ResQ-Graph: Autonomous Fleet Optimization System

Agile Sprint Schedule (14 Weeks)

Project Duration: 14 weeks | **Sprint Length:** 1 week | **Total Sprints:** 14 **Team Load:** 6-course load + project (light to moderate commitment per sprint)

PHASE 1: THE SKELETON (Weeks 1-3)

Goal: Establish foundation with working map and basic navigation

Sprint 1: Project Setup & Map Infrastructure (Week 1)

Sprint Goal: Set up development environment and prepare baseline map data

User Stories:

1. **US-001: Initialize Development Environment**
    - As a developer, I want all required libraries installed and configured so that I can begin coding
    - Acceptance Criteria:
        - pip environment set up with osmnx, networkx, numpy, matplotlib
        - All imports tested and verified working
        - Project structure created (src/, tests/, data/, docs/ folders)
    - Estimated Points: 3
    - Tasks: Install packages, create folder structure, verify imports
2. **US-002: Select and Validate Sandbox Map**
    - As a project lead, I want a well-defined, manageable sandbox area so that testing is efficient
    - Acceptance Criteria:
        - Model Town, Lahore selected as primary sandbox
        - Map boundaries documented (lat/lon coordinates)
        - OSM data quality verified (no isolated islands expected)
    - Estimated Points: 2
    - Tasks: Research Model Town, identify boundaries, validate OSM coverage
3. **US-003: Implement Map Baking Pipeline (bake_map.py)**
    - As a system architect, I want raw OSM data converted to a clean graph so that algorithms can operate efficiently

- Acceptance Criteria:
  - OSM data downloaded for Model Town
  - Isolated nodes removed (connectivity check enforced)
  - Graph exported as modeltown.graphml
  - Node/edge count and graph statistics logged
- Estimated Points: 5
- Tasks: Write OSMnx query, implement isolation removal, save graphml format

4. **US-004: Create Distance Matrix Utility Module**
   - As a developer, I want a reusable function to compute and cache distances so that lookups are O(1)
   - Acceptance Criteria:
     - Floyd-Warshall or NetworkX shortest path pre-computed
     - Distance matrix saved as .npy file
     - Loading/caching function created
     - Verification script confirms matrix correctness
   - Estimated Points: 5
   - Tasks: Implement matrix computation, create serialization, write verification tests

---

Sprint 2: Graph Navigation & A* Search (Week 2)

Sprint Goal: Implement custom A* pathfinder and validate on real map

User Stories:

1. *US-005: Implement Custom A Pathfinding Algorithm**
   - As an AI engineer, I want A* search implemented from scratch so that I understand the algorithm and can extend it
   - Acceptance Criteria:
     - Priority queue-based A* written without nx.shortest_path
     - Haversine heuristic implemented (lat/lon coordinates)
     - Function returns ordered list of nodes from start to goal
     - Handles edge cases: unreachable nodes, same start/goal, empty graph
   - Estimated Points: 8
   - Tasks: Write priority queue logic, implement heuristic, handle edge cases, code review
2. *US-006: Validate A with Manual Test Cases**

- As a QA engineer, I want concrete test cases so that A* correctness is verified
- Acceptance Criteria:
  - 5 test cases defined: short path, long path, unreachable, same node, complex routing
  - Visual validation: plot start, goal, and path on map
  - Path validity confirmed (all edges exist in graph)
  - Performance logged: path computation < 50ms on sandbox
- Estimated Points: 4
- Tasks: Define test cases, write validation script, visualize paths, benchmark

3. **US-007: Create Path Visualization Module**
   - As a developer, I want to visualize paths on the map so that correctness is obvious
   - Acceptance Criteria:
     - Matplotlib-based visualization created
     - Start node marked (green), goal marked (red), path in blue
     - Map background shows street network
     - Interactivity: zoom/pan functional
   - Estimated Points: 4
   - Tasks: Write plotting function, style map layers, test interactivity
4. **US-008: Document Navigation Layer API**
   - As a future maintainer, I want clear API documentation so that the module is reusable
   - Acceptance Criteria:
     - Function signatures documented (docstrings)
     - Algorithm explanation (pseudocode)
     - Example usage script provided
     - README updated with navigation module overview
   - Estimated Points: 2
   - Tasks: Write docstrings, create examples, update README

