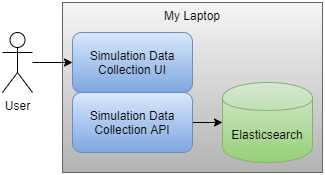
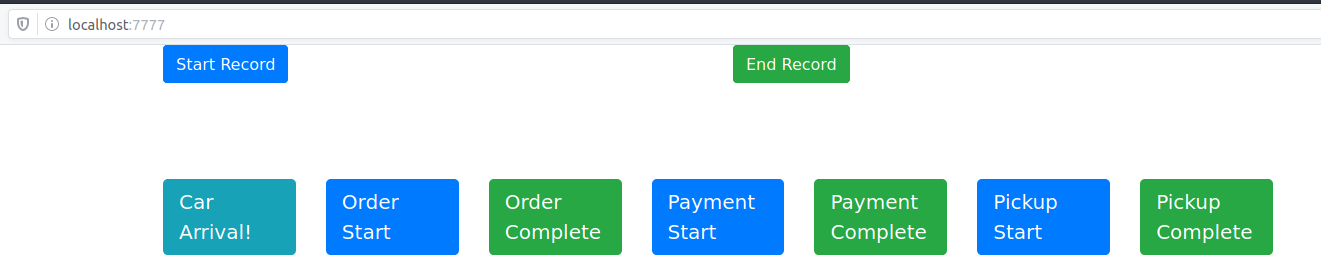
# Data Collection

As I was performing this alone and time was critical, I knew that recording information needed to be streamlined. I developed a system for recording the events that could be stored within a document store (Elasticsearch) for later extraction into data sets for submissions.



The allowed me to find a location where I could see traffic entering the parking lot, the order station, and both windows. The UI with my laptops touch screen meant I could focus on observing without my efforts to record interfering, simply tapping when each event was spotted.



# Software

All aspects of this project were developed and stored to Github:

<https://github.com/DigitalGoetz/SimProject2>

The mechanisms support the above collection capabilities can be found within my projects github repository under the “gathering” directory. Instructions for building/usage can be found within the notes.txt within. Analysis of the data can be found through the scripts and data files within the “analysis” directory. This includes the pruned and curated data sets from our class. Lastly, the derived model for the restaurant is stored within the “model” directory.

# Arrivals

## Arrival Data Preparation

* Threw out data sets in XLSX files (just to save the time spent converting into CSV files)
* Arrival Data Curation
  + Cleaned data so that each source contained only the listing of timestamps (for easier programmatic iterations through each file’s content
  + Some data sets contained multiple columns of data the appeared to have a potential gap
    - Broken into separate files
    - Ex: Hill\_Somayire data set
  + Made some assumptions above the data represented
    - Ex: Jasinski data set contained values formatted to what looked like HH:MM:SS, but the content was more likely MM:SS:00 (unless they monitored the restaurant over its slowest 15 hour timeframe!). For these I just converted the MM:SS values to HH:MM:SS to match expected format
  + Excluded some data sets based on their time for recording arrivals
    - Ex: The Richards\_Teeters and Wical data sets only contained a handful of arrival times within their data sets compared to the normal 15-20.
    - Ex: Removed an entry from Ogletree\_Navuluri data set which contained the mythic 66th second within a minute
  + Within each data set, I use each adjacent timestamp to determine how many minutes between the arrivals. This list of “times between arrivals” are placed within an array and additional statistical properties are recorded for each value

