Amusement Park Simulation

Abdullah (200041126), Amina (200041155)
Department of Computer Science and Engineering
Islamic University of Technology
Board Bazar, Gazipur

I. Problem Statement

Develop an amusement park simulation model that accurately represents the dynamics of visitor movement, ticket sales, and system operations. The simulation model should incorporate the following aspects:

- Visitor Behaviour:
  - Simulate the arrival and departure of visitors
  - Model visitor movement patterns and route choices within the park.
- Ticketing System:
  - Replicate the ticket sales process
  - Capture ticket sales data by counter location
- Park Systems:
  - Park Management System: Oversees overall park operations and resource allocation.
  - Ticket Counter Systems: Manage ticket sales and visitor entry, including:
    - Entrance Ticket Counter System
  - Ride Systems: Simulate the operation and capacity of various rides, including:
    - Roller Coaster System
    - Carousel System
    - Ferris Wheel System
    - Cable Car System
    - Water Slide System
  - General Systems:
    - Route System: Represents the layout of paths and visitations within the park.
    - Food System: Models the availability and consumption of food and beverages throughout the park.

The simulation model should be used to analyse the following performance metrics:

- Visitor Flow:
  - Average waiting times for rides.
  - Visitor density in different areas of the park.
- Ticketing System Efficiency:
  - Revenue generated from ticket sales.
  - Impact of counter location on ticket sales.
- Park System Performance:
  - Utilization rates for rides and attractions.
  - Impact of visitor behaviour on ride wait times and system congestion.
  - Effectiveness of resource allocation strategies.

The simulation results should be used to identify potential bottlenecks, optimize resource allocation, and improve overall park operations.

II. Conceptual Model

A. *Ticket Counter Systems:*

A.1 Entrance Ticket Counter System

i) State Variables:
- Status, s_i
- Queue Length, q_i
  where, i = 0, 1, 2, 3 and N = 4

ii) Input Variables:
- Departure Mean, dm_i *(exponential random variate)*
  where, i = 0, 1, 2, 3 and N = 4

iii) Events:
- Receive Event *(arrival of a visitors)*
- Departure Event *(departure of a visitors)*
- Jockey Event *(jockey of a visitor to another queue)*

iv) Output Variables:
- Total Waiting Time, wt_{total}
- Average Waiting Time, wt_{avg}
- Total Visitor-Queue Time, vqt_{total}
- Average Visitor-Queue Time, vqt_{avg}
- Total Visitor-System Time, vst_{total}
- Average Visitor-System Time, vst_{avg}

B. *General Systems:*

B.1 Route System

i) State Variables:
- Status, s *(number of visitors on the route)*

ii) Input Variables:
- Duration Mean, dm *(exponential random variate)*

iii) Events:
- Route Proceed Event *(starting the route to particular point for a visitor)*
- Route End Event *(end of the route to particular point for a visitor)*

iv) Output Variables:
- Total Route Time, rt_{total} *(per route)*
- Average Route Time, rt_{avg} *(per route)*

B.2 Food System

i) State Variables:
- Status, s
- Queue Length, q

ii) Input Variables:

- ○ Departure Mean, dm *(exponential random variate)*

iii) Events:
- ○ Receive Event *(arrival of a visitors)*
- ○ Departure Event *(departure of a visitors)*

iv) Output Variables:
- ○ Total Waiting Time, wt_{total}
- ○ Average Waiting Time, wt_{avg}
- ○ Total Visitor-Queue Time, vqt_{total}
- ○ Average Visitor-Queue Time, vqt_{avg}
- ○ Total Visitor-System Time, vst_{total}
- ○ Average Visitor-System Time, vst_{avg}

C. Ride Systems:

C.1 Roller Coaster System, Carousel System

i) State Variables:
- ○ Status, s
- ○ Queue Length, q

ii) Input Variables:
- ○ Service Time, st *(fixed time duration)*
- ○ Waiting Mean, wm *(exponential random variate)*

iii) Events:
- ○ Receive Event *(arrival of a visitors)*
- ○ Departure Event *(departure of a visitors)*
- ○ Load Event *(wait for loading the ride)*

iv) Output Variables:
- ○ Total Waiting Time, wt_{total}
- ○ Average Waiting Time, wt_{avg}
- ○ Total Visitor-Queue Time, vqt_{total}
- ○ Average Visitor-Queue Time, vqt_{avg}
- ○ Total Visitor-System Time, vst_{total}
- ○ Average Visitor-System Time, vst_{avg}

C.2 Ferris Wheel System, Cable Car System

i) State Variables:
- ○ Status, s
- ○ Queue Length, q
- ○ Number of Loaded Carriage, c

ii) Input Variables:
- ○ Inter-carriage Swap Time, it *(fixed time duration)*
- ○ Unloading Mean, um_i *(exponential random variate)* where, i = 0, 1, 2, 3, 4 and N = 5

iii) Events:
- ○ Receive Event *(arrival of a visitors)*
- ○ Load Event *(loading of a carriage)*
- ○ Unload Event *(unloading of a carriage)*

iv) Output Variables:
- ○ Total Waiting Time, wt_{total}
- ○ Average Waiting Time, wt_{avg}
- ○ Total Visitor-Queue Time, vqt_{total}
- ○ Average Visitor-Queue Time, vqt_{avg}
- ○ Total Visitor-System Time, vst_{total}
- ○ Average Visitor-System Time, vst_{avg}

C.3 Water Slide System

i) State Variables:
- ○ Status, s_i
- ○ Queue Length, q_i

where, i = 0, 1, 2, 3, 4 and N = 5

ii) Input Variables:
- ○ Departure Mean, dm_i *(exponential random variate)* where, i = 0, 1, 2, 3, 4 and N = 5

iii) Events:
- ○ Receive Event *(arrival of a visitors)*
- ○ Departure Event *(departure of a visitors)*

iv) Output Variables:
- ○ Total Waiting Time, wt_{total}
- ○ Average Waiting Time, wt_{avg}
- ○ Total Visitor-Queue Time, vqt_{total}
- ○ Average Visitor-Queue Time, vqt_{avg}
- ○ Total Visitor-System Time, vst_{total}
- ○ Average Visitor-System Time, vst_{avg}

III. Specification Model

A. Ticket Counter Systems:

A.1 Entrance Ticket Counter System

i) State Equations:

$$x_i\,(t^+) = \begin{cases} x_i\,(t) == 0 ?\, 1 : x_i\,(t), & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t) == 0 ?\, 0 : x_i\,(t), & arrival\ at\ time\ t, in\ ith\ server \\ x_i\,(t), & otherwise \end{cases}$$

$$q_i\,(t^+) = \begin{cases} x_i\,(t) == 0 ?\, 0 : q_i\,(t) + 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t) == 0 ?\, 0 : q_i\,(t) - 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t), & otherwise \end{cases}$$

where, i = 0, 1, 2, 3

ii) State Space:

$$x_i\,(t) = \{0, 1\} \qquad for\ i = 0, 1, 2, 3$$
$$q_i\,(t) = \{0, 1, 2 \ldots.\} \quad for\ i = 0, 1, 2, 3$$

$$S = \begin{Bmatrix} (0,0), (0,0), (0,0) \ldots \ldots \ldots \ldots.. \\ (1,0), (0,0), (0,0) \ldots \ldots \ldots \ldots \ldots \\ (1,0), (1,0), (0,0) \ldots \ldots \ldots \ldots \ldots \\ \ldots \ldots \ldots \ldots.. \end{Bmatrix}$$

iii) Event Set:
- ○ E = {r, d, j}

iv) Output Equations:
- ○ Total Waiting Time, $wt_{total} = \sum_{i=1}^{n} w_i$
- ○ Average Waiting Time, $\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i$
- ○ Total Service Time, $s = \sum_{i=1}^{n} s_i$
- ○ Average Service Time, $\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$
- ○ Total Visitor-System Time, $vst_{total} = \sum_{i=1}^{n} vs_i$
- ○ Average Visitor-System Time,

$$\overline{vs} = \frac{1}{n} \sum_{i=1}^{n} vs_i$$
$$= \frac{1}{n} \sum_{i}^{n} (w_i + s_i)$$
$$= \frac{1}{n} \sum_{i=1}^{n} w_i + \frac{1}{n} \sum_{i=1}^{n} s_i$$
$$= \overline{w} + \bar{s}$$