Sprint 3: Genetic Algorithm & Strategic Solver (Week 3)

Sprint Goal: Implement facility location optimization using evolutionary algorithm

User Stories:

1. **US-009: Implement Genetic Algorithm Framework**

- As an optimization engineer, I want a generic GA so that base station locations are evolved
- Acceptance Criteria:
  - Genome representation: list of 5 node IDs
  - Population initialization: 50 random genomes
  - Selection mechanism: fitness-proportionate (roulette wheel)
  - Mutation: random node swap
  - Crossover: single-point crossover
- Estimated Points: 8
- Tasks: Implement GA class, write fitness evaluation, implement operators

2. **US-010: Implement Fitness Function (Facility Location)**
   - As a data scientist, I want a fitness function that minimizes total travel time so that GA evolves good solutions
   - Acceptance Criteria:
     - Fitness = sum of distances from all nodes to nearest ambulance base
     - Distances pre-computed from distance matrix (O(1) lookup)
     - Handles edge cases: duplicate bases penalized, all nodes reachable
     - Fitness values logged per generation
   - Estimated Points: 5
   - Tasks: Implement fitness calculation, add penalty logic, test with known solutions
3. **US-011: Run GA to Convergence & Track Progress**
   - As a solver developer, I want GA to converge and visualize evolution so that solution quality is observable
   - Acceptance Criteria:
     - GA runs for 100 generations
     - Best fitness logged per generation
     - Convergence plot created (matplotlib)
     - Final 5 station locations recorded and exported
     - Execution time < 2 minutes on sandbox
   - Estimated Points: 4
   - Tasks: Write GA execution loop, create convergence tracker, plot results
4. **US-012: Visualize Optimal Station Locations on Map**
   - As a product manager, I want to see where stations are placed so that the solution is interpretable

- Acceptance Criteria:
  - Map plotted with street network
  - Optimal stations marked as blue stars
  - Coverage zones visualized (Voronoi or heatmap)
  - Comparison: random stations (gray) vs. optimal (blue)
  - HTML report generated with visualization
- Estimated Points: 4
- Tasks: Implement Voronoi/coverage visualization, create comparison plot, generate report

---

PHASE 2: THE BRAIN (Weeks 4-7)

Goal: Add simulation engine and dynamic intelligence

Sprint 4: Simulation Engine & Ambulance Agents (Week 4)

Sprint Goal: Create basic simulation loop with mobile ambulance agents

User Stories:

1. **US-013: Implement Ambulance Agent Class**
   - As a simulation developer, I want an Ambulance class so that individual agents are represented
   - Acceptance Criteria:
     - State machine: IDLE, IN_TRANSIT, ON_SCENE
     - Properties: current_location, current_path, assigned_task
     - Methods: navigate(destination), update_position(), get_status()
     - Location updated based on A* path each simulation tick
   - Estimated Points: 5
   - Tasks: Design state machine, implement class, write unit tests
2. **US-014: Implement Event Spawner (Poisson Distribution)**
   - As a simulation engineer, I want random accident generation so that emergency calls are realistic
   - Acceptance Criteria:
     - Poisson process generates accidents with configurable λ (rate parameter)
     - Accidents spawn at random nodes
     - Accident attributes: timestamp, location, priority
     - Spawning logic isolated in separate function for easy modification

- Estimated Points: 4
- Tasks: Implement Poisson logic, test distribution, create configurability

