Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Software Specification

Agent-Based Simulation and 3D Visualization of a Logistics Network

Mateusz Polis, Mateusz Nędzi

16.10.2025

Table of Contents

1 Abstract

This document presents the requirements for our bachelor's thesis project of an Agent-Based Simulation and 3D Visualization of a Logistics Network. Functional requirements are specified in the form of use case descriptions and diagrams. Non-functional requirements covering Usability, Reliability, Performance and Supportability are enumerated in a table. A detailed schedule explains the steps we need to take to complete the project. Deliverables which we plan to deliver for each development and testing milestone are explained. Finally, we conducted an analysis of Strengths, Weaknesses, Opportunities and Threats, to identify potential future issues and be prepared to handle them.

1.1 History of changes

Date	Author	Description	Version
16.10.2025	M. Nędzi, M. Polis	Initial document revision	1.0

2 Vocabulary

Map - A data structure representing the complete logistics network modeled as a directed multigraph.

Edge - A connection between two **Nodes** in the **Map's** graph, representing a traversable route (e.g. road). Edges are directed, defining the allowed direction of movement between **Nodes**. Each edge includes attributes relevant to the simulation of traffic flow – such as distance, capacity and maximum speed – along with any additional parameters necessary to capture route-specific conditions. Edges may also be constrained (e.g. weight/height limits). **Agents** move along the edges. **Events** like delays or blockages are modelled on edges.

Node - A vertex in the **Map's** graph. Nodes mark where **Edges** meet. A node can correspond to one or more physical locations – **Buildings** – such as warehouses, depots etc. Each node carries geographic coordinates.

Agent - An autonomous entity within the simulation that perceives its environment and acts according to defined behavioral rules. Agents may represent mobile entities (moving along the **Edges**), stationary entities (**Buildings** located at **Nodes**), or external entities not directly represented on the **Map** (e.g. a Broker agent coordinating routes). Each agent type possesses its own set of attributes and decision-making logic, allowing for dynamic interactions and emergent behaviors within the simulated **Logistics Network**.

Event – An occurrence within the simulation that represent a change of state in the **Logistics Network**. Events may take place at **Nodes,** within **Buildings**, along **Edges**, or for specific **Agents**. They capture dynamic phenomena such as vehicle arrivals, loading and unloading operations, traffic delays, equipment failures or decision triggers. Each event is characterized by a type, time, and affected entities. Events can occur deterministically or with a degree of randomness to reflect the uncertainty and variability of real-world logistic processes, enhancing the realism of the simulation.

Building – A physical facility located at a **Node** within the **Map**. Buildings represent logistic infrastructure such as warehouses, depots or retail outlets. A Building may function as an **Agent**, actively participating in the simulation (e.g. managing inventory, dispatching vehicles), but it does not have to. Some may serve as passive locations or resources. Each building possesses its own set of parameters relevant to its role, such as storage capacity, processing rate, operational hours, or handling costs. However, these parameters are not common across all Buildings, as each type exhibits distinct behavior and functionality within the simulated **Logistics Network**.

Logistics Network – The environment in which all simulation activity takes place, representing the interconnected system of transportation routes, facilities and operational entities responsible for the movement of goods. The Logistics Network is composed of **Nodes**, **Edges**, and **Buildings**, forming the structural backbone of the simulation's **Map**. Within this environment, the **Agentic System** operates. The Logistics Network serves both as a physical model of infrastructure and as a context in which agent behaviors and system-wide logistics processes emerge.

Agentic System – A multi-agent environment operating within the **Logistics Network**, composed of autonomous **Agents** that perceive their surroundings, make decisions and act according to predefined or adaptive behavioral rules. The Agentic System governs the interactions between **Agents** such as Vehicles, **Buildings** or Brokers and their responses to **Events** occurring across the

Map. Through these interactions, the **Agentic System** produces emergent behaviors that reflect the complexity of real-world logistics processes, such as congestion, resource allocation and coordination.

Frontend – A web-based interface responsible for visualizing and interacting with the simulation. It renders the 3D environment using Three.js [1] and provides and interactive overlay built with React [2] for controls, information panels, and event details. The Frontend communicates with the **Backend** through a WebSocket [3] connection, sending **Actions** that can influence the simulation and receiving **Signals** that describe state changes and events to be displayed.