B. General Systems:

B.1 Route System

i) State Equations:

$$x(t^{+}) = \{x(t) + 1\ , \quad arrival\ on\ route\ system$$

ii) State Space:

X = {1, 2 , …….}

iii) Event set:

E = {r, d}

iv) Output Equations:

- Total Waiting Time, $\mathrm{wt}_{\mathrm{total}\ =\ 0}$
- Average Waiting Time, $\mathrm{wt}_{\mathrm{avg}\ =\ 0}$
- Total Visitor-System Time, $\mathrm{vst}_{\mathrm{total}\ =\ 0}$
- Average Visitor-System Time, $\mathrm{vst}_{\mathrm{avg}\ =\ 0}$

B.2 Food System

i) State Equations:

$$x(t^{+}) = \begin{cases} x(t) == 0 ?\, 1 : x(t), & arrival\ at\ time\ t \\ q(t) == 0 ?\, 0 : x(t), & arrival\ at\ time\ t \\ x(t), & otherwise \end{cases}$$

$$q(t^{+}) = \begin{cases} x(t) == 0 ?\, 0 : q(t) + 1, & arrival\ at\ time\ t \\ q(t) == 0 ?\, 0 : q(t) - 1, & arrival\ at\ time\ t \\ q(t), & otherwise \end{cases}$$

ii) State space:

X = {(0, 0), (1, 0)} ∪ ({1} × ℕ)
= {(0, 0),(1, 0),(1, 1),(1, 2),(1, 3), ...}

iii) Event set:

- E = {r, d}

iv) Output Equations:

- Total Waiting Time, $\mathrm{wt}_{\mathrm{total}} = \sum_{i=1}^{n} w_i$
- Average Waiting Time, $\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i$
- Total Service Time, $s = \sum_{i=1}^{n} s_i$
- Average Service Time, $\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$
- Total Visitor-System Time, $\mathrm{vst}_{\mathrm{total}} = \sum_{i=1}^{n} vs_i$
- Average Visitor-System Time,

$$\overline{vs} = \frac{1}{n} \sum_{i=1}^{n} vs_i$$
$$= \frac{1}{n} \sum_{i}^{n} (w_i + s_i)$$
$$= \frac{1}{n} \sum_{i=1}^{n} w_i + \frac{1}{n} \sum_{i=1}^{n} s_i$$
$$= \overline{w} + \bar{s}$$

C. Ride Systems:

C.1 Roller Coaster System, Carousel System

i) State Equations:

$$x(t^{+}) = \begin{cases} x(t) == 0 ?\, 1 : x(t), & arrival\ at\ time\ t \\ q(t) == 0 ?\, 0 : x(t), & arrival\ at\ time\ t \\ x(t), & otherwise \end{cases}$$

$$q(t^{+}) = \begin{cases} x(t) == 0 ?\, 0 : q(t) + 1, & arrival\ at\ time\ t \\ q(t) == 0 ?\, 0 : q(t) - 1, & arrival\ at\ time\ t \\ q(t), & otherwise \end{cases}$$

ii) State space:

X = {(0, 0), (1, 0)} ∪ ({1} × ℕ)
= {(0, 0),(1, 0),(1, 1),(1, 2),(1, 3), ...}

iii) Event set:

- E = {r, d}

iv) Output Equations:

- Total Waiting Time, $\mathrm{wt}_{\mathrm{total}} = \sum_{i=1}^{n} w_i$
- Average Waiting Time, $\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i$
- Total Service Time, $s = \sum_{i=1}^{n} s_i$
- Average Service Time, $\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$
- Total Visitor-System Time, $\mathrm{vst}_{\mathrm{total}} = \sum_{i=1}^{n} vs_i$
- Average Visitor-System Time,

$$\overline{vs} = \frac{1}{n} \sum_{i=1}^{n} vs_i$$
$$= \frac{1}{n} \sum_{i}^{n} (w_i + s_i)$$
$$= \frac{1}{n} \sum_{i=1}^{n} w_i + \frac{1}{n} \sum_{i=1}^{n} s_i$$
$$= \overline{w} + \bar{s}$$

C.2 Ferris Wheel System, Cable Car System

i) State Variables:

$$x_i\,(t^{+}) = \begin{cases} x_i\,(t) == 0 ?\, 1 : x_i\,(t), & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t) == 0 ?\, 0 : x_i\,(t), & arrival\ at\ time\ t, in\ ith\ server \\ x_i\,(t), & otherwise \end{cases}$$

$$q_i\,(t^{+}) \begin{cases} x_i\,(t) == 0 ?\, 0 : q_i\,(t) + 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t) == 0 ?\, 0 : q_i\,(t) - 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i\,(t), & otherwise \end{cases}$$

$$c_i\,(t^{+}) = \{c_i\,(t) < 4 ?\, c_i\,(t) + 1 : c_i\,, \qquad 4\ arrival\ at\ time\ t, in\ ith\ server$$

ii) State space:

$$S = \begin{Bmatrix} (0,0,0), (0,0,0), (0,0,0) \ldots\ldots\ldots\ldots.. \\ (1,0,0), (0,0,0), (0,0,0) \ldots\ldots\ldots\ldots\ldots \\ (1,0,0), (1,0,1), (0,0,0) \ldots\ldots\ldots\ldots\ldots \\ (1,0,0), (1,0,1), (1,0,2) \ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots.. \end{Bmatrix}$$

iii) Events:

- E = {r, l, u}

iv) Output Variables:

- Total Waiting Time, $\mathrm{wt}_{\mathrm{total}} = \sum_{i=1}^{n} w_i$
- Average Waiting Time, , $\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i$
- Total Visitor-Queue Time, $\mathrm{vqt}_{\mathrm{total}} = \sum_{i=1}^{n} inter_i$
- Average Visitor-Queue Time,
  $\mathrm{vqt}_{\mathrm{avg}} = \frac{1}{n} \sum_{i=1}^{n} inter_i$
- Total Visitor-System Time, , $\mathrm{vst}_{\mathrm{total}} = \sum_{i=1}^{n} vs_i$
- Average Visitor-System Time,

$$\overline{vs} = \frac{1}{n} \sum_{i=1}^{n} vs_i$$

$$= \frac{1}{n} \sum_{i}^{n} (w_i + s_i)$$

$$= \frac{1}{n} \sum_{i=1}^{n} w_i + \frac{1}{n} \sum_{i=1}^{n} s_i$$

$$= \overline{w} + \overline{s}$$

C.3 Water Slide System

i) State Variables:

$$x_i (t^+) = \begin{cases} x_i (t) == 0 ? 1 : x_i (t), & arrival\ at\ time\ t, in\ ith\ server \\ q_i (t) == 0 ? 0 : x_i (t), & arrival\ at\ time\ t, in\ ith\ server \\ x_i (t), & otherwise \end{cases}$$

$$q_i (t^+) = \begin{cases} x_i (t) == 0 ? 0 : q_i (t) + 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i (t) == 0 ? 0 : q_i (t) - 1, & arrival\ at\ time\ t, in\ ith\ server \\ q_i (t), & otherwise \end{cases}$$

where, i = 0, 1, 2, 3, 4 and N = 5

ii) State space:

$$x_i (t) = \{0, 1\} \qquad for\ i = 0, 1, 2, 3$$
$$q_i (t) = \{0, 1, 2 \ldots .\} \quad for\ i = 0, 1, 2, 3$$

$$S = \begin{Bmatrix} (0,0), (0,0), (0,0) \ldots \ldots \ldots \ldots . . \\ (1,0), (0,0), (0,0) \ldots \ldots \ldots \ldots \ldots \\ (1,0), (1,0), (0,0) \ldots \ldots \ldots \ldots \ldots \\ \ldots \ldots \ldots \ldots . . \end{Bmatrix}$$

iii) Events:

- E = {r, d}

iv) Output Variables:

- Total Waiting Time, wt$_{total}$ = $\sum_{i=1}^{n} w_i$
- Average Waiting Time, $\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i$
- Total Visitor-Queue Time, vqt$_{total}$ = $\sum_{i=1}^{n} inter_i$
- Average Visitor-Queue Time, vqt$_{avg}$ = $\frac{1}{n} \sum_{i=1}^{n} inter_i$
- Total Visitor-System Time, , vst$_{total}$ = $\sum_{i=1}^{n} vs_i$
- Average Visitor-System Time,

$$\overline{vs} = \frac{1}{n} \sum_{i=1}^{n} vs_i$$

$$= \frac{1}{n} \sum_{i}^{n} (w_i + s_i)$$

$$= \frac{1}{n} \sum_{i=1}^{n} w_i + \frac{1}{n} \sum_{i=1}^{n} s_i$$

$$= \overline{w} + \overline{s}$$

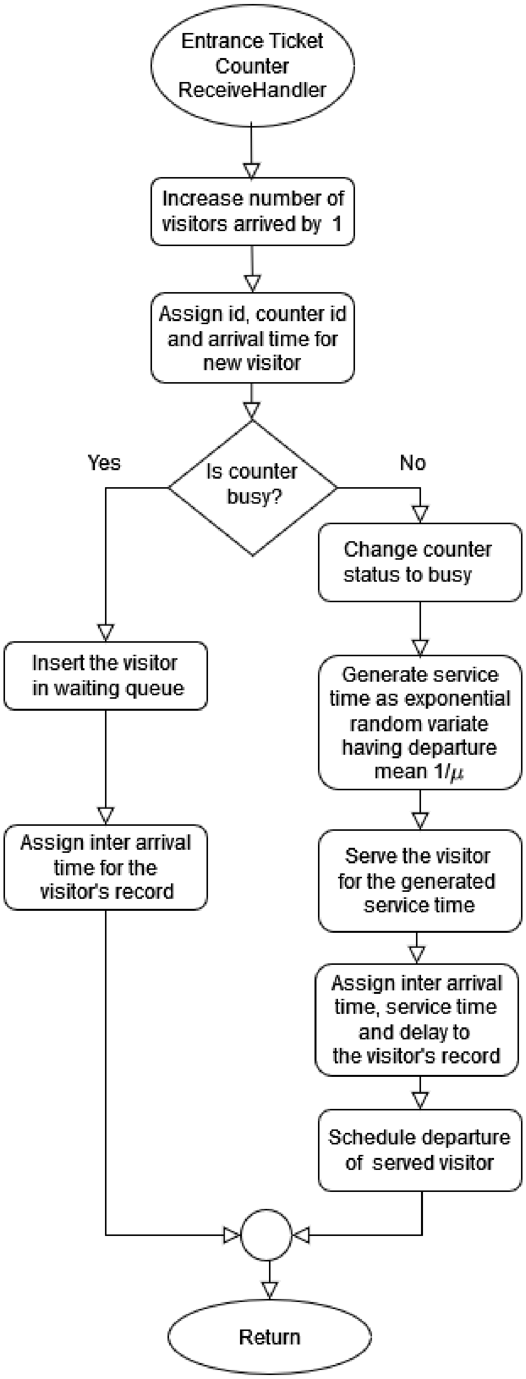
IV. Computational Model

A. Ticket Counter Systems:

A.1 Entrance Ticket Counter System

i) Receive Event Handler:

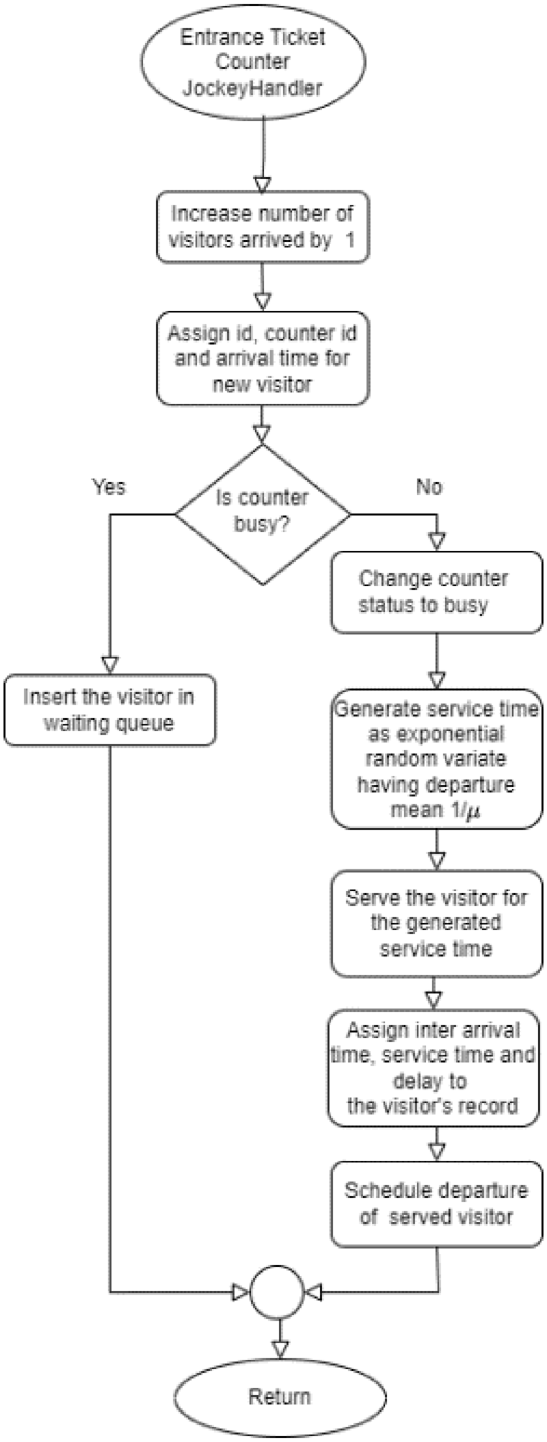
Visitors stream through the entrance, each assigned a unique ID, their arrival time recorded. Counters await, poised to serve. If available, a visitor joins a dedicated queue; if not, they patiently wait in the main line. Service times, tailored to each need, tick by as tickets are processed. Departures follow, orchestrated to create space for the next arrivals.

