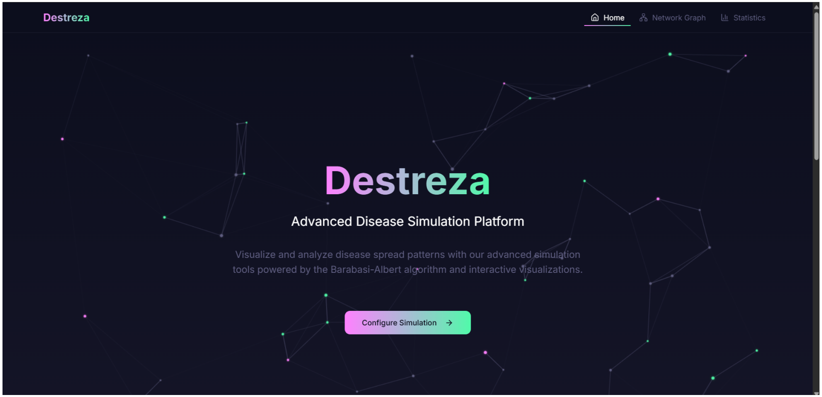
Destreza

Advanced Disease Simulation Platform

Ahmad Shahmeer

May 2025



Destreza

Advanced Disease Simulation Platform

Visualize and analyze disease spread patterns with our advanced simulation tools powered by the Barabasi-Albert algorithm and interactive visualizations.

Outline

Models and Libraries

- **Languages:** TypeScript & React (Next.js “use client”)
- **Visualization:** D3.js (SVG) & HTML5 Canvas API
- **Compute:** Web Workers (force layout), GPU.js (optional kernel)
- **UI Components:** shadcn/ui, lucide–react, Tailwind CSS

Note:
Shifted to this stack instead of original proposal because:

- Web Workers offload compute; GPU.js scales to 1,000+ nodes via GPU acceleration.
- D3.js with SVG/Canvas outshines Plotly for large networks.
- Next.js enables cross-device use without installs.
- TypeScript optimized custom algorithms (e.g., Fenwick Tree).

SEIRD Model Overview

We simulate disease with five compartments:

S (Susceptible) Uninfected individuals at risk.

E (Exposed) Infected but *not yet* infectious (incubation).

I (Infectious) Can transmit disease to neighbors.

R (Recovered) No longer infectious, immune (temporary).

D (Deceased) Removed by mortality.

Transitions per day:

$$S \xrightarrow{\beta} E, \quad E \xrightarrow{\sigma = \frac{1}{\text{incubation}}} I, \quad I \xrightarrow{\gamma} R, \quad I \xrightarrow{\mu} D, \quad R \xrightarrow{\rho = 0.01} S$$

Key Parameters

- **Population Size:** total number of individuals in the simulated population (nodes 100–2500)
- **Initial Infected:** number of infected individuals at the start of the simulation (scaled with population)
- **Basic Reproduction Number R_0:** average number of secondary infections produced by a single infected individual in a completely susceptible population. Higher values indicate more contagious diseases (0.1–10)
- **Recovery Rate:** probability that an infected individual will recover during each time step. Higher values indicate faster recovery $P[\mathrm{I} \rightarrow \mathrm{R}]$ (0.01–0.5)
- **Infectious Mortality Rate:** base probability that an infectious individual will die from the disease. This is modified by other factors like vaccination rate and social distancing $P[\mathrm{I} \rightarrow \mathrm{D}]$ (0.001–0.1)