Backend – A server-side system responsible for executing the Agentic Simulation and managing the overall logic of the **Logistics Network**. The Backend maintains the state of all **Agents**, processes **Events**, and produces **Actions** that describe updates to be reflected in the **Frontend**. It communicates with the **Frontend** through a WebSocket [3] connection, continuously sending **Signals** and receiving **Actions** that represent user commands or parameter changes. The Backend ensures deterministic simulation flow, consistency across connected clients, and real-time synchronization between the computational model and its 3D visualization.

Action – A message sent from the **Frontend** to the **Backend** through the WebSocket [3] connection to influence the simulation. Signals represent user commands or parameter changes, such as modifying an **Agent's** behavior, removing an **Element**, or controlling the simulation state (e.g. play, pause, reset).

Signal – A message sent from the **Backend** to the **Frontend** describing state changes or events within the simulation. Actions inform the **Frontend** what has occurred – such as an **Agent** moving, an **Event** being triggered, or metrics being updated, so that it can update the 3D visualization accordingly.

Element – An interactive object represented on the **Map** that can be selected by the user to inspect its details. Elements include all simulation relevant components such as **Buildings**, Vehicles, and other **Agents** that influence or participate in the **Logistics Network**. Each Element contains information describing its current state, parameters and recent Events. Non-functional visual objects such as trees, terrain or rivers are not considered Elements, as they serve only a decorative or contextual purpose and do not interact with the simulation logic.

Fleet – A collection of mobile **Agents** within the simulation, typically representing vehicles, such as trucks, that move along the **Edges** of the **Map**. The Fleet forms the dynamic component of the **Logistics Network** responsible for executing transport and delivery operations.

3 Specification

1.1 Executive Summary

Our project involves a web application for a truck logistics company that simulates traffic across a logistics network with respect to truck CO_2 emissions, capacity, speed and fuel consumption. The frontend emphasizes accessibility and ease of use, featuring in-browser 3D visualization, while the backend employs an agent-based simulation to ensure accuracy. To support flexible experimentation, the system includes a map editor for building custom networks and tools to define fleets of trucks with user-specified properties for running simulations.

1.2 Functional Requirements

We specify functional requirements in a form of use case description and a use-case diagram:

Table 1: Description of use cases of interaction of the user with the system

ID	Actor	Name	Description	System Response
UC1	User	Define Logistics Network	The user specifies the structure of the logistics system, including nodes, edges and buildings.	The system creates a new logistics network structure and visualizes it on the map.
UC2		Define Map	The user defines the map topology, adding nodes and edges with parameters such as distance, capacity, and speed limits.	The system registers the map structure and displays it in the 3D environment.
UC3		Define Fleet	The user defines the fleet composition.	The system adds the defined vehicles as agents to the simulation environment.
UC4		Define Truck	The user configures an individual truck kind to be used in the fleet, setting its parameters such as speed, capacity, and starting location.	The system creates the truck as an agent and associates it with the logistics network.
UC5		Interact With Simulation	The user selects and inspects elements or events directly from the 3D visualization.	The system displays an overlay panel with element details, attributes, and/or recent events.
UC6		Control Simulation Playback	The user controls the simulation state using playback buttons (play, pause, stop).	The frontend sends a corresponding action to the backend, and the simulation updates accordingly.
UC7		Speed Up Simulation	The user increases the simulation playback speed.	The system adjusts the simulation timestep and updates the visualization rate.
UC8		Slow Down Simulation	The user decreases the simulation playback speed.	The system adjusts the simulation timestep to a slower rate and updates accordingly.
UC9		Start Simulation	The user initiates the simulation run.	The backend starts the simulation engine, initializes all agents, and begins sending signals to the frontend.
UC10		Pause Simulation	The user pauses an ongoing simulation.	The backend halts simulation progression while preserving current agent states.
UC11		Lookup Simulation State	The user requests an overview of the current global simulation state (e.g., time, active agents, events).	The system retrieves and displays the current simulation metrics and timeline state.