## Arrival Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mean | Median | Std Dev | Min | Max | Source |
| 0.88 | 0.63 | 0.92 | 0.02 | 3.95 | Arrival\_Hill\_Somayire2.csv |
| 0.79 | 0.63 | 0.7 | 0.05 | 3.23 | Arrival\_Young.csv |
| 0.77 | 0.63 | 0.66 | 0.02 | 2.4 | Arrival\_Mitchell\_Gotteti.csv |
| 0.75 | 0.36 | 0.69 | 0.12 | 2.25 | Arrival\_Means.csv |
| 0.96 | 0.47 | 1.34 | 0.02 | 5.1 | Arrival\_Goetz.csv |
| 0.76 | 0.58 | 0.66 | 0.02 | 2.08 | Arrival\_Hill\_Somayire.csv |
| 0.85 | 0.6 | 0.8 | 0.08 | 3.37 | Arrival\_Woods\_Inman.csv |
| 1.04 | 0.6 | 0.99 | 0.07 | 3 | Arrival\_OgletreeNavuluri.csv |
| 1.67 | 0.97 | 2.03 | 0.08 | 10.55 | Arrival\_Read.csv |
| 0.83 | 0.57 | 0.73 | 0.03 | 2.73 | Arrival\_Marcum.csv |
| 1.1 | 1.02 | 0.81 | 0.33 | 3.73 | Arrival\_Hall.csv |
| 0.66 | 0.49 | 0.55 | 0 | 2.3 | Arrival\_Pettit.csv |
| 0.8 | 0.47 | 0.78 | 0.02 | 3.03 | Arrival\_Cheerala |
| 0.65 | 0.47 | 0.52 | 0.07 | 1.82 | Arrival\_Jasinski.csv |
| 0.94 | 0.33 | 1.15 | 0.02 | 3.58 | Arrival\_Mannam\_Susarapu.csv |

The remaining data has resulted in a mostly consistent collection of data sets. There are some outliers (i.e., Pettit’s instantaneous arrival time and Read’s nearly 11-minute gap in arrivals), but the overall collection appears safe to proceed with combining these into the overall arrival dataset.

When viewing the overall dataset, the 0 second arrival time isn’t so far-fetched (as there are many entries with similar values.) However, the 0 wreaks havok with some of the numpy functions. The singular 10.55-minute entry from the Read data set, however, is clearly an outlier as the nearest values are in the area of 4 to 5 minutes. To remedy these issues, I’m removing the timestamps from the Pettit and Read data sets that cause these discrepancies, as the remaining data set appears sound. Assuming the two identical timestamps in the Pettit data set are for the same arrival, I will simply prune the duplicate timestamp. For the Read data set I split it into two sets using the two timestamps that generate the 10.55-minute delay as the pivot.

# Orders

## Order Data Preparation

* Threw out data sets in XLSX files (just to save the time spent converting into CSV files)
* Order Data Curation
  + Cleaned data so that each source contained only the listing of timestamp pairs (for easier programmatic iterations through each file’s content
  + Some data sets contained multiple columns of data the appeared to have a potential gap
    - Broken into separate files
    - Ex: Hill\_Somayire data set
  + Made some assumptions above the data represented
    - Ex: Jasinski data set contained value pairs indicating what appears to be timestamp 0:0:0 to MM:SS:00. I’m maintaining the assumption that these indicate that each customer arrives at relative time 0:0:0 and completes their order at the second value. Converting second column to match HH:MM:SS and keeping the values as they appear consistent with other data set’s times.
  + Excluded some data sets completely had unclear data
    - Ex: The Ogletree\_Navuluri and Marcum data sets contain starting times that occur after completions, which results in negative order times. Fairly certain something went awry during the recording of these values.
    - Ex: The Means data set contains a list of numbers that might represent the exact moment that payment occurred, but I’m omitting the data set for the sake of timeliness
    - Ex: The Rupert data set contains a listing of timestamps that may be a row with an arrival timestamp, followed by a row with a completion timestamp, but the data granularity is by minutes and not seconds. This will have many orders taking 0 time within my analysis.