3. **US-015: Build Main Simulation Loop**
    - As a simulator, I want a tick-based engine so that time-stepped physics works
    - Acceptance Criteria:
        - Main loop processes N ticks (configurable)
        - Each tick: (1) update ambulance positions, (2) spawn events, (3) assign idle ambulances
        - Simulation state tracked (current_tick, active_events, ambulance_positions)
        - Loop runs without errors for 1000+ ticks
    - Estimated Points: 6
    - Tasks: Design loop architecture, implement tick logic, handle state management
4. **US-016: Create Basic Live Visualization (Matplotlib Interactive)**
    - As a developer, I want to see the simulation running so that behavior is observable
    - Acceptance Criteria:
        - Interactive matplotlib plot created (plt.ion() mode)
        - Ambulances shown as colored dots on map
        - Accidents shown as red X markers
        - Plot redraws every N ticks (configurable)
        - FPS sufficient for real-time observation
    - Estimated Points: 4
    - Tasks: Implement interactive plotting, optimize refresh rate, test stability

---

Sprint 5: Dispatcher Logic & Task Assignment (Week 5)

Sprint Goal: Implement dispatch brain that assigns ambulances to emergencies

User Stories:

1. **US-017: Implement Dispatcher Brain Class**
    - As a system architect, I want a centralized dispatcher so that assignment decisions are coordinated
    - Acceptance Criteria:
        - DispatcherBrain class created
        - Methods: assign_task(ambulance, event), rebalance_fleet()
        - Tracks active events and idle ambulances

        - Decision logic: assign nearest idle ambulance to new event
    - Estimated Points: 5
    - Tasks: Design dispatcher class, implement assignment logic, write tests
2. **US-018: Implement Task Assignment Algorithm**
    - As an operations researcher, I want optimal assignment so that response times are minimized
    - Acceptance Criteria:
        - Algorithm: find nearest idle ambulance to event using distance matrix
        - Ties broken by ambulance_id for determinism
        - Assignment creates navigation path using A*
        - Event marked as "assigned" in dispatcher state
    - Estimated Points: 4
    - Tasks: Implement assignment, integrate with A*, test edge cases
3. **US-019: Track Response Time Metrics**
    - As a data analyst, I want metrics so that performance is measurable
    - Acceptance Criteria:
        - Response time = spawn_time - arrival_time
        - All response times logged per event
        - Average Response Time (ART) computed after simulation
        - Metrics exported to CSV for analysis
    - Estimated Points: 3
    - Tasks: Implement metrics tracking, add logging, export function
4. **US-020: Integrate Dispatcher into Simulation Loop**
    - As an engineer, I want dispatcher integrated so that the full system works
    - Acceptance Criteria:
        - Dispatcher called each tick
        - New events auto-assigned to idle ambulances
        - Ambulances navigate using A* paths
        - Simulation runs with full dispatch logic for 1000 ticks
    - Estimated Points: 3
    - Tasks: Integrate components, test full loop, debug interaction

---

Sprint 6: K-Means Clustering & Dynamic Hotspot Detection (Week 6)

Sprint Goal: Implement unsupervised learning for demand prediction

User Stories:

1. **US-021: Implement K-Means Clustering from Scratch**
    - As a data scientist, I want K-Means implemented manually so that I understand the algorithm
    - Acceptance Criteria:
        - Manual K-Means written (no sklearn)
        - Centroid initialization: random or k-means++
        - Iterative update: reassign points, recompute centroids
        - Convergence check: centroid movement < epsilon
        - Returns cluster assignments and centroids
    - Estimated Points: 8
    - Tasks: Implement clustering algorithm, write convergence logic, test on sample data
2. **US-022: Create Demand Clustering Module**
    - As a tactical system, I want to detect accident hotspots so that ambulances pre-position
    - Acceptance Criteria:
        - Input: list of active accident locations
        - Process: run K-Means with k=2-3 (configurable)
        - Output: cluster centroids (hotspot locations)
        - Minimum cluster size enforced (to avoid noise)
    - Estimated Points: 4
    - Tasks: Wrap K-Means for demand data, add filtering, write validation
3. **US-023: Integrate Dynamic Hotspot Rebalancing into Dispatcher**
    - As a dispatcher, I want idle ambulances repositioned so that coverage is reactive
    - Acceptance Criteria:
        - Every 50 ticks, run K-Means on active events
        - If hotspots found, move idle ambulances to centroids
        - Ambulances not disrupted if already assigned
        - Rebalancing logged for analysis
    - Estimated Points: 5
    - Tasks: Create rebalancing function, integrate into dispatcher tick, add logging