Advanced Parameters

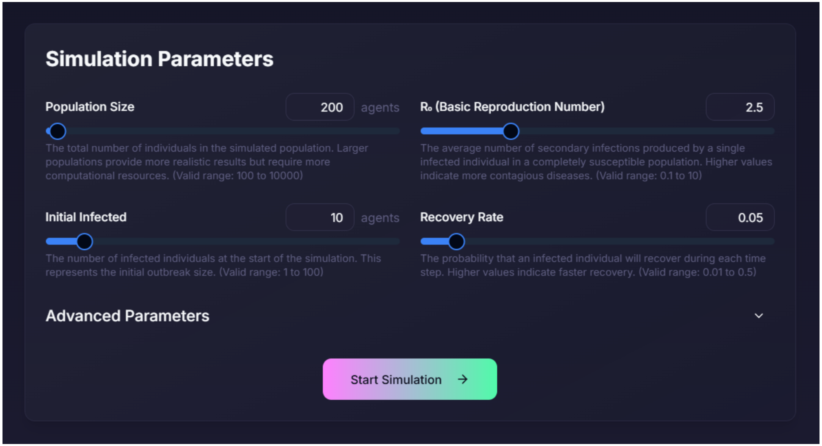
- **Incubation Period:** days before $E \rightarrow I$ (1–30)
- **Infectious Period:** duration during which an infected individual can transmit the disease to others. Longer periods increase the potential for spread (1–30)
- **Vaccination Rate (ν):** proportion of the population that is vaccinated against the disease. Higher vaccination rates provide greater herd immunity (0–1)
- **Social Distancing (α):** degree to which individuals reduce their social contacts. Higher values reduce disease transmission (0–1)

Population Mortality Rate

We compute a single "population mortality rate" (PMR) as:

$$\text{PMR} = \underbrace{\left(1 - \frac{1}{R_0(1-\alpha)}\right)}_{\text{Attack Rate}} \times \underbrace{\text{IFR}\,(1 - \nu\,\epsilon)}_{\text{Adjusted Fatality}}$$

R_0	Basic reproduction number
α	Social distancing factor
IFR	Infection fatality rate (per-case mortality)
ν	Vaccination rate
ϵ	Vaccine efficacy (we use 0.80)



Simulation Parameters

Population Size 200 agents

The total number of individuals in the simulated population. Larger populations provide more realistic results but require more computational resources. (Valid range: 100 to 10000)

R_0 (Basic Reproduction Number) 2.5

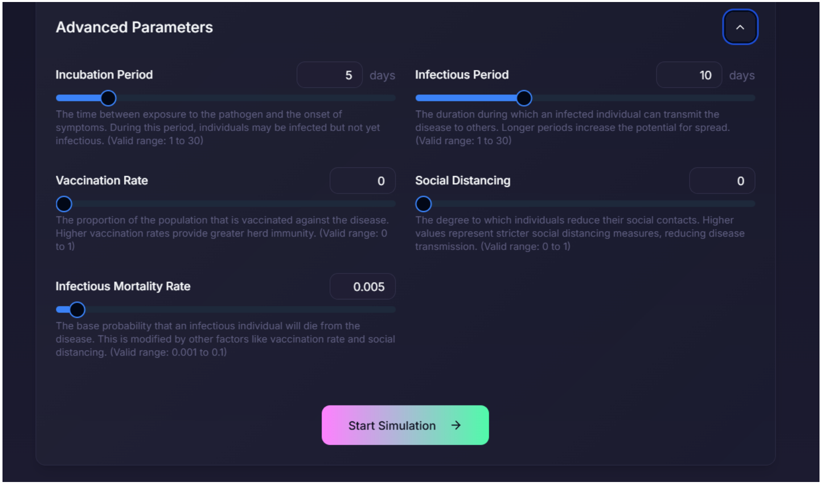
The average number of secondary infections produced by a single infected individual in a completely susceptible population. Higher values indicate more contagious diseases. (Valid range: 0.1 to 10)

Initial Infected 10 agents

The number of infected individuals at the start of the simulation. This represents the initial outbreak size. (Valid range: 1 to 100)

Recovery Rate 0.05

The probability that an infected individual will recover during each time step. Higher values indicate faster recovery. (Valid range: 0.01 to 0.5)


Advanced Parameters

Incubation Period 5 days

The time between exposure to the pathogen and the onset of symptoms. During this period, individuals may be infected but not yet infectious. (Valid range: 1 to 30)

Infectious Period 10 days

The duration during which an infected individual can transmit the disease to others. Longer periods increase the potential for spread. (Valid range: 1 to 30)

Vaccination Rate 0

The proportion of the population that is vaccinated against the disease. Higher vaccination rates provide greater herd immunity. (Valid range: 0 to 1)

Social Distancing 0

The degree to which individuals reduce their social contacts. Higher values represent stricter social distancing measures, reducing disease transmission. (Valid range: 0 to 1)

Infectious Mortality Rate 0.005

The base probability that an infectious individual will die from the disease. This is modified by other factors like vaccination rate and social distancing. (Valid range: 0.001 to 0.1)