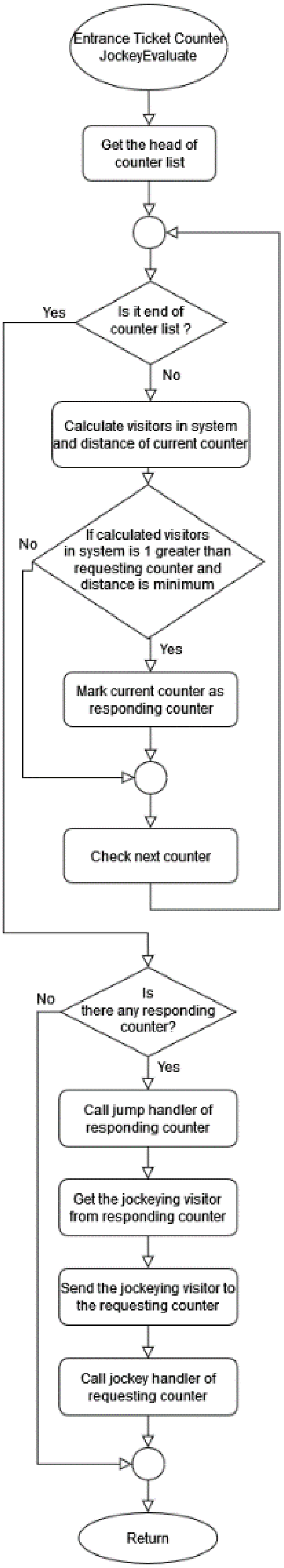


Entrance Ticket Counter ReceiveHandler
Increase number of visitors arrived by 1
Assign id, counter id and arrival time for new visitor
Is counter busy?
Yes
No
Change counter status to busy
Insert the visitor in waiting queue
Generate service time as exponential random variate having departure mean 1/μ
Assign inter arrival time for the visitor's record
Serve the visitor for the generated service time
Assign inter arrival time, service time and delay to the visitor's record
Schedule departure of served visitor
Return

ii) Jockey Handling Functions:

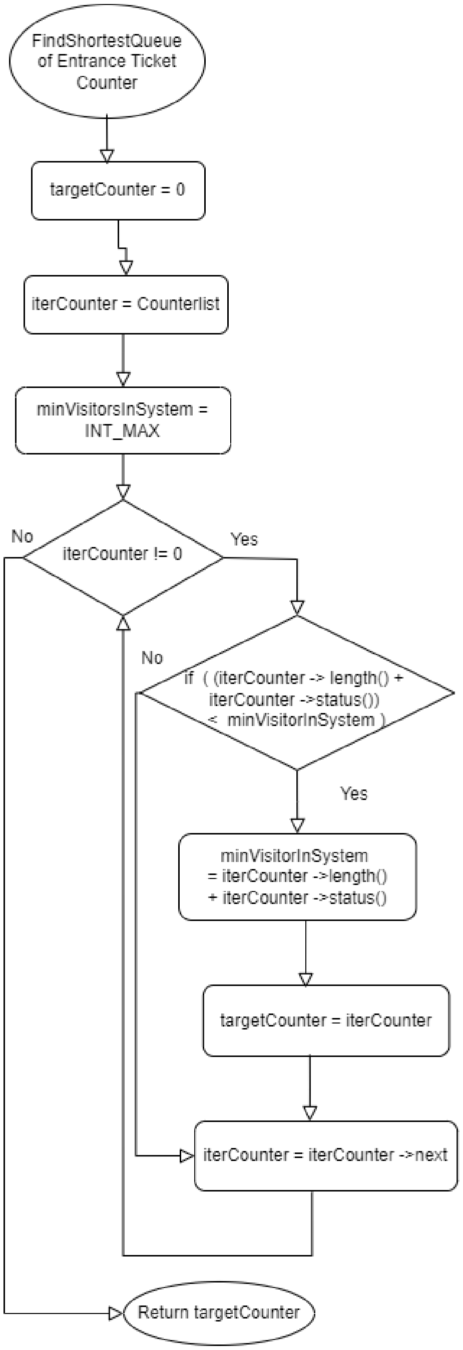
This process continues as new visitors arrive and are served. The flowchart also includes a "Jockeying Handler" which appears to handle situations where there may be more efficient ways to assign visitors to counters, but the details of this functionality are not shown in the image you provided.


Entrance Ticket Counter JockeyHandler
Increase number of visitors arrived by 1
Assign id, counter id and arrival time for new visitor
Is counter busy?
Yes
No
Change counter status to busy
Insert the visitor in waiting queue
Generate service time as exponential random variate having departure mean 1/μ
Serve the visitor for the generated service time
Assign inter arrival time, service time and delay to the visitor's record
Schedule departure of served visitor
Return


Entrance Ticket Counter JockeyEvaluate
Get the head of counter list
Is it end of counter list ?
Yes
No
Calculate visitors in system and distance of current counter
If calculated visitors in system is 1 greater than requesting counter and distance is minimum
No
Yes
Mark current counter as responding counter
Check next counter
Is there any responding counter?
No
Yes
Call jump handler of responding counter
Get the jockeying visitor from responding counter
Send the jockeying visitor to the requesting counter
Call jockey handler of requesting counter
Return

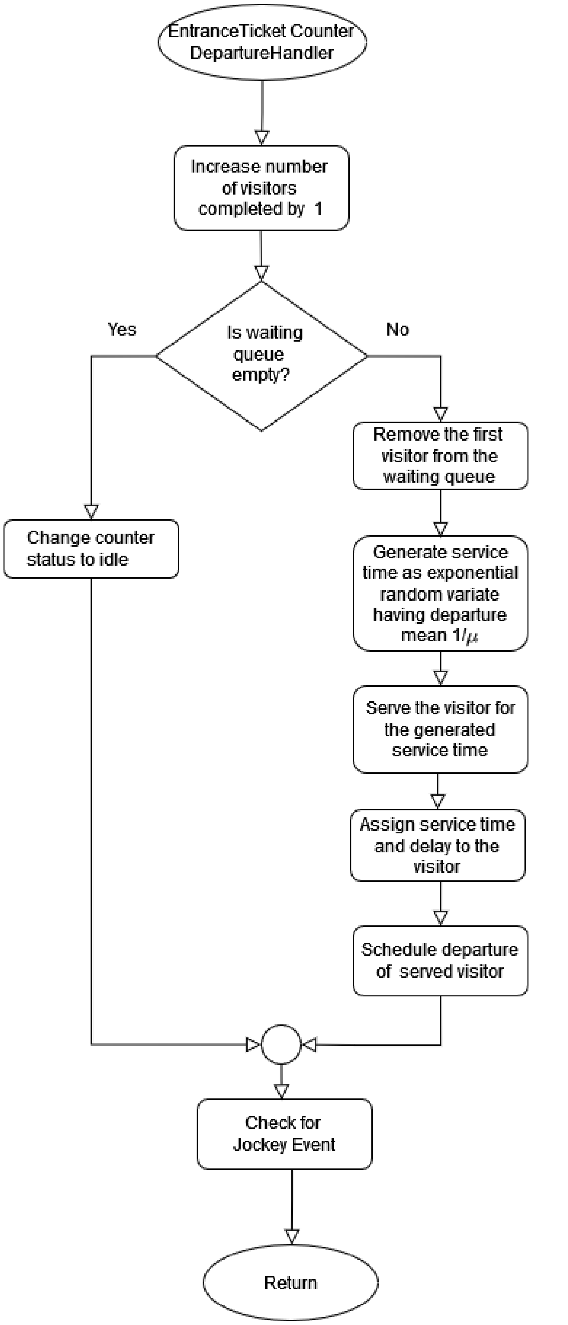
iii) Find Shortest Queue:

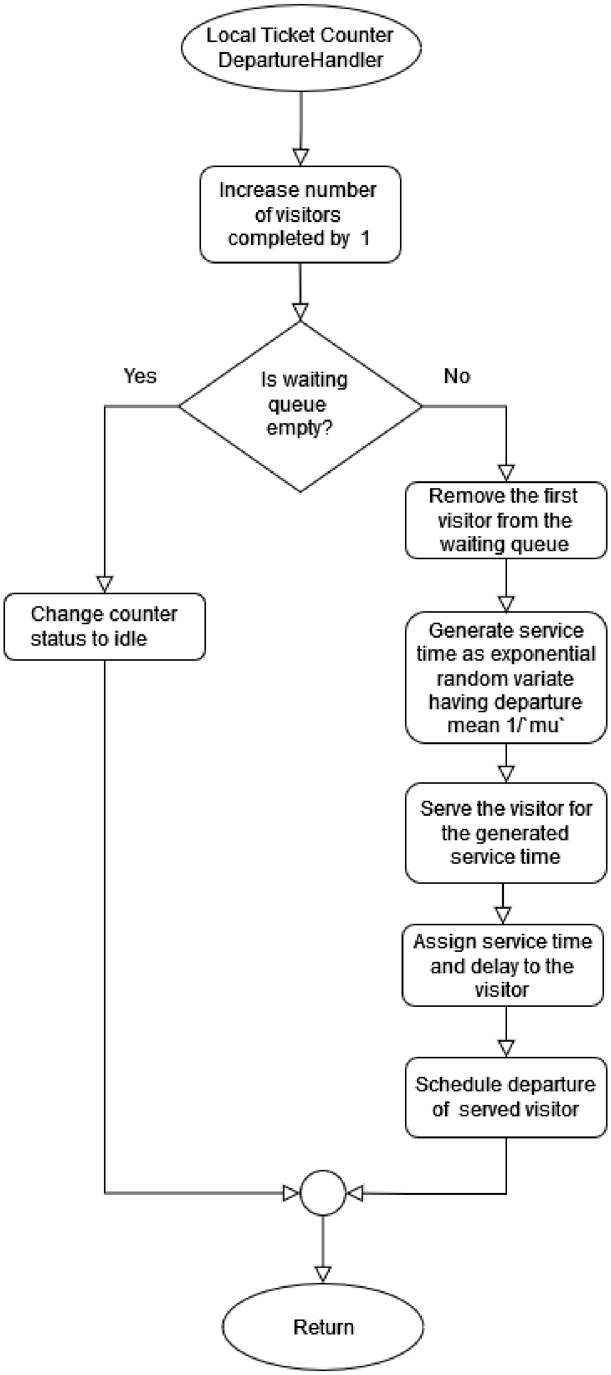
Arriving visitors are first counted. Then, for each ticket counter, the flowchart checks the number of people waiting in its queue and the status of the counter itself (busy or idle). If a counter is idle, it becomes the "target counter." If all counters are busy, the flowchart moves on to the next counter and repeats the check. Once the target counter is found, the visitor is routed to that queue. This process continues for each new visitor, ensuring they are directed to the shortest queue available.


FindShortestQueue of Entrance Ticket Counter
targetCounter = 0
iterCounter = Counterlist
minVisitorsInSystem = INT_MAX
iterCounter != 0
No
Yes
No
if ((iterCounter -> length() + iterCounter ->status()) < minVisitorInSystem)
Yes
minVisitorInSystem = iterCounter ->length() + iterCounter ->status()
targetCounter = iterCounter
iterCounter = iterCounter ->next
Return targetCounter

iv) Departure Event Handler:

Upon a visitor's departure, the system smoothly transitions to the next phase. It diligently marks the completion of their journey, then casts a watchful eye towards the waiting queue. If eager patrons remain, the first in line is swiftly ushered to the now-vacant counter. Service commences, tailored to their specific needs, with the duration determined by a carefully calculated formula. Once their transaction concludes, the system meticulously records their time spent, both in service and in patient waiting. With a final flourish, their departure is scheduled, and the counter, its task fulfilled, stands ready to welcome the next eager traveller.


EntranceTicket Counter DepartureHandler
Increase number of visitors completed by 1
Is waiting queue empty?
Yes
No
Remove the first visitor from the waiting queue
Change counter status to idle
Generate service time as exponential random variate having departure mean $1/\mu$
Serve the visitor for the generated service time
Assign service time and delay to the visitor
Schedule departure of served visitor
Check for Jockey Event
Return

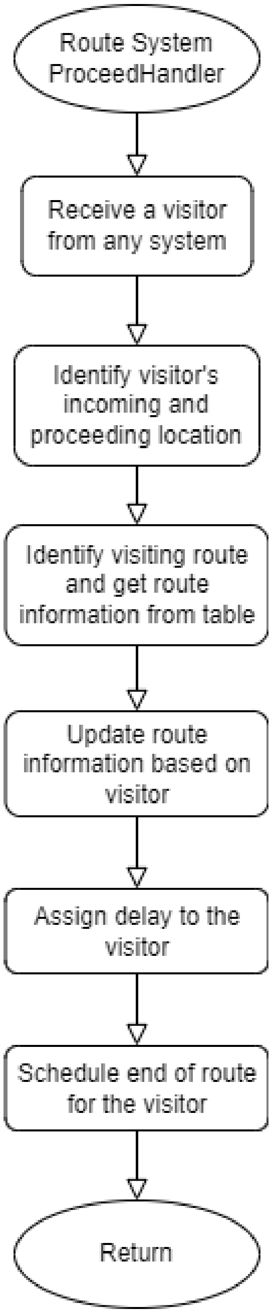

Local Ticket Counter DepartureHandler
Increase number of visitors completed by 1
Is waiting queue empty?
Yes
No
Change counter status to idle
Remove the first visitor from the waiting queue
Generate service time as exponential random variate having departure mean 1/`mu`
Serve the visitor for the generated service time
Assign service time and delay to the visitor
Schedule departure of served visitor
Return

B. General Systems:

B.1 Route System

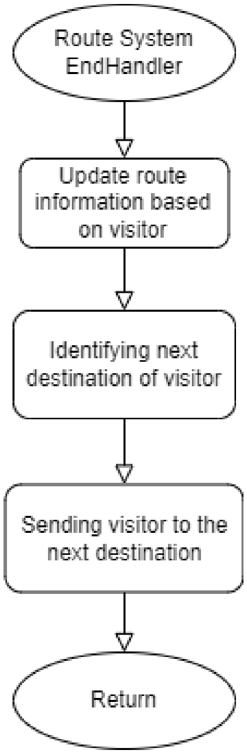
i) Proceed Event Handler:

It warmly welcomes each visitor, acknowledging their origins and intended destinations. Consulting a map of routes, it discerns the ideal path forward, weaving together a tapestry of destinations and delays. The visitor's journey is etched into the route's history, ensuring a seamless flow for those who follow. As their time within the system draws to a close, the handler orchestrates their departure, setting the stage for the next leg of their voyage.


