# COVID-19 Case Simulation with Lockdown Effects

**Overview**

This project was meticulously developed as part of the CECS 274 course, with the primary goal of simulating the dynamics of COVID-19 cases under varying conditions of lockdown. The simulation allows users to customize parameters or use preconfigured default settings, offering a sophisticated and insightful analysis of the pandemic's spread.

## Simulation Class

The Simulation Class serves as the foundation for the entire project, responsible for initializing parameters and managing the simulation process.

### Method: __init__(parameters)

This method handles the initialization of simulation parameters based on user input or default settings. Additionally, constants are defined to represent different states of the Nodes for enhanced readability. An ArrayList named "Nodes" is created to efficiently store essential "Node" objects, and some Nodes are randomly infected based on the INITIAL_SETTING.

### Method: getInteractionGraph()

This method constructs the Interaction Graph, which captures the complex interactions between Nodes during the simulation. An AdjacencyList is used to store these interactions, and random node selections with ALPHA probability create realistic interactions akin to those during the COVID-19 pandemic.

## Simulation Execution

The simulation is executed through the "run" function, which progresses the simulation day by day and provides visual representations.

### Method: run(days)

This method runs the simulation for a specified number of days, presenting daily statistics and generating a graph of the cases using the matplotlib library.

- The simulation evaluates interactions between INFECTED and CLEAN nodes based on the Interaction Graph, simulating the spread of the virus.
- The recovery process for infected nodes is managed, with each day reducing the number of days left to recover. Once the recovery period is over, there is a calculated FATALITY_RATE chance for an infected node to transition to the deceased state.
- Users have the flexibility to observe the simulation for any desired number of days, allowing them to gain insights into the pandemic's development.

**Acknowledgments**

The project owes its foundation to the invaluable contributions of Dr. Morales Ponce, who provided the Interfaces.py file and the essential skeletal framework for data structure implementations. With these foundational elements in place, the project seamlessly integrates data structures to simulate the intricate dynamics of COVID-19 cases.

The core of the project revolves around data structures such as the ArrayList, which efficiently stores and manages the crucial "Node" objects representing different states and interactions in the simulation. Additionally, the clever use of an AdjacencyList captures and organizes the complex relationships between Nodes, enabling the faithful replication of real-world interactions during the COVID-19 pandemic.

By thoughtfully implementing data structures, the project demonstrates an in-depth understanding of their functionality and how they can be effectively employed to tackle complex real-world scenarios. The project showcases the application of data structures in managing parameters, handling interactions, and presenting simulation outcomes, effectively bridging the gap between theoretical knowledge and practical problem-solving.