Network Generation

- Custom Barabási–Albert (BA) implementation:
  1. Start with m_0 fully connected nodes.
  2. For each new node $i = m_0, \dots, N-1$:
     - compute `total` = `fenw.query`(N-1)
     - repeat m times:
     - draw $r \leftarrow$ random() $\times$ total
     - $t \leftarrow$ `fenw.sample`(r)
     - connect(i, t); `fenw.update`$(i, 1)$; `fenw.update`$(t, 1)$
- Uses Fenwick tree sampling in $O(\log N)$ per edge ($O(N \log N)$ overall)

Network Generation: Pseudocode

Barabási–Albert with Fenwick Sampling

```
for i = m0 to N-1:
  total = fenw.query(N-1)
  for k = 1 to m:
    r = random() * total
    t = fenw.sample(r)
    connect(i, t)
    fenw.update(i, 1)
    fenw.update(t, 1)
  end for
end for
```

Fenwick Tree Optimization (Part 1)

- **What is a Fenwick Tree?** A data structure for efficient prefix sum queries and updates, both in $O(\log N)$ time.
- **Usage in Network Generation:**
  - Maintains cumulative sum of node degrees in an array `f[1..N]`.
  - Enables sampling of nodes proportional to their degrees in $O(\log N)$ per operation.
  - Replaces $O(N)$ linear sampling, reducing total complexity from $O(N^2)$ to $O(N \log N)$.

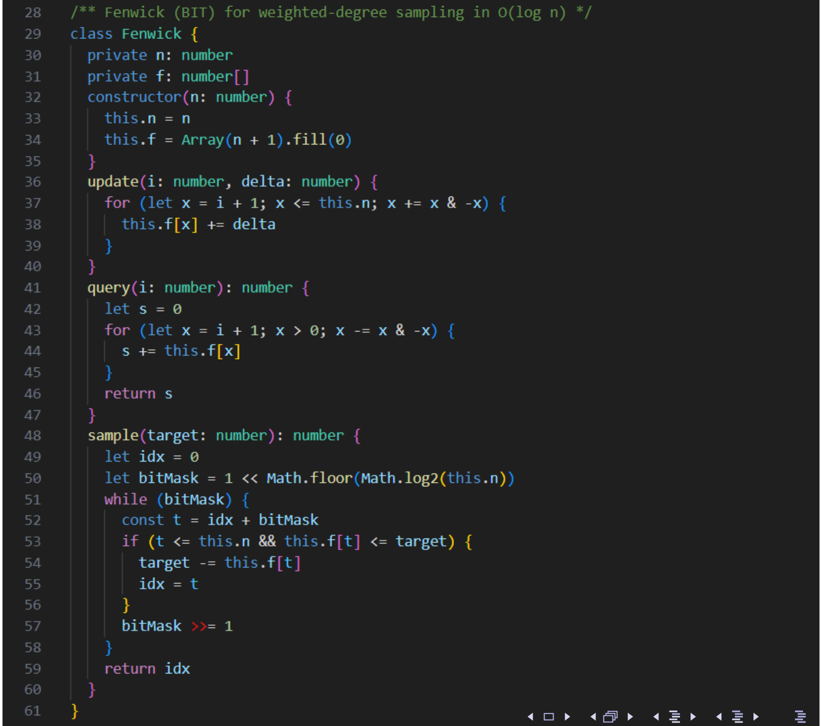
Fenwick Tree Optimization (Part 2)

- `update(idx, delta)`: Adds `delta` to node `idx`'s degree and updates the tree in $O(\log N)$.
- `query(i)`: Computes sum of degrees up to index `i` in $O(\log N)$.
- `sample(target)`: Finds node with cumulative degree sum $\geq$ `target` via binary search-like walk in $O(\log N)$.
- Ensures preferential attachment: nodes with higher degrees are more likely to gain edges.

