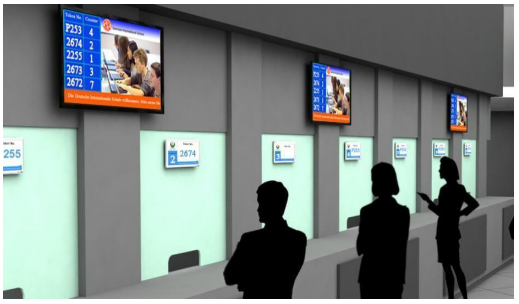
Queue Management System

Abstract

We encounter many queues in our day to day lives at banks, hospitals, restaurants, super markets, movie theaters, etc. These queues often get chaotic, inefficient, frustrating for the customer, and even decrease the number of customers the business can serve. Here, we are seeking to address the randomness by developing a discrete event simulation model. A discrete-event simulation (DES) models the operation of a system as a (discrete) sequence of events in time. Obtaining data about random events like a queue enables us to make more accurate predictions and give the user a smooth experience, hence we seek to collect statistics in our project about the queue parameters.



Introduction

In this project, taking the example of a bank, we seek to build a queue management system which makes an efficient and frustration-free method to manage queues. The users can enter a queue on this system with a GUI, get the estimated waiting time, and later get the service from the teller on their turn. The system then records the waiting time of the customer and updates the statistics for future estimations. In our implementation, we are taking the case where there are two tellers, and one common queue (called a M/M/2 queueing system).

For obtaining the statistics for multiple iterations of the simulation, we get randomized interarrival times between customers and have made the following assumptions:
When a customer enters the bank and both tellers are idle, they choose either one with equal probabilities. If a customer enters the bank and there are four people waiting in the line, they will leave the bank with probability 50%. If a customer enters the bank and there are five or more people waiting in the line, they will leave the bank with probability 60%.

System Requirements

Here, the main components of the system are,

• **State**: Variables that describe the state of the system at any time. The state functions over time vary stepwise in this case.
• The state variables here would be: Number-of-Customers-in-the-Queue (an integer from 0

to n) and Teller-Status (busy or idle).
• **Clock**: Keeps track of the time in the live system.
• **Event tracker**: It tracks the previous events and contains the list of next events based on previous events.
• The system events here would be Customer-Arrival and Customer-Departure.
• **Statistics**: Variables store statistical data about the user, like the time spent in line. This data is used in further uses of the program to predict the expected time.

GUI

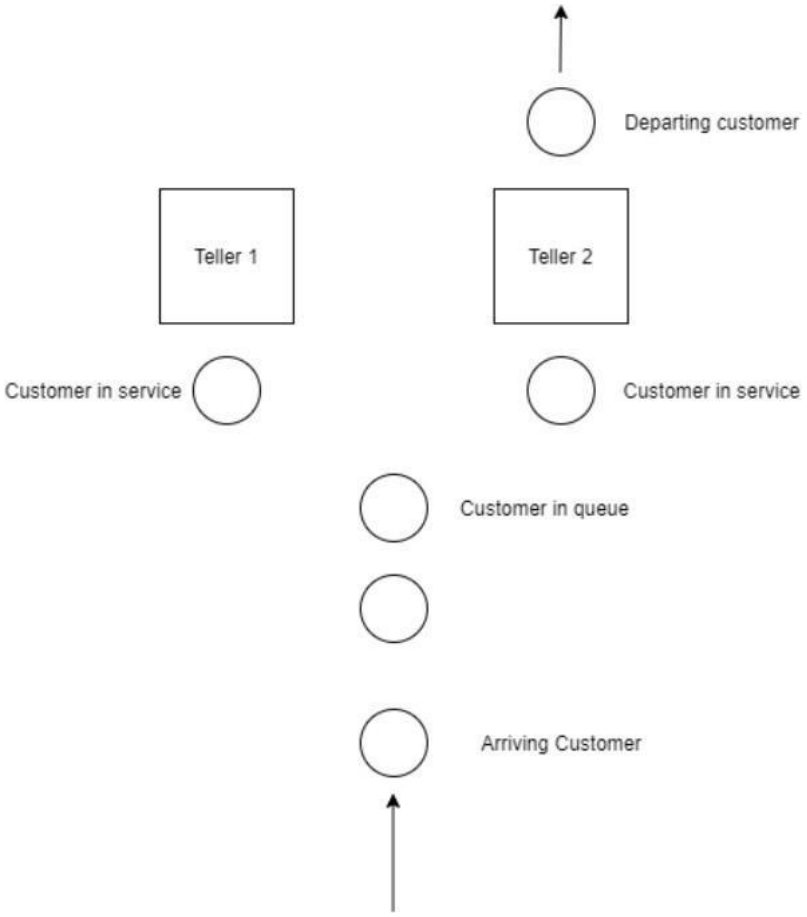
The following are the features of the GUI of this system,

