Java II – Project Design

Task A: Design Report

Zachary Patterson & Sarah Richeson

2019

Table of Contents

[1 Introduction 1](#_Toc112711)

[2 Scope 1](#_Toc112712)

[3 UML Diagram 2](#_Toc112713)

[4 Timeline 3](#_Toc112714)

[5 Stakeholders 3](#_Toc112715)

[6 Functional Requirements 3](#_Toc112716)

[6.1 Nodes 3](#_Toc112717)

[6.1.1 Adding Nodes 3](#_Toc112718)

[6.1.2 Removing Nodes 4](#_Toc112719)

[6.1.3 Minimum Number of Nodes 4](#_Toc112720)

[6.2 Edges 4](#_Toc112721)

[6.2.1 Adding Edges 4](#_Toc112722)

[6.2.2 Removing Edges 4](#_Toc112723)

[6.3 Network Connectivity 4](#_Toc112724)

[6.4 Shortest Path 4](#_Toc112725)

[6.4.1 Parallel Edges 4](#_Toc112726)

[6.5 Prioritized Distribution 5](#_Toc112727)

[6.6 Distribution of Multiple Packages 5](#_Toc112728)

[7 Non-Functional Requirements 5](#_Toc112729)

[7.1 Performance 5](#_Toc112730)

[7.2 Operating Constraints 5](#_Toc112731)

[7.3 Platform Constraints 5](#_Toc112732)

[7.4 Usability 5](#_Toc112733)