4. **US-024: Visualize Hotspots on Live Map**
    - As an observer, I want to see clusters so that behavior is intuitive
    - Acceptance Criteria:
        - Cluster centroids shown as blue circles on map
        - Cluster membership shown (e.g., shaded regions)
        - Hotspot evolution visible over time
        - Overlay on existing ambulance visualization
    - Estimated Points: 4
    - Tasks: Implement cluster visualization, optimize rendering, test with real runs

---

Sprint 7: Simulation Refinement & Traffic Dynamics (Week 7)

Sprint Goal: Add realism with traffic simulation and edge weight dynamics

User Stories:

1. **US-025: Implement Traffic Congestion Simulation**
    - As a realism engineer, I want traffic to slow ambulances so that simulation is realistic
    - Acceptance Criteria:
        - Traffic weight multiplier: 1.0 (free flow) to 2.5 (congestion)
        - Congestion zones: high-accident areas incur 2x time penalty
        - Dynamic: congestion increases with event density, decays over time
        - A* re-routes around congestion (uses current edge weights)
    - Estimated Points: 6
    - Tasks: Implement congestion model, integrate with graph weights, test rerouting
2. **US-026: Implement Ambulance Re-routing on Path**
    - As a navigator, I want ambulances to re-route around obstacles so that they adapt
    - Acceptance Criteria:
        - While ambulance is in transit, check for congestion updates
        - If blocked node's weight increased significantly, recompute path
        - New path computed using A* and current weights
        - Re-routing logged with reason (e.g., "congestion detected")
    - Estimated Points: 4
    - Tasks: Implement re-routing check, compute new paths, track re-routes

3. **US-027: Add Simulation Parameters Configuration File**
    - As a researcher, I want configurable parameters so that I can run experiments
    - Acceptance Criteria:
        - YAML/JSON config file with all parameters (λ, traffic_factor, k_update_interval, etc.)
        - Config loader: parse file and set simulation parameters
        - Documentation: each parameter explained with default/range
        - Example configs: baseline, high-traffic, cluster-heavy
    - Estimated Points: 3
    - Tasks: Design config schema, implement loader, create example configs
4. **US-028: Create Comprehensive Simulation Logging System**
    - As a debugger, I want detailed logs so that I can analyze and debug
    - Acceptance Criteria:
        - Event-level logging: spawn, assignment, arrival, completion
        - Ambulance-level logging: state changes, path updates, re-routes
        - Tick-level logging: number of active events, idle ambulances, hotspots
        - Logs written to CSV and live console output
    - Estimated Points: 3
    - Tasks: Implement logging framework, add log points throughout code, test output

---

PHASE 3: THE PULSE (Weeks 8-12)

Goal: Complete simulation, optimize, and run experiments

Sprint 8: Baseline Comparison & Random Fleet (Week 8)

Sprint Goal: Establish performance baseline with random station placement

User Stories:

1. **US-029: Implement Random Station Placement Generator**
    - As a baseline engineer, I want a random fleet so that I have a control group
    - Acceptance Criteria:
        - Generate 5 random node IDs from graph (no duplicates)
        - Repeat generation N times (e.g., 10) to get average baseline
        - Random placements logged for reproducibility

        - Function returns list of node IDs
    - Estimated Points: 2
    - Tasks: Implement random selection, test reproducibility