## Order Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mean | Median | Std Dev | Minimum | Maximum | Source |
| 0.57 | 0.53 | 0.37 | 0.08 | 1.2 | Order\_Goetz.csv |
| 0.45 | 0.37 | 0.27 | 0.22 | 0.98 | Order\_RIchardsTeeters.csv |
| 2.17 | 2 | 0.99 | 0.7 | 4.07 | Order\_Cheerala |
| 0.73 | 0.67 | 0.4 | 0.13 | 1.65 | Order\_Hill\_Somayire.csv |
| 0.46 | 0.43 | 0.3 | 0.08 | 1.73 | Order\_Pettit.csv |
| 2.1 | 2.15 | 1 | 0.67 | 3.37 | Order\_Jasinski.csv |
| 0.56 | 0.45 | 0.29 | 0.17 | 1.08 | Order\_Mitchell\_Gotteti.csv |
| 0.72 | 0.6 | 0.39 | 0.18 | 1.77 | Order\_Woods\_Inman.csv |
| 0.86 | 0.48 | 0.98 | 0.02 | 4.6 | Order\_Mannam\_Susarapu.csv |
| 0.86 | 0.68 | 0.57 | 0.25 | 2.07 | Order\_Wical - Sheet1.csv |
| 0.51 | 0.38 | 0.31 | 0.2 | 1.23 | Order\_Hall.csv |
| 0.64 | 0.59 | 0.32 | 0.27 | 1.45 | Order\_Hill\_Somayire2.csv |
| 0.91 | 0.63 | 0.66 | 0.17 | 2.25 | Order\_Read.csv |
| 0.6 | 0.47 | 0.46 | 0.1 | 2.45 | Order\_Young.csv |

# Payments

## Payment Data Preparation

* Threw out data sets in XLSX files (just to save the time spent converting into CSV files)
* Payment Data Curation
  + Cleaned data so that each source contained only the listing of timestamp pairs (for easier programmatic iterations through each file’s content
  + Some data sets had trailing timestamp pairs (like within my data set!) where the experimental observation completed before the customer finished ordering. I’m curious why all other data sets had completed pairs. Did they wait for all stations to completely empty out? Did they stop recording new arrivals for each station after a certain time to ensure they had completed pairs? Either way, I’ve removed my unmatched pairs.
  + Some data sets contained multiple columns of data the appeared to have a potential gap
    - Broken into separate files
    - Ex: Hill\_Somayire data set
  + Made some assumptions above the data represented
    - Ex: Jasinski data set contained value pairs indicating what appears to be timestamp 0:0:0 to MM:SS:00. I’m maintaining the assumption that these indicate that each customer arrives at relative time 0:0:0 and completes their payment at the second value. Converting second column to match HH:MM:SS and keeping the values as they appear consistent with other data set’s times.
  + Excluded some data sets completely had unclear data
    - Ex: The Means data set contains a list of numbers that might represent the exact moment that payment occurred, but I’m omitting the data set for the sake of timeliness
    - Ex: The Rupert data set contains a listing of timestamps that may be a row with an arrival timestamp, followed by a row with a completion timestamp, but the data granularity is by minutes and not seconds. This will have many orders taking 0 time within my analysis.
    - Ex: Some timestamps in the Woods\_Inman data set followed an HH:MM:SS format, while others only appeared to be HH:MM (or perhaps MM:SS?) either way, the unclear determination rendered this data set overly complicated to curate.
    - Ex: The Mannam\_Susarapu data set contains starting times that occur after completions, which results in negative order times. Fairly certain something went awry during the recording of these values.

## Payment Data

| Mean | Median | Std Dev | Minimum | Maximum | Source |
| --- | --- | --- | --- | --- | --- |
| 1.91 | 2.02 | 0.71 | 0.83 | 3.15 | Payment\_Hall.csv |
| 0.61 | 0.52 | 0.41 | 0.22 | 2.53 | Payment\_Young.csv |
| 0.51 | 0.5 | 0.32 | 0.12 | 1.53 | Payment\_Marcum.csv |
| 0.57 | 0.53 | 0.25 | 0.18 | 1.18 | Payment\_Hill\_Somayire2.csv |
| 0.73 | 0.58 | 0.49 | 0.22 | 2.25 | Payment\_Read.csv |
| 0.72 | 0.65 | 0.35 | 0.27 | 1.7 | Payment\_OgletreeNavuluri.csv |
| 0.55 | 0.54 | 0.26 | 0.18 | 1.27 | Payment\_Hill\_Somayire.csv |
| 0.48 | 0.45 | 0.27 | 0.15 | 1.37 | Payment\_Wical - Sheet1.csv |
| 0.99 | 0.95 | 0.53 | 0.33 | 2.45 | Payment\_Jasinski.csv |
| 0.54 | 0.43 | 0.36 | 0.15 | 1.7 | Payment\_Mitchell\_Gotteti.csv |
| 0.74 | 0.69 | 0.25 | 0.38 | 1.25 | Payment\_Goetz.csv |
| 0.38 | 0.28 | 0.28 | 0.05 | 1.38 | Payment\_Pettit.csv |
| 0.49 | 0.32 | 0.29 | 0.25 | 1.03 | Payment\_RichardsTeeters.csv |
| 2.44 | 2.4 | 1.09 | 0.95 | 4.7 | Payment\_Cheerala |