UC12		Lookup Element State	The user requests detailed information about a specific agent or element.	The system returns detailed attributes, parameters, and status of the selected entity.
UC13		Control Camera	The user interacts with the 3D view using mouse or keyboard to rotate, pan, and zoom.	The system updates the camera position and orientation in real time.
UC14		Control Camera Position	The user moves the camera to a predefined or selected element position.	The system transitions the camera smoothly to focus on the selected location or element.
UC15		Control Camera	The user adjusts the viewing angle or perspective within the 3D environment.	The system applies the new camera parameters and updates the visualization.
UC16		Export Logistics Network	The user exports the defined logistics network configuration.	The system saves the current network (nodes, edges, buildings) to a file.
UC17		Import Logistics Network	The user imports a saved logistics network file.	The system parses the data, recreates the map, and visualizes it within the scene.
UC18		Export Simulation State	The user exports the current simulation state, including agent positions and parameters.	The system serializes the simulation data and saves it to an external file.
UC19		Import Simulation State	The user loads a previously saved simulation state.	The system restores agent states, simulation time, and environment conditions to resume execution.

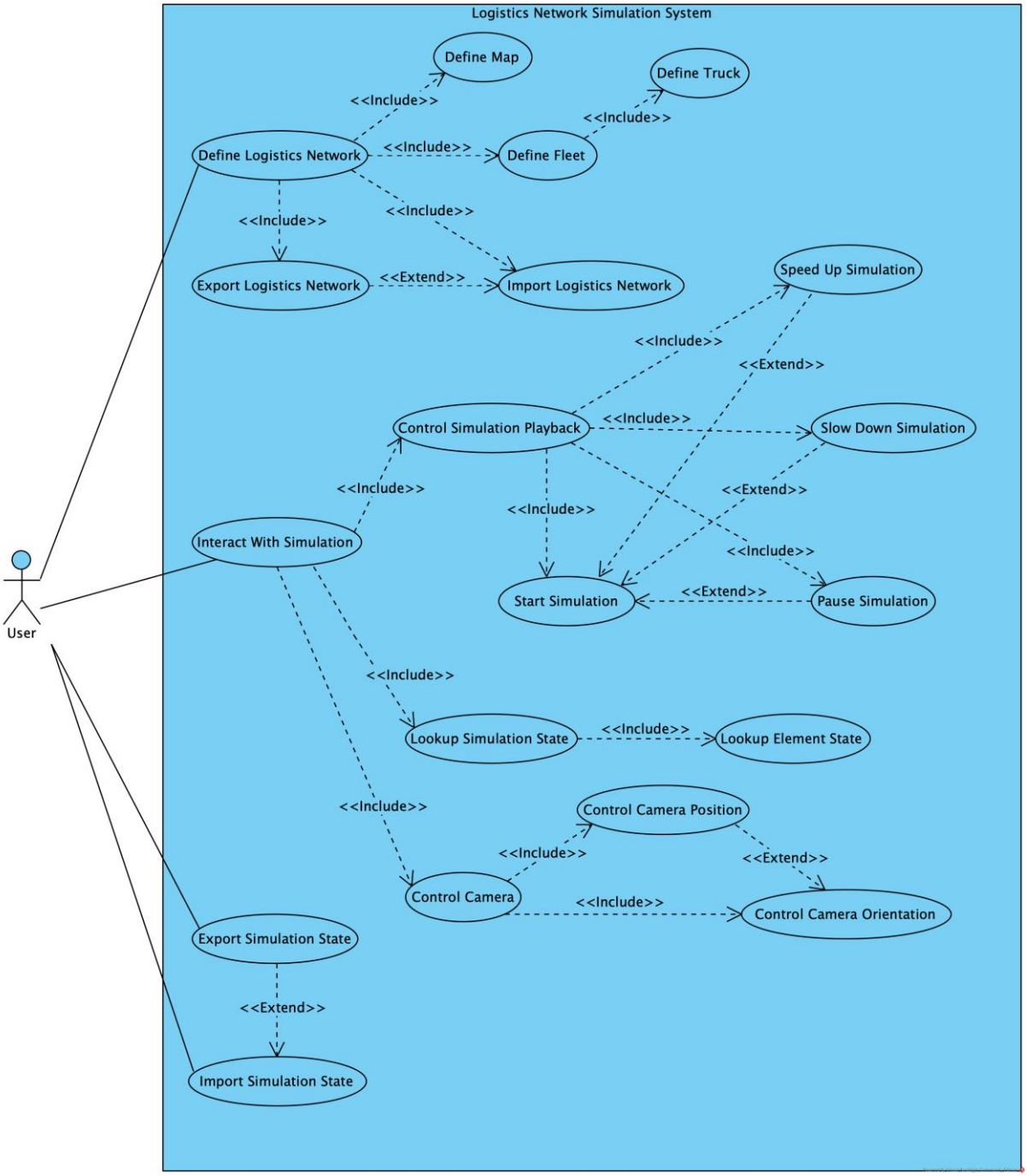

Figure 1: Use-case diagram showing interaction of the user with the system

3.1 Non-functional requirements

Table 2: List of non-functional requirements

Requirements area	Requirement No.	Description
Utility (*Usability*)	1	**Functional and responsive interface**. The Frontend adapts dynamically to various screen sizes and resolutions while maintaining full functionality and readability. It must operate correctly on displays with a minimum resolution of 1280×720 (HD).
	2	**Intuitive navigation**. The user interface must be self-explanatory and easy to learn without formal training. Users should be able to perform basic interactions such as selecting Elements, viewing details, or controlling simulation without external documentation.
