (real-time recommendations).

Visualization: Render using NetworkX / Gephi.

Global Optimization: Cross-network BC pairing.

9. Visualization Framework

Nodes: color by domain, size by centrality.

Edges: thickness by weight, dashed lines for recommended bridges.

Communities: color-highlighted clusters.

10. Example Run

Input:

nodes.csv → id,domain,strength

0,3,8.5

1,3,7.0

2,1,6.5

3,4,9.0

4,2,8.0

edges.csv → 0,1,4 | 1,2,2 | 3,4,3

Output:

Local Recommendations for Node 0:

1) Node 2 | Domain 1 | Strength 6.5 | Score 0.78

2) Node 4 | Domain 2 | Strength 8.0 | Score 0.62

3) Node 3 | Domain 4 | Strength 9.0 | Score 0.59

Bridge Recommendations:

1) Node 3 | Domain 4 | Strength 9.0 | BridgeScore 0.84

2) Node 4 | Domain 2 | Strength 8.0 | BridgeScore 0.72