In Google Collaboratory, we implemented two simulation scenarios as following:

**The first scenario**: This scenario uses the proposed incentive mechanism to motivate the FNvs for participation. We implement this scenario using Stackelberg game theory where there is a leader that interacts with a group of followers.

We implement this scenario as following:

* The leader is the FSN that receives the offloading request, and the followers are the FNvs that are registered in the iVFC system and are available at the time of the offloading request.
* The leader offers a certain reputation for the FNvs, and the followers have their own reputation scores. The goal is to find the optimal combination of reward and penalty values that maximize the number of followers that accept the leader's offer.
* We assume that the number of the available FNvs at the time of the offloading request are **m** and the number of the TNvs needed to execute the offloaded task i are **ni**, where n can be set as required based on the computation resources needed for processing the offloaded task.
* We assume that the leader offers a reputation value in range (1.0, 6.0) as a reward for the FNv that accepts to participate in the task execution process. We also assume theta the leader offers a reputation value in range (-1.0, 0.0) as a penalty for the FNv that rejects to participate.
* We define the desired number of followers to execute the task as the optimal participants, where the optimal participants = ni.
* We define the maximum number of iterations the model will run as 100 times. The code initializes the number of participants to 0 and creates empty lists to store the results of rewards, penalties, number of accepts, and behaviours (i.e., the behaviours are the reactions of the FNvs in response to the offered reputation, behaviours are: accept or reject).
* A loop is set up to iterate max\_iterations times or until the optimal number of participants is reached. Inside the loop, a new reward and penalty score for the leader are randomly generated within the ranges specified in the fourth step.
* The behaviour of each follower is determined based on their current reputation score and the offered reward/penalty values. The behaviour is stored in the behaviours list, and the updated reputation scores are stored in the new\_scores list.
* After the loop, the results (reward, penalty, and number of accepts) for each iteration are appended to their respective lists (rewards, penalties, num\_accepts).
* If the optimal number of participants is reached (i.e., the number of accepts is greater than or equal to optimal\_participants) or the final iteration reached, the loop breaks and the optimal reward and penalty values are printed.

In summary, the code runs a simulation to find the optimal reward and penalty values that maximize the number of followers accepting the leader's offer.

**The second scenario:** This scenario is running without using the proposed incentive mechanism. In Google Collaboratory, we implement another scenario without using rewards and penalties to incentive the FNvs to participate. Instead, the followers' acceptance is solely based on their reputation score, where a score of 4-6 represents acceptance, a score of less than 4 and greater than 0 represents rejection. The followers with different reputation scores make decisions (accept or reject) based on their scores. The goal is to find the optimal number of participants by iterating until the desired number is reached or a maximum number of iterations is reached. Depending on the old score, the behaviour of the FNvs is determined as follows:

* If 4 <= old score <= 6, the behaviour is set to "accept".
* If 0 <= old score < 4, the behaviour is set to "reject".

We assume the number of the available FNvs at the time of the offloading request are m and the number of TNvs needed to execute the offloaded task i are ni, where m and n can be set as required.

We define the desired number of followers to execute the task as the optimal-participants. We define the maximum number of iterations the model will run as 100 times.

In summary, the code runs a simulation to find the optimal number of followers accepting to participate.