Reliability (*Reliability*)	3	**Crash recovery**. The Backend should handle unexpected client disconnections gracefully, preserving simulation continuity for other connected users.
	4	**Network failure resistance**. The system must remain stable in the event of temporary network disruptions. The Frontend automatically attempts to re-connect to the Backend through the WebSocket connection and restores synchronization without user intervention.
	5	**Uptime requirement**. The system should maintain an operational availability of at least 99% during planned usage periods.
	6	**Error reporting**. All errors and connection issues should be logged and reported in a structured way for debugging and diagnostics.
Performance (*Performance*)	7	**Simulation capacity**. The application must support simulations of logistics networks containing up to 100 Nodes and 20 active Agents while maintaining stable performance and real-time responsiveness.
	8	**Rendering efficiency**. The Frontend should maintain a minimum frame rate of 30 FPS under typical simulation conditions and avoid noticeable lag when visualizing agent movement or events.
Maintenance (*Supportability*)	9	**Testing coverage**. Core simulation logic and communication protocols should include automated unit tests to ensure consistent behavior across updates.
	10	**Clear codebase structure**. The software codebase must be organized in a logical and consistent directory structure, with clear separation between modules. File organization should facilitate readability, maintainability, and team collaboration.
	11	**Environment independence**. The entire system should be deployable in a Docker container environment to ensure consistent behavior across different operating systems and hardware configurations. Containerization simplifies setup, testing, and maintenance by encapsulating all dependencies and runtime environments.

4 Project schedule

The project is planned to be implemented per the following schedule:

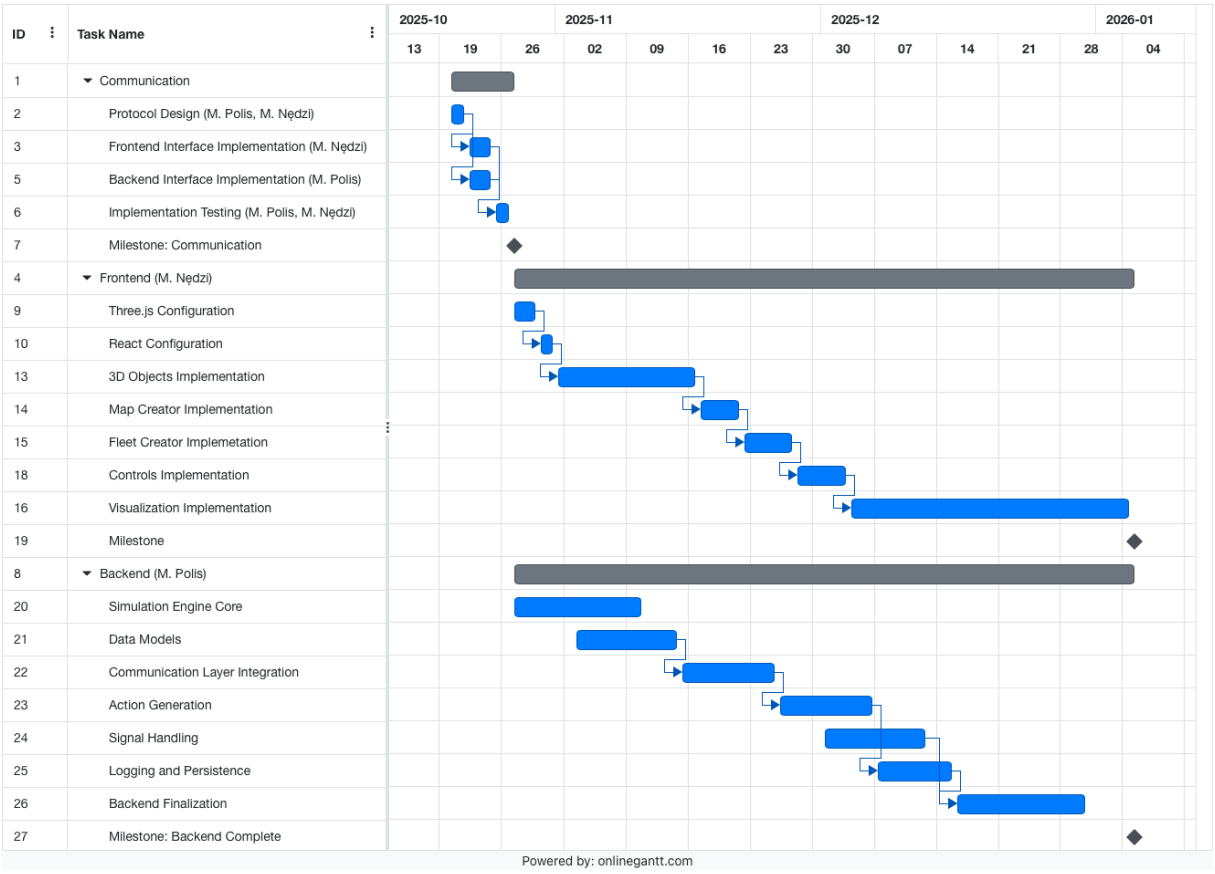

Figure 2: Project schedule, Gantt diagram.

5 Risk Analysis

SWOT	Threats	Opportunities
Internal	1. Limited time due to other academic projects, jobs, thesis-related work. 2. Lack of prior experience with 3D visualization frameworks (Three.js) and WebSocket-based communication. 3. Integration issues between simulation logic (Backend) and visualization (Frontend).	1. Opportunity to gain hands-on experience in full-stack system design combining simulation and web technologies. 2. Possibility to reuse developed components for future research or educational purposes. 3. Strengthening collaboration and project management skills within a software team.
External	4. Unexpected compatibility issues between browsers, operating systems, or hardware when running 3D rendering. 5. Potential delays due to dependencies on external libraries or APIs.	4. Technological advancement and open-source support for agent-based modeling and 3D visualization frameworks. 5. Scalability for further development into an interactive decision-support tool.

Threat 1 - Limited time due to other academic commitments and jobs.

- **Likelihood**: High
- **Impact**: Medium
- **Mitigation**: Create a detailed schedule with milestones and assign responsibilities early. Hold short weekly progress reviews to maintain steady development pace.

Threat 2 - Lack of experience with 3D visualization and WebSocket communication

- **Likelihood**: Medium
- **Impact**: Medium
- **Mitigation**: Begin with a small prototype integrating Three.js and WebSocket early in the project. Allocate time for learning key libraries and reviewing example implementations.

Threat 3 - Integration challenges between Backend and Frontend

- **Likelihood**: Medium
- **Impact**: High
- **Mitigation**: Define a clear data exchange protocol (Actions/Signals) early and test integration continuously rather than at the final stage.

Threat 4 - Browser or hardware compatibility issues

- **Likelihood**: Low to Medium
- **Impact**: Medium

- **Mitigation**: Test the application across multiple browsers (Chrome, Firefox, Safari) and devices; use fallback rendering modes and responsive design principles.

Threat 5 - External dependency delays (library updates or bugs)

- **Likelihood**: Low
- **Impact**: Medium
- **Mitigation**: Freeze library versions during development and document dependency versions for reproducibility.

6 Bibliography

[1] Three.js documentation https://threejs.org/docs/ [16.10.2025]

[2] React learn https://react.dev/learn [16.10.2025]

[3] Websockets API documentation https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API [16.10.2025]