# Pickup

## Pickup Data Preparation

* Threw out data sets in XLSX files (just to save the time spent converting into CSV files)
* Pickup Data Curation
  + Cleaned data so that each source contained only the listing of timestamp pairs (for easier programmatic iterations through each file’s content
  + Some data sets had trailing timestamp pairs (again, this was found in my data set) where the experimental observation completed before the customer finished ordering. I’ve removed my unmatched pairs from this data set.
  + Some data sets contained multiple columns of data the appeared to have a potential gap
    - Broken into separate files
    - Ex: Hill\_Somayire data set
  + Made some assumptions above the data represented
    - Ex: Jasinski data set contained value pairs indicating what appears to be timestamp 0:0:0 to MM:SS:00. I’m maintaining the assumption that these indicate that each customer arrives at relative time 0:0:0 and completes their pickup at the second value. Converting second column to match HH:MM:SS and keeping the values as they appear consistent with other data set’s times.
  + Excluded some data sets completely had unclear data
    - Ex: The Ogletree\_Navuluri data set contains starting times that occur after completions, which results in negative order times. Fairly certain something went awry during the recording of these values.
    - Ex: The Means data set contains a list of numbers that might represent the exact moment that pickup occurred, but I’m omitting the data set for the sake of timeliness
    - Ex: The Rupert data set contains a listing of timestamps that may be a row with an arrival timestamp, followed by a row with a completion timestamp, but the data granularity is by minutes and not seconds. This will have many orders taking 0 time within my analysis.
    - Ex: Some timestamps in the Woods\_Inman data set followed an HH:MM:SS format, while others only appeared to be HH:MM (or perhaps MM:SS?) either way, the unclear determination rendered this data set overly complicated to curate.

## Pickup Data

| Mean | Median | Std Dev | Minimum | Maximum | Source |
| --- | --- | --- | --- | --- | --- |
| 0.74 | 0.42 | 0.65 | 0.15 | 2.4 | Pickup\_Hill\_Somayire.csv |
| 0.69 | 0.48 | 0.69 | 0.03 | 3 | Pickup\_Mannam\_Susarapu.csv |
| 0.63 | 0.3 | 0.7 | 0.03 | 2.52 | Pickup\_Pettit.csv |
| 1.5 | 1.13 | 0.89 | 0.4 | 2.83 | Pickup\_RichardsTeeters.csv |
| 0.45 | 0.33 | 0.25 | 0.12 | 0.9 | Pickup\_Mitchell\_Gotteti.csv |
| 3.21 | 3.26 | 0.66 | 2.13 | 4.13 | Pickup\_Hall.csv |
| 0.8 | 0.73 | 0.32 | 0.28 | 1.45 | Pickup\_Jasinski.csv |
| 0.85 | 0.45 | 0.83 | 0.02 | 2.97 | Pickup\_Wical - Sheet1.csv |
| 1.27 | 1.2 | 0.76 | 0.15 | 3.07 | Pickup\_Goetz.csv |
| 1.15 | 0.93 | 0.88 | 0.1 | 2.97 | Pickup\_Cheerala |
| 0.55 | 0.23 | 0.76 | 0.03 | 3.87 | Pickup\_Young.csv |
| 0.44 | 0.28 | 0.29 | 0.13 | 1.13 | Pickup\_Hill\_Somayire2.csv |
| 0.65 | 0.4 | 0.54 | 0.15 | 2.2 | Pickup\_Marcum.csv |
| 1.13 | 0.85 | 1.34 | 0.17 | 5.77 | Pickup\_Read.csv |

# Combined Data Sets

Each event type’s data sets were combined into a single aggregate data set, resulting in the below tables and histograms. Using a variety of bin sizes for each, the respective histograms were selected in hopes of aiding in the identification of an ideal probability distribution(s) for use in the restaurant’s model.