Route System ProceedHandler
Receive a visitor from any system
Identify visitor's incoming and proceeding location
Identify visiting route and get route information from table
Update route information based on visitor
Assign delay to the visitor
Schedule end of route for the visitor
Return

ii) End Event Handler:

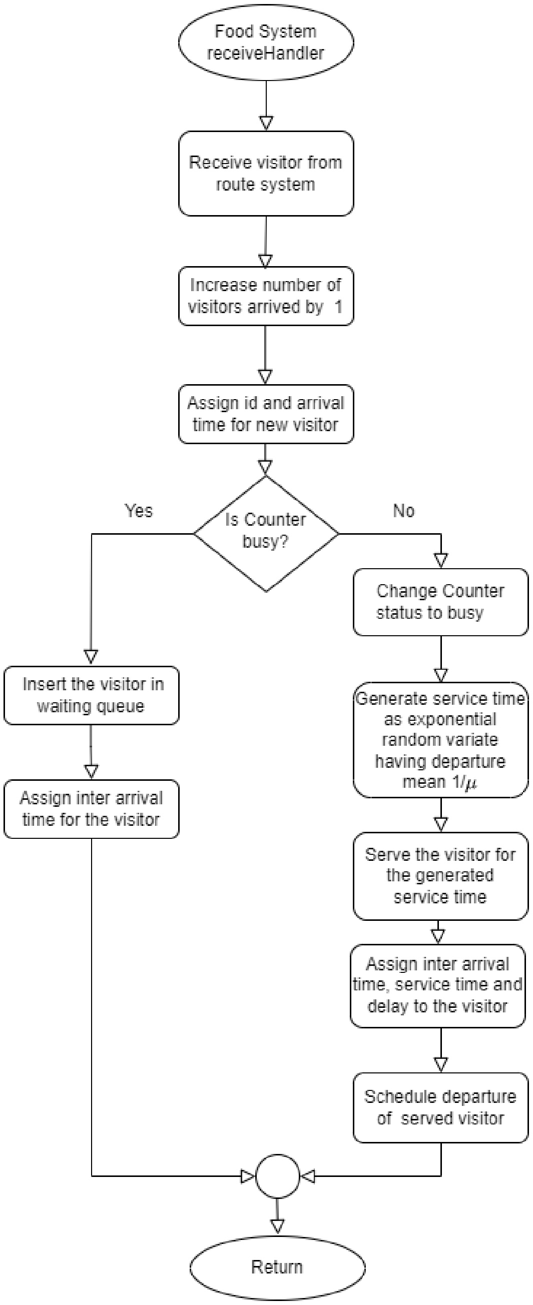
As a visitor's journey nears its end, the EndHandler steps in to ensure a smooth transition. It diligently refreshes the route information, ensuring future travelers benefit from the lessons learned. With a keen eye, it identifies the next destination, charting a course for continued exploration. The visitor is gently guided towards their next adventure, their progress carefully recorded to guide those who follow.


Route System EndHandler
Update route information based on visitor
Identifying next destination of visitor
Sending visitor to the next destination
Return

B.2 Food System

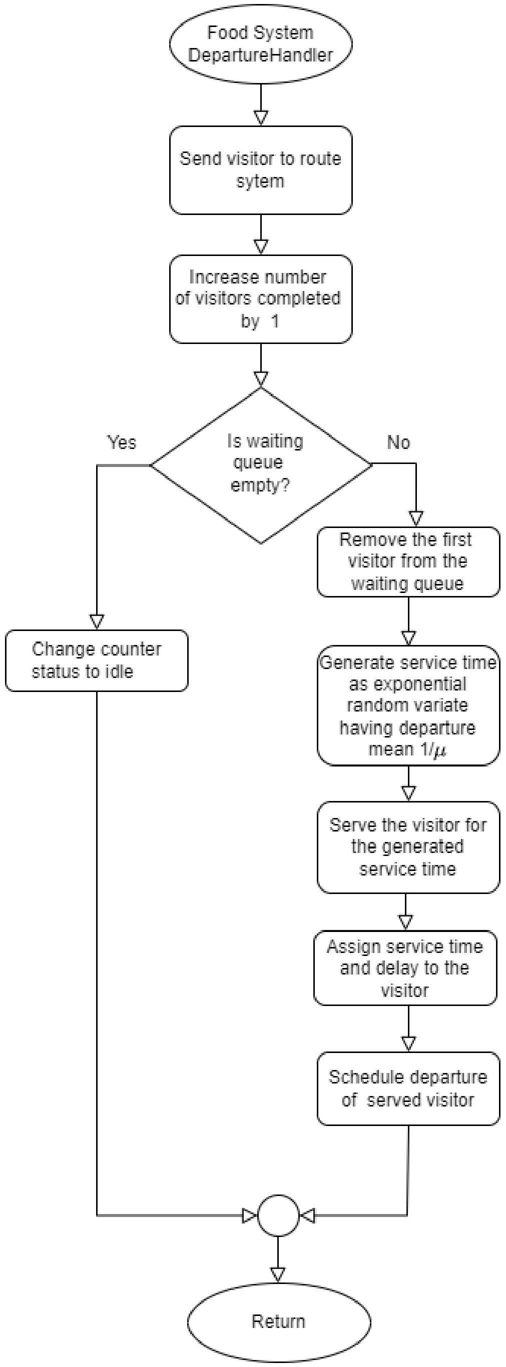
i) Receive Event Handler:

It assigns each a unique identity, marking their arrival within this nourishing realm. It expertly assesses the availability of counters, guiding those fortunate enough to find an open space directly to their feast. For those who must briefly linger, it offers solace in a patient queue, ensuring no soul goes unfed.


Food System receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign id and arrival time for new visitor
Is Counter busy?
Yes
No
Change Counter status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve the visitor for the generated service time
Assign inter arrival time, service time and delay to the visitor
Schedule departure of served visitor
Insert the visitor in waiting queue
Assign inter arrival time for the visitor
Return

ii) Departure Event Handler:

As a visitor departs, the Food System's DepartureHandler gracefully orchestrates their transition. It bids them a fond farewell, acknowledging their completion of a fulfilling journey. It diligently updates the system's records, tracking their departure and paving the way for new arrivals. With a keen eye, it scans the waiting queue, seeking those who eagerly anticipate their turn to savor the system's offerings. Should a hungry soul await, they are swiftly ushered forth, their service time determined by a delicate balance of anticipation and chance.