Fenwick Tree Optimization

Pseudocode:

```
Initialize Fenwick tree fenw with size N
for i from 0 to m0-1:
    Add node i to network
    fenw.update(i, m0 - 1)
    for j from 0 to i-1:
        Add edge (i, j)
for i from m0 to N-1:
    Add node i to network
    total = fenw.query(N-1)
    targets = empty set
    while |targets| < m:
        r = random() * total
        t = fenw.sample(r)
        if t != i and t not in targets:
            Add t to targets
    for t in targets:
        Add edge (i, t)
        fenw.update(i, 1)
        fenw.update(t, 1)
```



```typescript
/** Fenwick (BIT) for weighted-degree sampling in O(log n) */
class Fenwick {
  private n: number
  private f: number[]
  constructor(n: number) {
    this.n = n
    this.f = Array(n + 1).fill(0)
  }
  update(i: number, delta: number) {
    for (let x = i + 1; x <= this.n; x += x & -x) {
      this.f[x] += delta
    }
  }
  query(i: number): number {
    let s = 0
    for (let x = i + 1; x > 0; x -= x & -x) {
      s += this.f[x]
    }
    return s
  }
  sample(target: number): number {
    let idx = 0
    let bitMask = 1 << Math.floor(Math.log2(this.n))
    while (bitMask) {
      const t = idx + bitMask
      if (t <= this.n && this.f[t] <= target) {
        target -= this.f[t]
        idx = t
      }
      bitMask >>= 1
    }
    return idx
  }
}
```

Force-Directed Layout

- D3.js forces (in Web Worker):

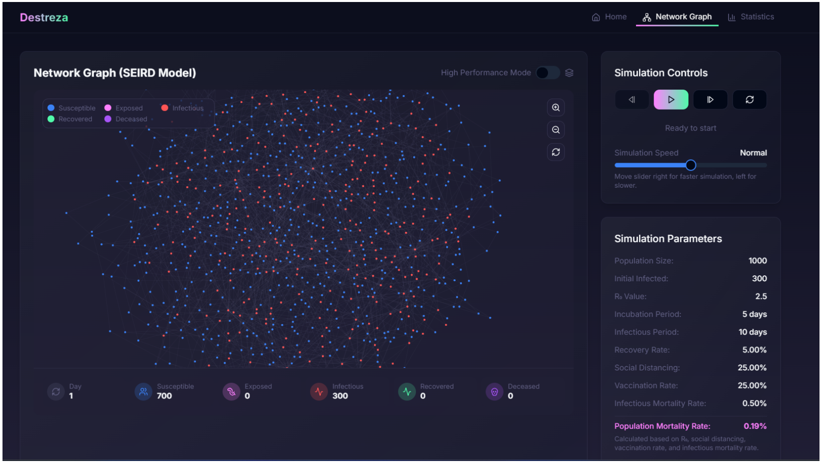
link	spring to connect edges
charge	repulsion between nodes
center	attract to canvas center
collision	prevent overlap

- Render via SVG *or* Canvas API (20 fps throttled)
- Optional GPU.js kernel for all-pairs repulsion

Force-Directed Graph: Pseudocode

Force Simulation Loop

```
function forceSimulation(nodes, links):
    Initialize simulation with forces:
        link = spring force on edges
        charge = repulsion between nodes
        center = attraction to canvas center
        collision = prevent node overlap
    alpha = 1  // Initial energy
    while alpha >= threshold:
        for each node:
            netForce = lF + chF + ceForce + colForce
        for each node:
            velocity += netForce * alpha
            position += velocity
            Apply constraints (e.g., collision boundaries)
        alpha *= (1 - alphaDecay)  // Cool down simulation
    return final node positions
```



Simulation Speed: Normal

Move slider right for faster simulation, left for slower.

Day	1
Susceptible	700
Exposed	0
Infectious	300
Recovered	0
Deceased	0

Simulation Parameters

Population Size:	1000
Initial Infected:	300
R_0 Value:	2.5
Incubation Period:	5 days
Infectious Period:	10 days
Recovery Rate:	5.00%
Social Distancing:	25.00%
Vaccination Rate:	25.00%
Infectious Mortality Rate:	0.50%
Population Mortality Rate:	0.19%

Calculated based on R_0, social distancing, vaccination rate, and infectious mortality rate.