• At the first screen, the user sees the current length of the queue and estimated time of waiting, and a button to join the queue.
• After clicking the join button, the customer is added to the queue, the position in the queue, and the expected time to get to the teller is shown.
• There is also a button to exit the queue at any time.
• If a person ahead in the line exits the queue before getting to the teller, the queue length would be decreased by one and new expected time of waiting will be shown.
• When customers get to the teller, they will be shown a message like “It’s turn of user (user number)”.

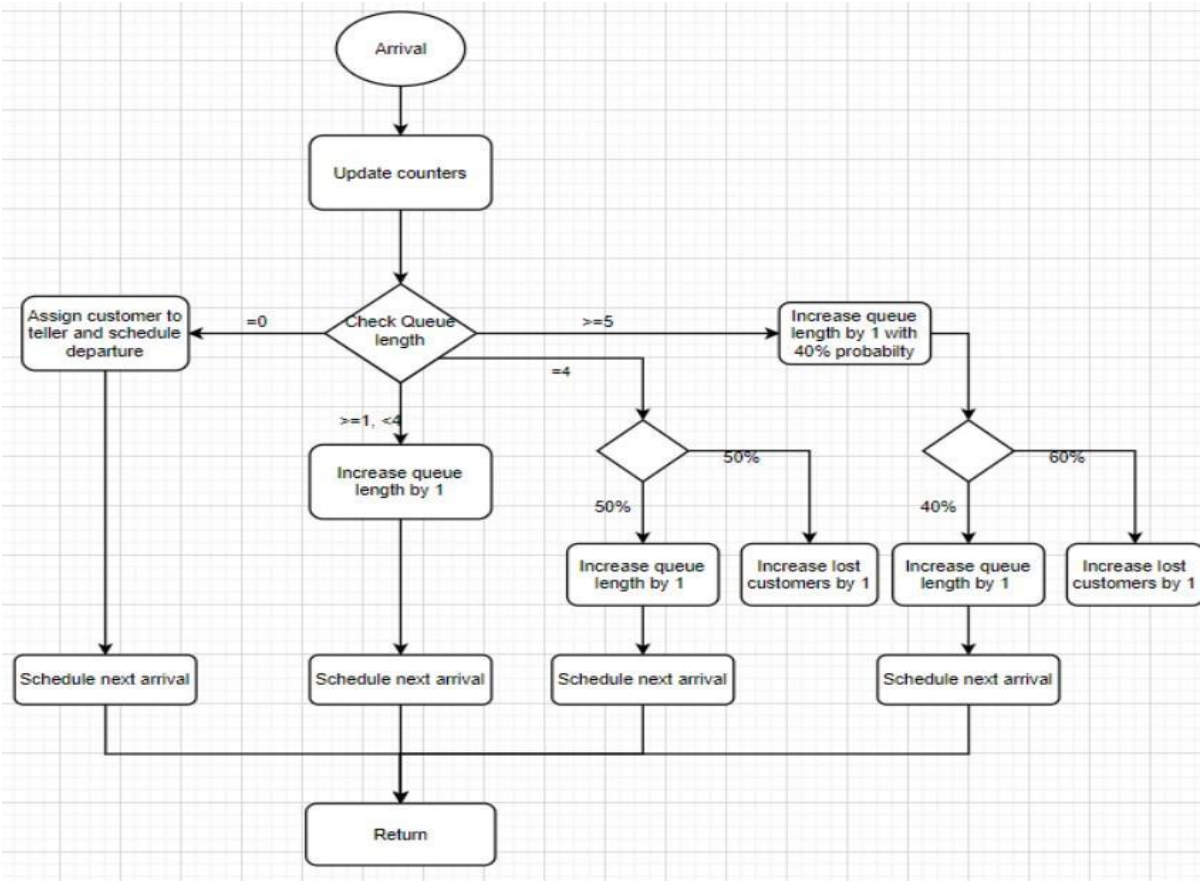
We will be using the queue data structure mainly to represent the queue. We will be mainly using the library **numpy** to build the backend, and **pandas** to store, analyze and display the statistics like estimated time of waiting, average interarrival time, lost customers, average service time of each teller. We will be building the GUI using the **tkinter** library