2. **US-030: Run Simulation with Random Fleet (Baseline)**
    - As a researcher, I want a baseline ART so that optimization gains are measurable
    - Acceptance Criteria:
        - Run simulation 10 times with different random seeds
        - Each run: 1000+ ticks with Poisson event spawning
        - Collect ART (Average Response Time) for each run
        - Calculate mean ART and standard deviation
        - Results exported to CSV
    - Estimated Points: 5
    - Tasks: Create baseline runner script, run experiments, analyze results

3. **US-031: Document Random Fleet Baseline Results**
    - As a analyst, I want clear baseline documentation so that comparisons are meaningful
    - Acceptance Criteria:
        - Report: baseline ART, std dev, number of events processed
        - Visualizations: ART distribution (histogram), time-series response times
        - Analysis: observations about random placement inefficiencies
        - Markdown report generated automatically
    - Estimated Points: 3
    - Tasks: Create analysis script, generate visualizations, write report

4. **US-032: Create Baseline Configuration & Reproducibility Seed**
    - As a researcher, I want reproducible results so that experiments are valid
    - Acceptance Criteria:
        - Baseline seed documented (random_seed for random fleet generation)
        - Simulation seeds set (event spawning, ambulance initialization)
        - Config version tracked
        - README documents how to reproduce baseline run
    - Estimated Points: 2
    - Tasks: Set up seed management, document process

Sprint 9: AI-Optimized Fleet & Comparison (Week 9)

Sprint Goal: Run optimized simulation and compare against baseline

User Stories:

1. **US-033: Run Simulation with AI-Optimized Fleet**
    - As a researcher, I want AI fleet performance so that optimization impact is measured
    - Acceptance Criteria:
        - Load GA-optimal stations from Week 3 results
        - Run simulation 10 times (same seed series as baseline for fair comparison)
        - Each run: 1000+ ticks with identical event distribution
        - Collect ART for each run
        - Results exported to CSV
    - Estimated Points: 4
    - Tasks: Load optimal stations, run experiment loop, export results
2. **US-034: Implement Head-to-Head Comparison Analysis**
    - As a data scientist, I want statistical comparison so that the improvement is quantified
    - Acceptance Criteria:
        - Compare baseline ART vs. AI ART (paired samples)
        - Calculate: absolute improvement, percentage improvement, standard error
        - Statistical test: t-test for significance (p-value)
        - All metrics documented
    - Estimated Points: 4
    - Tasks: Implement comparison logic, run statistical tests, document results
3. **US-035: Generate Comparison Visualizations**
    - As a presenter, I want visual comparisons so that results are compelling
    - Acceptance Criteria:
        - Plot 1: Side-by-side ART distributions (baseline vs. AI)
        - Plot 2: Time-series response times for both fleets
        - Plot 3: Heatmaps showing demand coverage (baseline vs. AI)
        - Plot 4: Station placement comparison (random vs. optimal)
        - All plots with error bars and significance markers
    - Estimated Points: 5

- Tasks: Create comparison plots, style for presentation, verify correctness

4. **US-036: Create Results Summary Document**
    - As a reporter, I want a concise summary so that key findings are clear
    - Acceptance Criteria:
        - Executive summary: key metrics and findings
        - Detailed results table: all ART values, improvements
        - Methodology: how experiments were conducted
        - Caveats & limitations noted
        - PDF/Markdown report generated
    - Estimated Points: 3
    - Tasks: Write summary, compile all results, generate PDF

Sprint 10: Sensitivity Analysis & Parameter Tuning (Week 10)

Sprint Goal: Understand how parameters affect performance

User Stories:

1. **US-037: Run Sensitivity Analysis on Event Rate (Lambda)**
    - As an analyst, I want to understand load sensitivity so that performance is characterized
    - Acceptance Criteria:
        - Run simulations with $\lambda = 0.01, 0.05, 0.1, 0.15$ events/tick
        - For each λ: test both baseline and AI fleets
        - Collect ART and total events processed
        - Plot ART vs. λ for both strategies
    - Estimated Points: 4
    - Tasks: Implement parameter sweep, run experiments, plot results
2. **US-038: Run Sensitivity Analysis on Number of Ambulances**
    - As a fleet manager, I want to understand scaling so that fleet sizing is informed
    - Acceptance Criteria:
        - Run simulations with 3, 5, 7, 10 ambulances
        - For each count: test both baseline and AI fleets
        - Collect ART and coverage metrics
        - Plot ART vs. fleet size