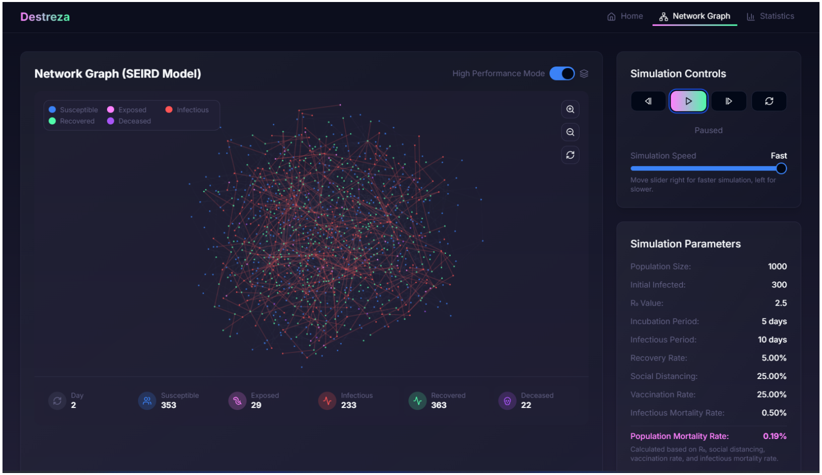
SEIRD Simulation Step

1. For each infectious node i:
   - For each neighbor j in adjacency list:
     - if j susceptible: infect w.p. p_{inf} (adjusted by vax/distancing)
   - $i \to D$ w.p. PMR or $i \to R$ w.p. recoveryRate
2. Exposed $\to$ Infectious w.p. 1/incubation
3. Recovered $\to$ Susceptible w.p. 0.01 (SIRS)
4. Update stats + history

SEIRD Simulation Step

Pseudocode:

```
function simulationStep(nodes, links, params):
    newDay = currentDay + 1
    for each node i:
        if nodes[i].status == "infectious":
            for each neighbor j in adj[i]:
                if nodes[j].status == "susceptible":
                    p_infect = params.infectionRate *
                               (1 - params.socialDistancing) *
                               (1 - params.vaccinationRate * 0.8)
                    if random() < p_infect:
                        nodes[j].status = "exposed"
                        nodes[j].day = newDay
                        Add link (i, j) with day = newDay, active = true
            if random() < params.mortalityRate:
                nodes[i].status = "deceased"
                nodes[i].day = newDay
                Deactivate all links connected to i
            elif random() < params.recoveryRate:
                nodes[i].status = "recovered"
                nodes[i].day = newDay
        elif nodes[i].status == "exposed":
            if random() < params.exposedRate:
                nodes[i].status = "infectious"
                nodes[i].day = newDay
        elif nodes[i].status == "recovered":
            if random() < 0.01:  // SIRS model
                nodes[i].status = "susceptible"
                nodes[i].day = newDay
    Update statistics (S, E, I, R, D counts)
    Save state to history
    return nodes, links
```



Destreza

Home | Network Graph | Statistics

Network Graph (SEIRD Model)

High Performance Mode

● Susceptible ● Exposed ● Infectious
● Recovered ● Deceased

Day	Susceptible	Exposed	Infectious	Recovered	Deceased
2	353	29	233	363	22

Simulation Controls

Paused

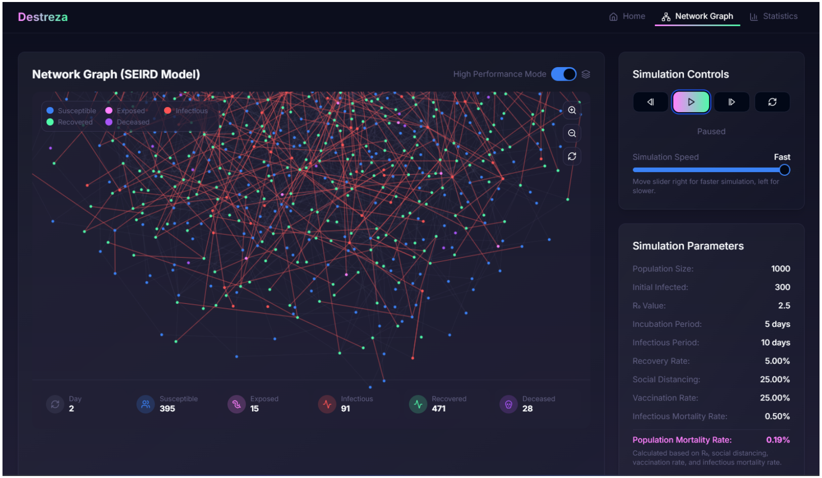
Simulation Speed Fast

Move slider right for faster simulation, left for slower.