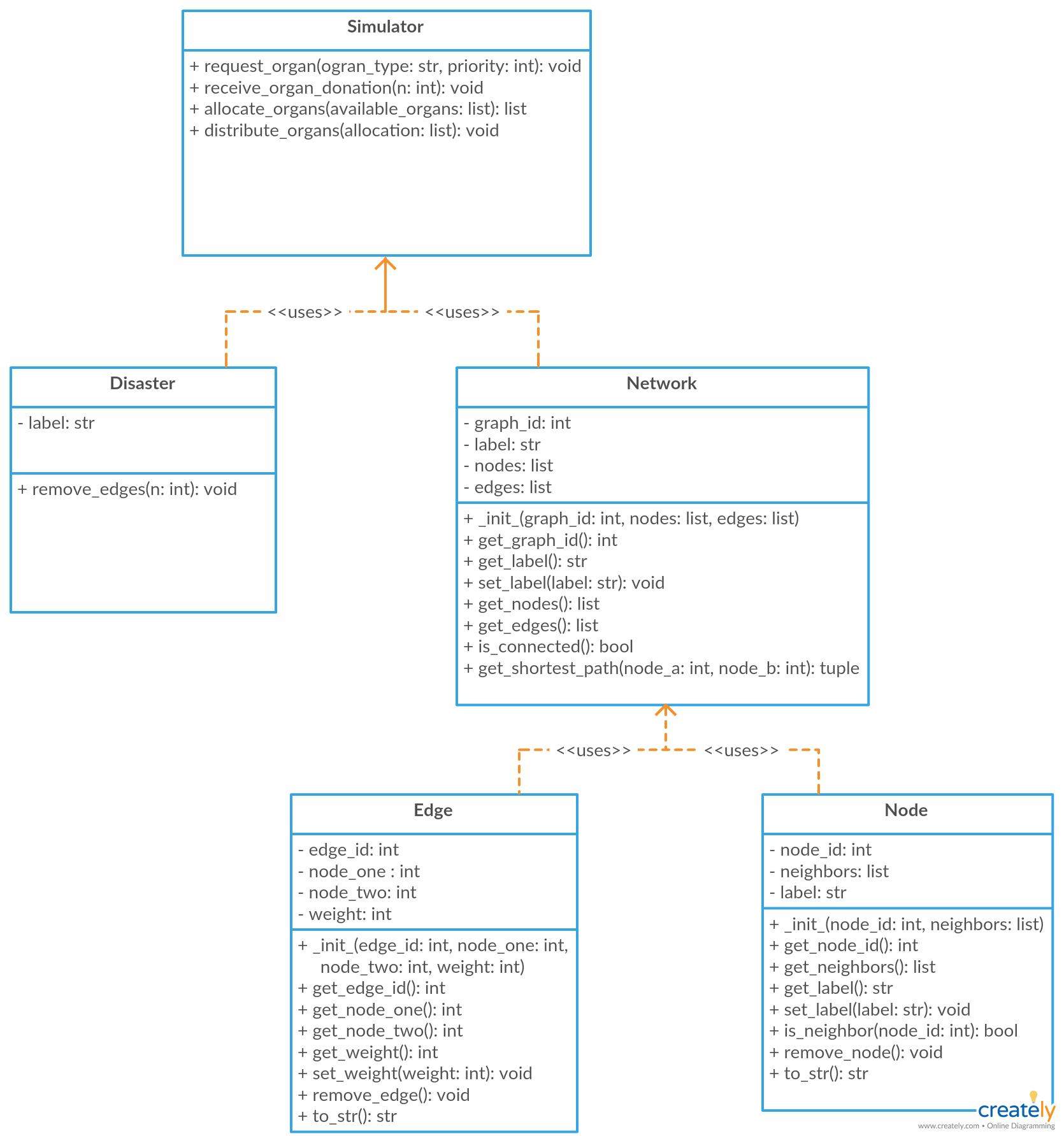
# **Introduction**

This project is designed to simulate an organ transplant system. Its aim is to simulate potential problems that may occur while transporting organs from facility to facility. Factors that could affect the system include: the number of vehicles available in circulation, construction (closed roads), the amount of time an organ remains viable out of the human body, traffic, and natural disasters. For the purposes of this simulated model, problems during transport will be represented by removing paths between two given nodes. The system will handle these problems by finding alternative routes between two destinations. Our system will be implemented using the Python programming language.

# **Scope**

This system will model a minimum of fifty nodes that will be connected by edges, thus creating the network. Nodes and edges have the possibility of being added or removed. The network will be tested within the program to ensure that the graph is connected after the removal of edges. So long as there is a path between two destinations, the shortest path will be determined. This system will take parallel edges into consideration. This system is not meant to be applied to a hospital network in real time. This is only meant to model the behavior of a network.

# **UML Diagram**



# **Timeline**

# **Stakeholders**

[not sure what to do here… we can remove stakeholders if need be]

# **Functional Requirements**

## Nodes

Each node will represent a hospital which is independent from the other hospitals throughout the network.

### Adding Nodes

The system must be capable of adding nodes.

### Removing Nodes

The system must be capable of removing nodes.

### Minimum Number of Nodes

The system must be able to represent a network with a minimum of at least 50 nodes.

## Edges

Each edge represents a path between two given nodes (hospitals). These edges will represent any given mode of transportation between two nodes such as travel via vehicle, helicopter, airplane, etc.

### Adding Edges

The system must be capable of adding edges between two given nodes.

### Removing Edges

The system must be capable of removing edges between two given nodes.

## Network Connectivity

After an event (disaster) occurs, the simulator must determine if the network is still connected.

## Shortest Path

The system must be able to determine the shortest path between any two given nodes.

### Parallel Edges

Parallel edges must be taken into consideration and handled accordingly when determining the shortest path.

## Prioritized Distribution

The system must be able to disassemble a package (organ donation) and prioritize the distribution of the contents across the network.

## Distribution of Multiple Packages

The system must be able to handle multiple packages concurrently.

# **Non-Functional Requirements**

## Performance

The system shall complete the simulation within 250,000 ms.

## Operating Constraints

Execution will require Python 3.6 or later, a processor speed of 1.2GHz or greater, 2GB of memory or greater, and 10MB of storage or greater.

## Platform Constraints

Any version of Windows, MacOS, or Linux distribution compatible with Python 3.6 or later.

## Usability

System should be easy to use. Users should be able to grasp how to interact with the system within 30 minutes.