- Estimated Points: 4
- Tasks: Implement fleet size variation, run experiments, analyze scalability

3. **US-039: Analyze K-Means Sensitivity (k parameter, update frequency)**
    - As a clustering engineer, I want optimal hotspot parameters so that dynamic dispatch is tuned
    - Acceptance Criteria:
        - Test k=2, 3, 4 for clustering
        - Test update frequencies: every 25, 50, 100 ticks
        - Measure ART and rebalancing count for each combination
        - Recommend optimal configuration
    - Estimated Points: 4
    - Tasks: Parameter sweep for clustering, run experiments, recommend settings
4. **US-040: Generate Sensitivity Analysis Report**
    - As a researcher, I want comprehensive analysis so that design choices are justified
    - Acceptance Criteria:
        - Report: all sensitivity analyses with visualizations
        - Table: recommended parameters with justification
        - Trade-offs discussed (e.g., responsiveness vs. churn)
        - Future optimization directions noted
    - Estimated Points: 3
    - Tasks: Compile all analyses, create visualizations, write report

---

Sprint 11: Integration Testing & Edge Cases (Week 11)

Sprint Goal: Harden system against edge cases and verify robustness

User Stories:

1. **US-041: Implement Comprehensive Unit Tests**
    - As a QA engineer, I want unit tests so that components are reliable
    - Acceptance Criteria:
        - Tests for A*: shortest paths, unreachable nodes, single-node
        - Tests for GA: fitness calculation, mutation, convergence
        - Tests for K-Means: convergence, single-cluster, divergent data
        - Tests for Dispatcher: assignment logic, re-routing

    - Minimum 80% code coverage
- Estimated Points: 6
- Tasks: Write unit tests, run coverage analysis, fix coverage gaps

2. **US-042: Implement Integration Tests**
    - As a tester, I want end-to-end tests so that system behavior is correct
    - Acceptance Criteria:
        - Test 1: Full simulation run (setup → 100 ticks → shutdown)
        - Test 2: GA→Dispatcher→Simulation pipeline
        - Test 3: Dynamic rebalancing triggers and works
        - Test 4: Traffic congestion causes re-routing
        - All tests pass without errors
    - Estimated Points: 5
    - Tasks: Design integration test scenarios, implement, run tests
3. **US-043: Test Edge Cases & Failure Modes**
    - As a reliability engineer, I want edge cases handled so that system is robust
    - Acceptance Criteria:
        - Empty graph / no nodes: handled gracefully
        - No idle ambulances: event queued for later
        - All ambulances busy: response time degraded but valid
        - Unreachable accident location: logged and skipped
        - Traffic congestion 100%: routing adapts
    - Estimated Points: 4
    - Tasks: Identify edge cases, implement handling, test each case
4. **US-044: Create Regression Test Suite**
    - As a developer, I want regression tests so that refactoring is safe
    - Acceptance Criteria:
        - Regression suite: 10+ test scenarios with known expected outputs
        - All baseline results recorded
        - Test suite runs in < 2 minutes
        - CI-ready (all tests pass before merge)
    - Estimated Points: 3
    - Tasks: Create test scenarios, record baselines, set up test runner

---

Sprint 12: Performance Optimization & Scaling (Week 12)

Sprint Goal: Optimize for larger simulations and longer runs

User Stories:

1. **US-045: Profile Code & Identify Bottlenecks**
    - As a performance engineer, I want profiling data so that optimization is targeted
    - Acceptance Criteria:
        - Python profiler (cProfile) run on full simulation
        - Top 5 slowest functions identified
        - Call graph generated
        - Bottleneck root causes documented
    - Estimated Points: 3
    - Tasks: Run profiler, analyze output, document findings
2. *US-046: Optimize A Search Performance**
    - As a pathfinding engineer, I want faster A* so that real-time routing is feasible
    - Acceptance Criteria:
        - Current A*: target < 50ms (baseline from Week 2)
        - Optimizations: better heuristic, pruning, caching
        - Benchmark: 10-path average time
        - No correctness regression
    - Estimated Points: 5
    - Tasks: Implement optimizations, benchmark, test correctness
3. **US-047: Optimize K-Means Convergence**
    - As a clustering engineer, I want faster convergence so that rebalancing is quicker
    - Acceptance Criteria:
        - Current K-Means: target < 100ms for 100 points
        - Optimizations: early stopping, vectorization (numpy)
        - Benchmark: convergence time per run
        - Cluster quality maintained
    - Estimated Points: 4

- Tasks: Vectorize code, implement early stopping, benchmark

4. **US-048: Enable Simulation Scaling (10000+ ticks, 10+ ambulances)**
    - As a researcher, I want long-running simulations so that patterns emerge
    - Acceptance Criteria:
        - Simulate 10000+ ticks without memory issues
        - 10+ ambulances without slowdown
        - Logging streamlined (write to disk, not memory)
        - Simulation runs in < 10 minutes
    - Estimated Points: 4
    - Tasks: Implement disk logging, optimize data structures, run long simulations

---

PHASE 4: THE POLISH (Weeks 13-14)

Goal: Documentation, reporting, and final refinement

Sprint 13: Documentation & Code Quality (Week 13)

Sprint Goal: Complete documentation and ensure code quality

User Stories:

1. **US-049: Complete Code Documentation**
    - As a maintainer, I want clear documentation so that code is understandable
    - Acceptance Criteria:
        - All functions have docstrings (numpy format)
        - Module-level documentation for each .py file
        - Algorithm explanations (pseudocode for complex algorithms)
        - Examples provided for public APIs
    - Estimated Points: 4
    - Tasks: Write docstrings, create module READMEs, add examples
2. **US-050: Create Architecture Documentation**
    - As an onboarder, I want system architecture so that I understand the design
    - Acceptance Criteria:
        - Architecture diagram (components, data flow)
        - Design patterns documented (Entity-Component, simulation loop)

        - Class hierarchy documented
        - Data structure descriptions (CityGraph, DistanceMatrix, etc.)
    - Estimated Points: 4
    - Tasks: Create diagrams, write architecture doc, explain design choices
3. **US-051: Refactor Code for Readability & Maintainability**
    - As a code reviewer, I want clean code so that future development is easier
    - Acceptance Criteria:
        - Code style consistent (PEP 8)
        - Variable names clear and descriptive
        - Functions single-responsibility
        - No dead code or commented-out blocks
    - Estimated Points: 4
    - Tasks: Run linter (pylint/flake8), refactor, remove dead code
4. **US-052: Create User Guide & Usage Documentation**
    - As an end user, I want clear instructions so that I can run the system
    - Acceptance Criteria:
        - Setup instructions (step-by-step)
        - Configuration guide (parameters, config file)
        - Running the simulation (baseline, AI, experiments)
        - Interpreting results and visualizations
    - Estimated Points: 3
    - Tasks: Write user guide, create quick-start, provide example commands

---

Sprint 14: Final Report & Presentation (Week 14)

Sprint Goal: Complete final project deliverables

User Stories:

1. **US-053: Compile Final Project Report**
    - As an author, I want a comprehensive report so that the project is fully documented
    - Acceptance Criteria:
        - Title page with project info
        - Executive summary (1 page)

- Introduction & motivation
- Technical approach (algorithms, architecture)
- Results: baseline, AI, comparisons, sensitivity analysis
- Conclusions & recommendations
- All visualizations embedded
- References and bibliography

- Estimated Points: 5
- Tasks: Write sections, compile visualizations, format report, proofread