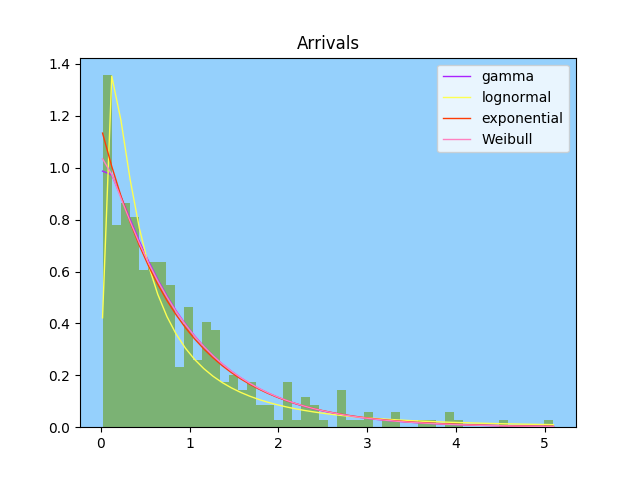
Each of the below histograms are similar in generalized shape, but there are a few distributions that can match these patterns. This includes:

* The Log-Normal Distribution
* The Exponential Distribution
* The Gamma Distribution
* The Weibull Distribution

Each data set is compared against the respective distribution’s theoretical values with respective parameters set to their best fits, or maximum likelihood estimates (MLE) and for each event type, the distribution with the best “goodness of fit” metric (using the Chi-Square method) will be used within the restaurant’s model.

## Arrival Data Set

| Mean | Median | Std Dev | Min | Max |
| --- | --- | --- | --- | --- |
| 0.86 | 0.6 | 0.85 | 0.0 | 5.1 |



The overall shape of the arrivals dataset when evaluated against the 4 distributions results in the Chi-Squared metrics:

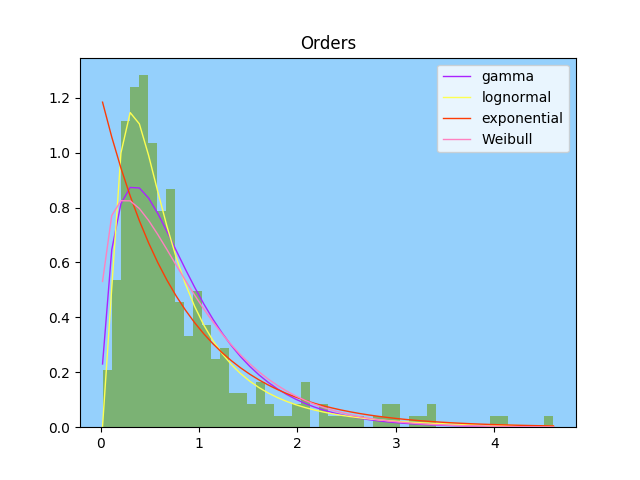
* Gamma 0.0065
* Log Normal 0.0306
* Exponential 0.005
* Weibull 0.0058

From this, we accept that the Exponential distribution most closely fits our observed data set. The exponential parameters to generate this are:

* Location: 0.0
* Scale: 0.862052785924

## Order Data Set

| Mean | Median | Std Dev | Min | Max |
| --- | --- | --- | --- | --- |
| 0.82 | 0.57 | 0.76 | 0.02 | 4.6 |



The overall shape of the orders dataset when evaluated against the 4 distributions results in the Chi-Squared metrics:

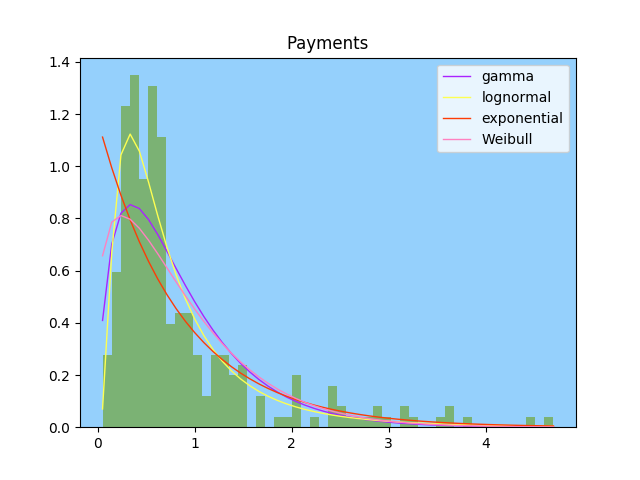
* Gamma 0.0144
* Log Normal 0.0052
* Exponential 0.0417
* Weibull 0.0202

From this, we accept that the Log Normal distribution most closely fits our observed data set. The log normal parameters to generate this are:

* Shape: -0.5137166258214053
* Scale: 0.799858597093

## Payment Data Set

| Mean | Median | Std Dev | Min | Max |
| --- | --- | --- | --- | --- |
| 0.85 | 0.57 | 0.79 | 0.05 | 4.7 |



The overall shape of the payments dataset when evaluated against the 4 distributions results in the Chi-Squared metrics:

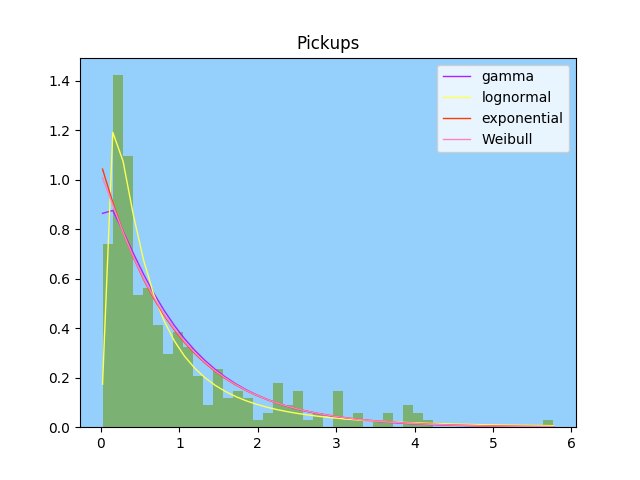
* Gamma 0.0254
* Log Normal 0.0132
* Exponential 0.0455
* Weibull 0.0319

From this, we accept that the Log Normal distribution most closely fits our observed data set. The log normal parameters to generate this are:

* Shape: -0.4903622325760071
* Scale: 0.794724403708

## Pickup Data Set

| Mean | Median | Std Dev | Min | Max |
| --- | --- | --- | --- | --- |
| 0.94 | 0.57 | 0.98 | 0.02 | 5.77 |



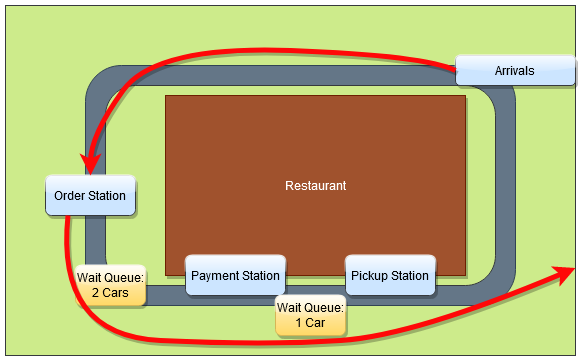
The overall shape of the pickup dataset when evaluated against the 4 distributions results in the Chi-Squared metrics:

* Gamma 0.0132
* Log Normal 0.0134
* Exponential 0.0135
* Weibull 0.0134

From this, we accept that the Gamma distribution most closely fits our observed data set. The Gamma parameters to generate this are:

* Shape: 1.0811523137
* Location: 0.0
* Scale: 0.8677960404637084

# The Restaurant Model



The restaurant modeled with the data obtained through our classes observations differs from the first project in a few key aspects. While we still maintain an arrival queue, we no longer track arrival queue length (nor record any balking behaviors). There are still static ordering stations (only the station count is now 1.) Finally, there is a waiting queue for cars to enter post-ordering prior to reaching the subsequent stations.