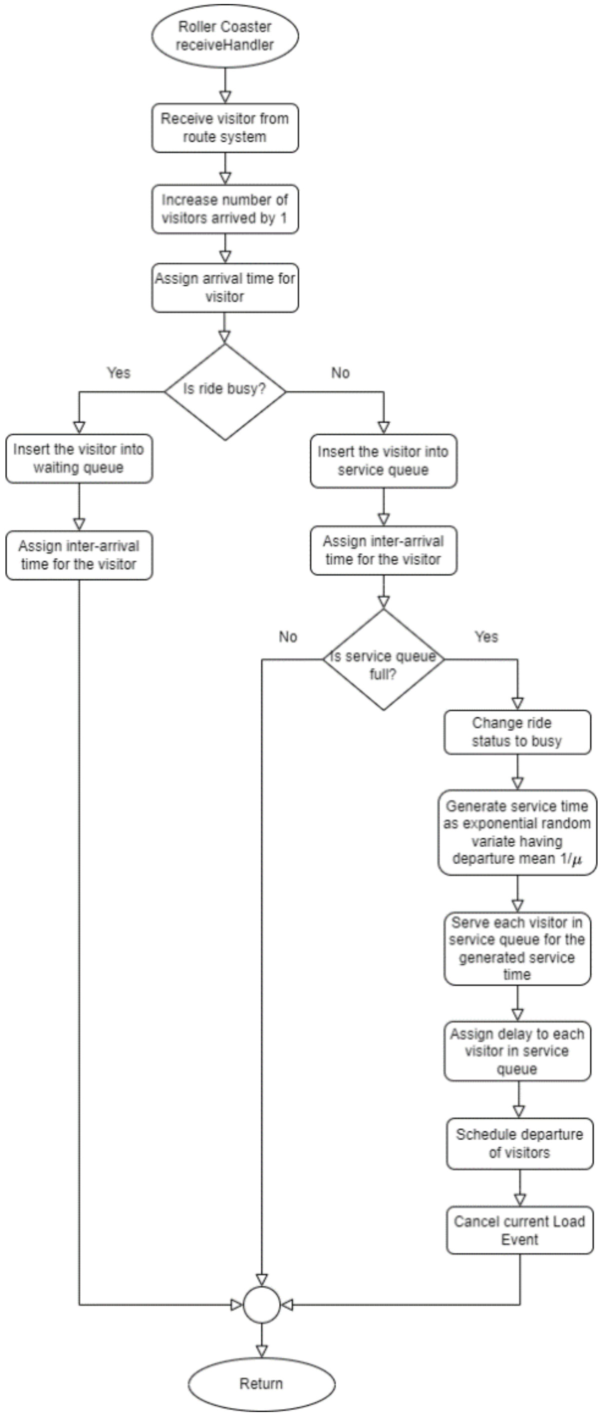

Food System DepartureHandler
Send visitor to route sytem
Increase number of visitors completed by 1
Is waiting queue empty?
Yes
No
Change counter status to idle
Remove the first visitor from the waiting queue
Generate service time as exponential random variate having departure mean 1/μ
Serve the visitor for the generated service time
Assign service time and delay to the visitor
Schedule departure of served visitor
Return

C. *Ride Systems:*

C.1 Roller Coaster System

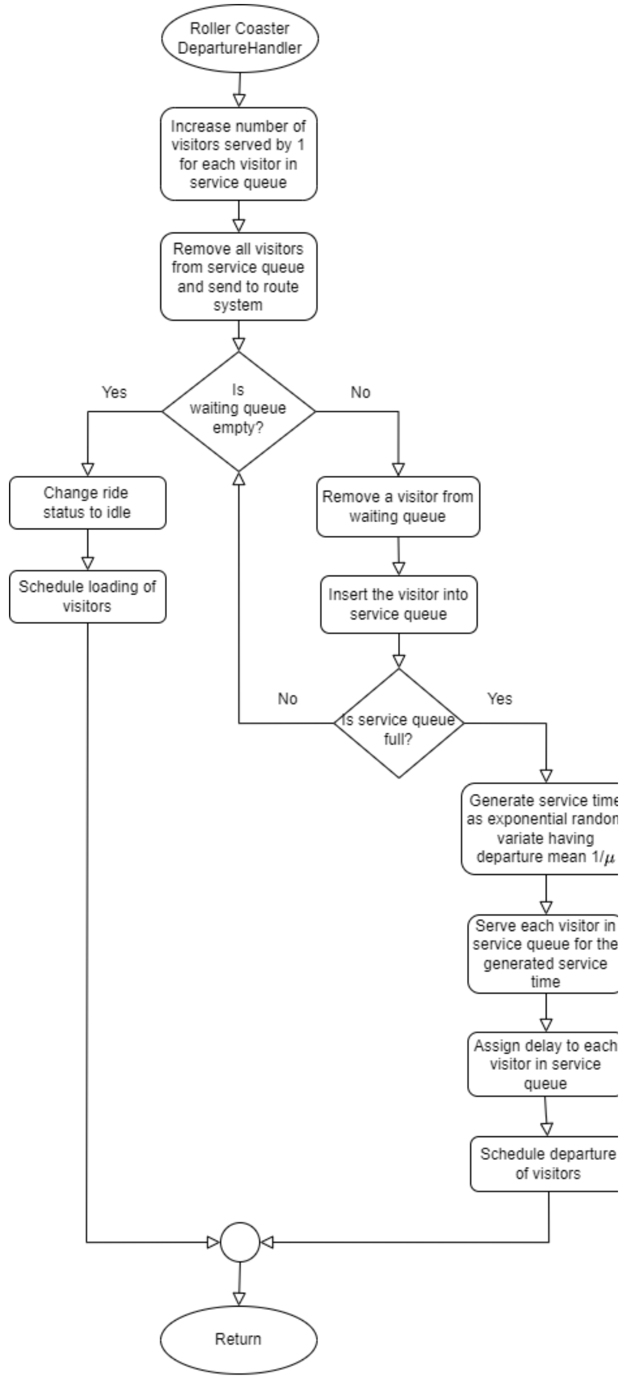
i) Receive Event Handler:

As riders disembark the rollercoaster, the receive handler welcomes them back, noting their arrival. Each rider receives a unique identifier for tracking. The system checks if any ride vehicles are available. If so, the rider is directed to the corresponding queue. If all vehicles are occupied, the rider waits in the main queue. The handler records the arrival time for each rider, capturing the system's flow.


Roller Coaster receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign arrival time for visitor
Is ride busy?
Yes
No
Insert the visitor into waiting queue
Insert the visitor into service queue
Assign inter-arrival time for the visitor
Assign inter-arrival time for the visitor
Is service queue full?
No
Yes
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Cancel current Load Event
Return

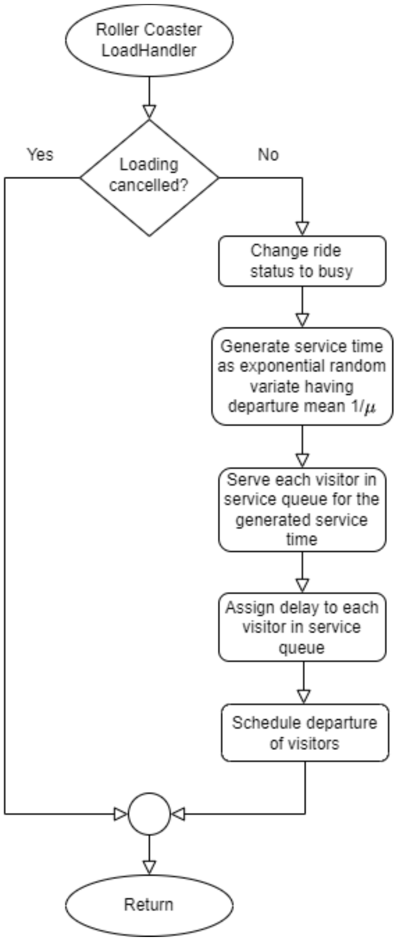
ii) Departure Event Handler:

As riders joyfully exit the rollercoaster, the departure handler gracefully ushers them on. It diligently marks their completion of the thrilling ride. Casting a watchful eye towards the waiting queue, it seeks the next eager souls to experience the excitement. If riders patiently await, the first in line is swiftly guided to the now-vacant vehicle. Their journey through the queue and service time are meticulously recorded. With a final flourish, their departure is scheduled, and the ride vehicle stands ready for the next wave of adventurers.