2. **US-054: Create Presentation Slides**
    - As a presenter, I want compelling slides so that the project shines
    - Acceptance Criteria:
        - Title slide
        - Problem statement (motivation)
        - Technical overview (brief)
        - Key algorithms (visuals)
        - Results comparison (impact)
        - Demo / live simulation walkthrough
        - Conclusions & future work
        - 20-30 minute duration
    - Estimated Points: 4
    - Tasks: Design slides, gather visuals, practice presentation
3. **US-055: Prepare Live Demo & Walkthroughs**
    - As a demonstrator, I want working demo so that the system is tangible
    - Acceptance Criteria:
        - Quick-start script runs in < 5 minutes
        - Visualization shows ambulance movements, events, hotspots clearly
        - Optional: interactive mode to pause/resume/adjust parameters
        - Fallback: pre-recorded video if live demo fails
    - Estimated Points: 3
    - Tasks: Create demo script, test reproducibility, record video backup
4. **US-056: Final Review, Polish, & Submission**
    - As a project lead, I want quality assurance so that deliverables are professional

- Acceptance Criteria:
  - Code review: all PRs merged, tests passing
  - Report review: spelling, grammar, formatting
  - Presentation review: clarity, flow, visuals
  - Final README with all deliverables listed
  - All files organized in submission folder
- Estimated Points: 3
- Tasks: Final review, address feedback, prepare submission package

---

Summary Table

Sprint	Week	Phase	Focus	Key Deliverables
1	1	Skeleton	Setup & Map	Environment, modeltown.graphml, distance matrix
2	2	Skeleton	Navigation	A* algorithm, test cases, path visualization
3	3	Skeleton	Optimization	GA solver, optimal stations, coverage visualization
4	4	Brain	Agents & Simulation	Ambulance class, event spawner, main loop, live viz
5	5	Brain	Dispatch	Dispatcher class, task assignment, response tracking
6	6	Brain	Clustering	K-Means implementation, hotspot detection, visualization
7	7	Brain	Realism	Traffic dynamics, re-routing, config system, logging
8	8	Pulse	Baseline	Random fleet baseline, ART measurement, comparison setup
9	9	Pulse	Optimization	AI fleet run, head-to-head comparison, results viz
10	10	Pulse	Sensitivity	Parameter analysis (λ, fleet size, k), report
11	11	Pulse	Testing	Unit tests, integration tests, edge cases, regression suite
12	12	Pulse	Scaling	Performance profiling, optimization, long-run capability
13	13	Polish	Documentation	Code docs, architecture docs, refactoring, user guide
14	14	Polish	Delivery	Final report, presentation, demo, submission

---

Effort Distribution

- **Total Estimated Points:** 168 points (averaging 12 points per sprint)
- **Phase 1 (Weeks 1-3):** 47 points — Foundation building
- **Phase 2 (Weeks 4-7):** 48 points — Feature development
- **Phase 3 (Weeks 8-12):** 56 points — Validation & optimization
- **Phase 4 (Weeks 13-14):** 17 points — Documentation & delivery

---

Weekly Time Commitment

Given your 6-course load, estimated time per sprint:

- **Light Sprints (1-2, 8, 13-14):** 8-10 hours/week
- **Medium Sprints (3-5, 9-12):** 12-15 hours/week
- **Heavy Sprints (6-7, 10-11):** 15-18 hours/week

Average: ~12 hours/week for project work

Sprint Planning Tips

1. **Backlog Refinement:** Review user stories before each sprint
2. **Estimation:** Use planning poker or T-shirt sizing for better accuracy
3. **Standups:** Quick daily 15-min check-ins (even solo, helps track progress)
4. **Retrospective:** End-of-sprint reflection on what went well, what to improve
5. **Buffer:** Allocate 10-20% contingency for unexpected issues

---

Success Criteria

✓ All user stories completed by end of sprint
✓ Code tested (unit + integration)
✓ Documentation updated
✓ No technical debt carried forward
✓ Final project demonstrates 20%+ ART improvement over baseline