Simulation Parameters

Population Size:	1000
Initial Infected:	300
R_0 Value:	2.5
Incubation Period:	5 days
Infectious Period:	10 days
Recovery Rate:	5.00%
Social Distancing:	25.00%
Vaccination Rate:	25.00%
Infectious Mortality Rate:	0.50%
Population Mortality Rate:	0.19%

Calculated based on R_0, social distancing, vaccination rate, and infectious mortality rate.

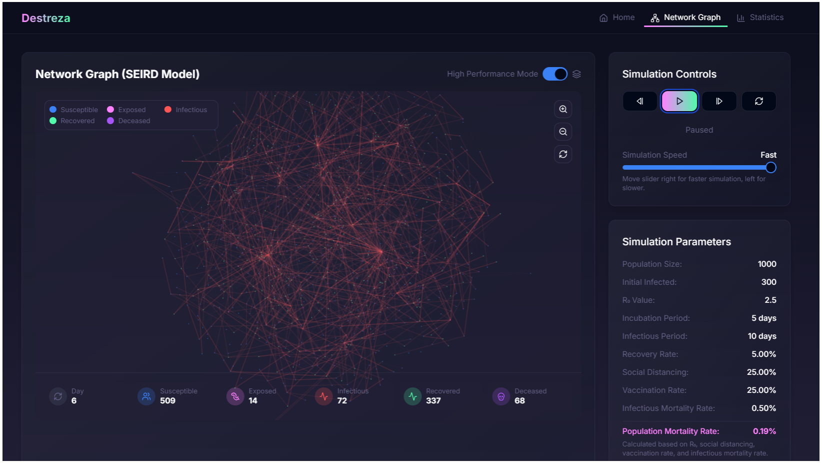


Day 2
Susceptible 395
Exposed 15
Infectious 91
Recovered 471
Deceased 28

Simulation Parameters

Population Size:	1000
Initial Infected:	300
R_0 Value:	2.5
Incubation Period:	5 days
Infectious Period:	10 days
Recovery Rate:	5.00%
Social Distancing:	25.00%
Vaccination Rate:	25.00%
Infectious Mortality Rate:	0.50%
Population Mortality Rate:	0.19%

Calculated based on R_0, social distancing, vaccination rate, and infectious mortality rate.



Network Graph (SEIRD Model)

● Susceptible ● Exposed ● Infectious
● Recovered ● Deceased

Day	Susceptible	Exposed	Infectious	Recovered	Deceased
6	509	14	72	337	68

Simulation Controls

Simulation Speed: Fast

Move slider right for faster simulation, left for slower.

Simulation Parameters

Population Size:	1000
Initial Infected:	300
R_0 Value:	2.5
Incubation Period:	5 days
Infectious Period:	10 days
Recovery Rate:	5.00%
Social Distancing:	25.00%
Vaccination Rate:	25.00%
Infectious Mortality Rate:	0.50%
Population Mortality Rate:	0.19%

Calculated based on R_0, social distancing, vaccination rate, and infectious mortality rate.

Observations

- Deceased nodes connections are disabled during simulation
- Low populations of 200 or less, with few infected individuals, result in a very short simulation with all nodes eventually healthy, even without any preventive measures like vaccinations or social distancing
- But for networks as large as 500 nodes, due to increased number of connections, disease spread is faster and number of infectious and deceased is noticeably higher
- PMR formula accurately predicts final death toll in benchmarks
- Colored transmission links reveal that infection spreads tend to have "epicenters" or "super-spreaders" and these tend to be near the center of the network
- Many other interesting observations result from manipulatng different values of the parameters

Potential Improvements

- **Network Generation:**
  - Parallelize Barabási–Albert model using Web Workers or GPU.js to speed up generation for 100k+ nodes.
- **Force-Directed Layout:**
  - Could have Barnes-Hut approximation for repulsion forces, reducing complexity from $O(N^2)$ to $O(N \log N)$.
- **Rendering:**
  - Implement level-of-detail (LOD) rendering: render clusters as single points when zoomed out.
- **Data Structures:**
  - Switch to sparse matrices or compressed sparse row (CSR) formats for adjacency data to save memory.

Thank You

Questions?