Flowcharts

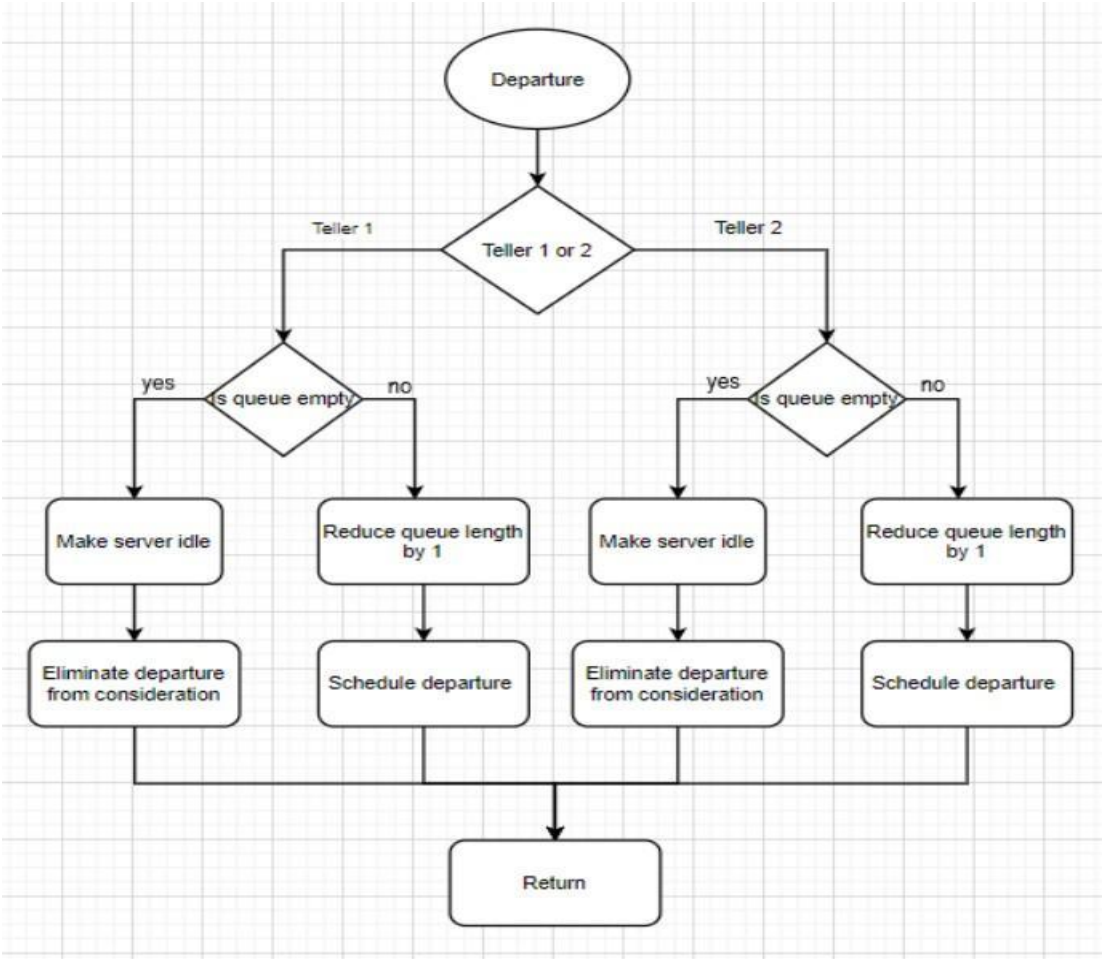
This is an overview of our system,



Arrival Event

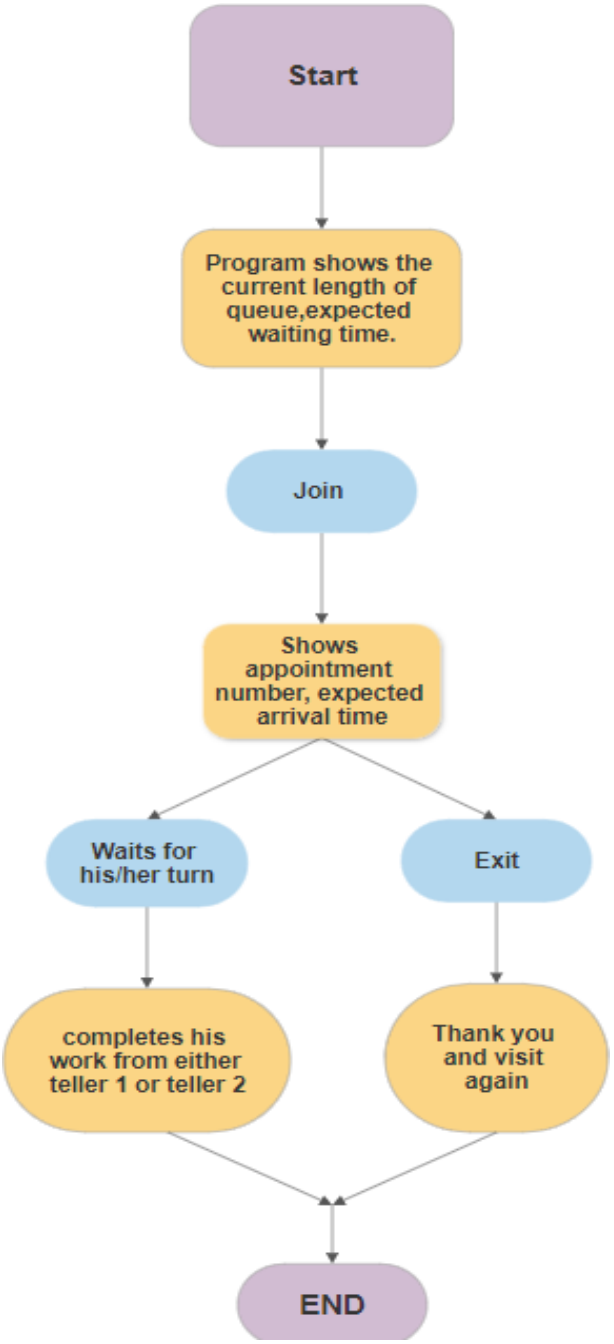


Departure Event



Demonstration

The control flow of the program is shown in the flowchart below. Blue colored boxes indicate choices or buttons, orange colored boxes give the basic function at that step.

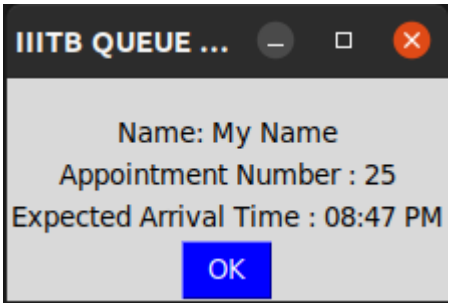


The above screen is visible to the user at the beginning

The user can check the expected wait time for him/her and the number of people present in the queue already. The customer can thus decide whether to join the queue to receive the service or not.

The user should enter his/her name in the entry box given and hit the “JOIN THE QUEUE” button.

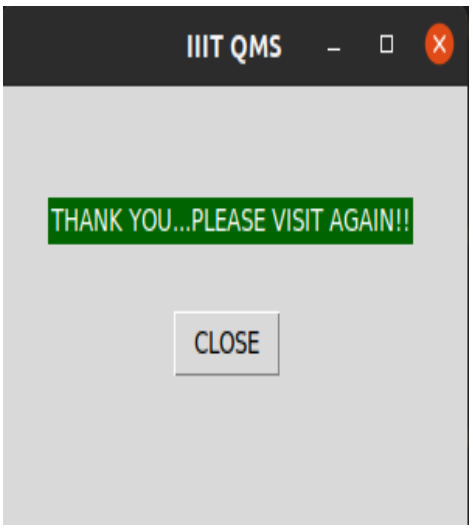
This will a window showing the users
name, appointment number
and the expected time when he/she can avail the service.



Then comes the CANCELLATION window :(

The cancellation window will display the user's name, his/her appointment number, present queue length and expected arrival time.

An "EXIT" button will get displayed, which on assertion removes the customer from the queue and the last window pops up:



Future Aspects

We can expand the system to include cases where the services provided by different counters are different and the customer aims to get to a specific counter.

We can give the customers priority in the queue.

We can allot customers according to the time they are going to be taking at the counter, for example someone might have a work of a few seconds and another might have a work of a long duration.

References

https://en.wikipedia.org/wiki/Discrete-event_simulation
https://en.wikipedia.org/wiki/M/M/c_queue
https://en.wikipedia.org/wiki/Queueing_theory
https://numpy.org/
https://pandas.pydata.org/ • https://docs.python.org/3/library/tkinter.html

Github Repository of the project

https://github.com/born2win685/QUEUE-MANAGEMENT-SYSTEM

CONTRIBUTORS :

Sathiya :

Arrival(queue isn’t empty),GUI(functional part).

Ishaan :

Timing Routines, GUI(Aesthetic part).

Saketh :

Arrival(queue is empty), Departure, GUI(Improvisation , adding backgrounds, ordering).

Dheeraj :

Departure event

Pranav :

Most of the backend, Backend-frontend interface, User methods for class, data frames.