The differences between the two systems also include the splitting of the payment-pickup station into two distinct stations both separated by a waiting queue that can containing a maximum of one car. Since balking behaviors are no longer tracked in the data sets, I’m going to assume that no balking occurs. While this is clearly false in reality, I’m still ignoring this detail in my model since I cannot have concrete information identifying such events. Finally, order preparation is not addressed by our recorded data.

Each station’s behaviors are also incomplete regarding start and stop times due to the nature of the observed system and the detail of recorded information. We cannot tell when an order is completed except by when a driver pulls away from the order station. When the queue between order and payment stations is full, the person placing an order may take only a few seconds to order but remain stationary at the order station for minutes awaiting a spot in the waiting queue leading into the payment station. This aspect is an observed behavior of all stations. We can only make assumptions that each stations action is completed when the car leaves that station; no accounting of “completed and awaiting room” in the subsequent queues is considered.

import simpy, numpy, random, time

class Sim(object):

def nextArrival(self):

# Exponential Variable with

# scale: 0.862052785924

return round(numpy.random.exponential(0.862052785924), 4);

def nextOrder(self):

# LogNormal Variable with

# scale: -0.5137166258214053

# shape: 0.799858597093

return round(numpy.random.lognormal(-0.5137166258214053, 0.799858597093), 4)

def nextPayment(self):

# LogNormal Variable with

# scale: -0.4903622325760071

# shape: 0.794724403708

return round(numpy.random.lognormal(-0.4903622325760071, 0.794724403708), 4)

def nextPickup(self):

# Gamma Variable with

# shape: 1.0811523137

# scale: 0.8677960404637084

return round(numpy.random.gamma(1.0811523137, 0.8677960404637084), 4)

def formatTime(self, time):

return "{:.2f}".format(time)

def \_\_init\_\_(self, env):

self.env = env

self.orderstation = simpy.Resource(self.env, capacity=1)

self.paymentqueue = simpy.Resource(self.env, capacity=2)

self.paymentstation = simpy.Resource(self.env, capacity=1)

self.pickupqueue = simpy.Resource(self.env, capacity=1)

self.pickupstation = simpy.Resource(self.env, capacity=1)

self.numberserved = 0

self.customerId = 0

def arrivals(self):

while True:

print(self.formatTime(self.env.now) + "\tCustomer " + str(self.customerId) + " arrives.")

c = self.customer(self.customerId)

self.customerId = self.customerId + 1

self.env.process(c)

yield self.env.timeout(self.nextArrival())

def customer(self, id):

with self.orderstation.request() as oReq:

yield oReq

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " starts ordering.")

yield self.env.timeout(self.nextOrder())

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " completes ordering.")

with self.paymentqueue.request() as payQReq:

yield payQReq

with self.paymentstation.request() as payReq:

yield payReq

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " starts paying.")

yield self.env.timeout(self.nextPayment())

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " completes paying.")

with self.pickupqueue.request() as pickQReq:

yield pickQReq

with self.pickupstation.request() as pickReq:

yield pickReq

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " starts pickup.")

yield self.env.timeout(self.nextPickup())

print(self.formatTime(self.env.now) + "\tCustomer " + str(id) + " completes pickup.")

self.numberserved = self.numberserved + 1

def report(self):

return self.numberserved

def run(minutesToRun):

random.seed(int(round(time.time() \* 1000)))

env = simpy.Environment()

env.process(Sim(env).arrivals())

env.run(until=minutesToRun)

run(60)