Roller Coaster DepartureHandler
Increase number of visitors served by 1 for each visitor in service queue
Remove all visitors from service queue and send to route system
Is waiting queue empty?
Yes
No
Change ride status to idle
Schedule loading of visitors
Remove a visitor from waiting queue
Insert the visitor into service queue
Is service queue full?
No
Yes
Generate service time as exponential random variate having departure mean $1/\mu$
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Return

iii) Load Event Handler:

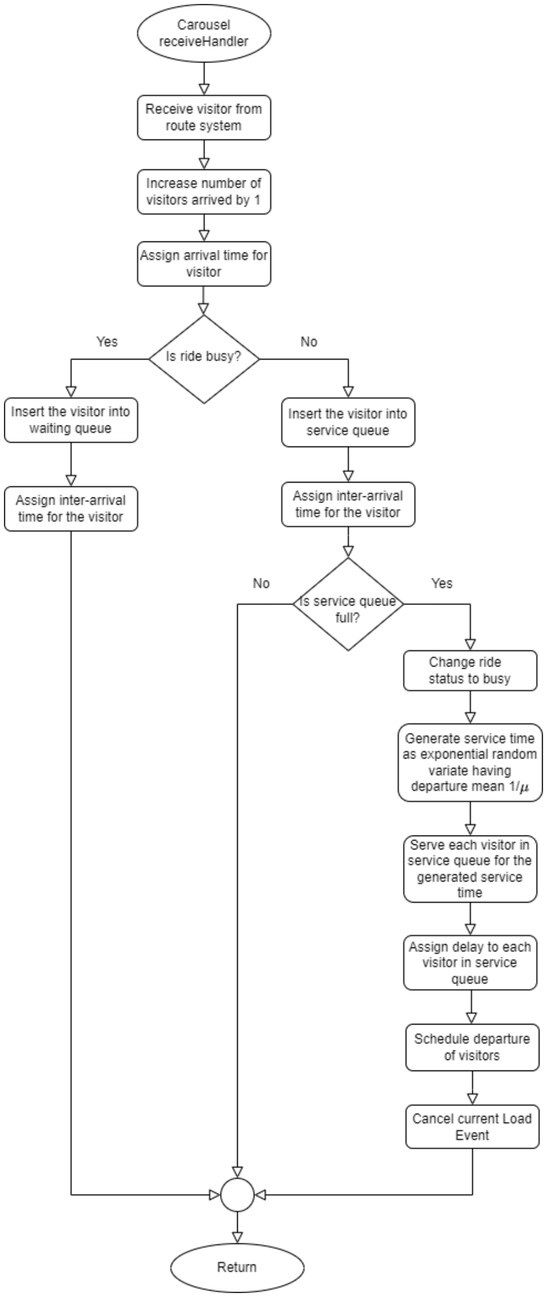
As riders prepare to embark on the roller coaster adventure, the load event handler steps in. It carefully checks if the ride is cancelled, ensuring everyone's safety. If the coast is clear, the handler assigns each rider a unique ID and tracks their arrival time. It then scans the available cars, directing riders to the shortest queue for a swift boarding experience. If all queues are full, riders join the main line and patiently wait their turn. This meticulous process ensures a smooth and efficient loading sequence for all eager adventurers.


Roller Coaster LoadHandler
Loading cancelled?
Yes
No
Change ride status to busy
Generate service time as exponential random variate having departure mean $1/\mu$
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Return

C.2 Carousel System

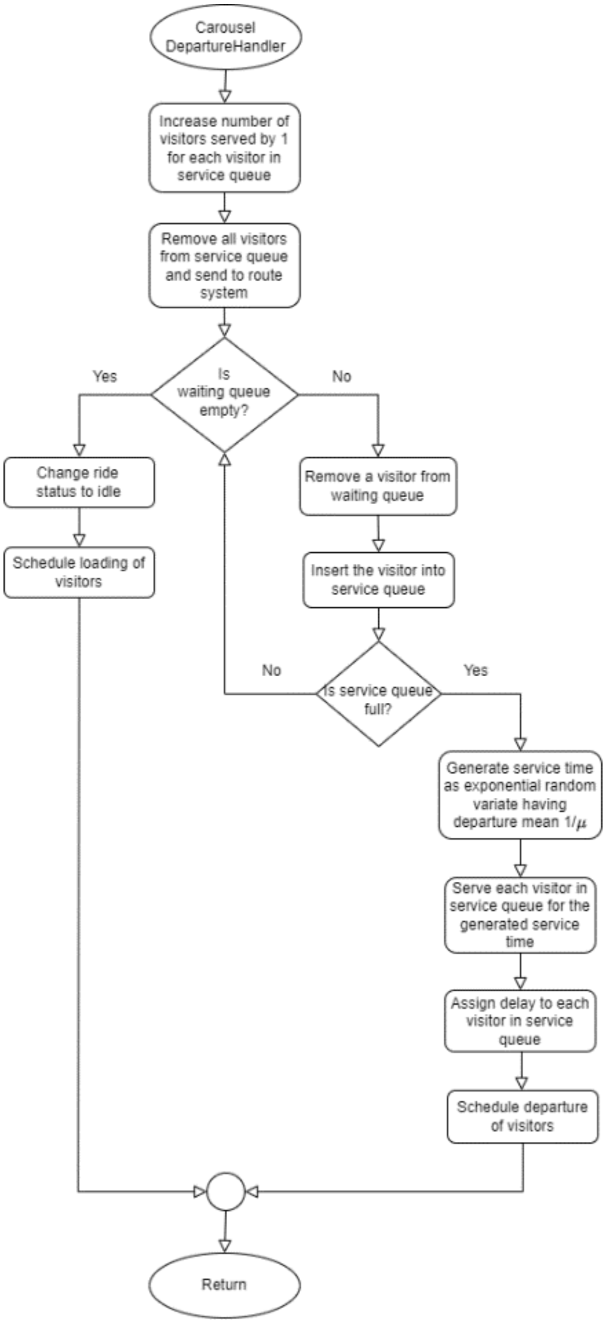
i) Receive Event Handler:

As riders disembark the Carousel, the receive handler welcomes them back, noting their arrival. Each rider receives a unique identifier for tracking. The system checks if any ride vehicles are available. If so, the rider is directed to the corresponding queue. If all spaces are occupied, the rider waits in the main queue. The handler records the arrival time for each rider, capturing the system's flow.


Carousel receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign arrival time for visitor
Is ride busy?
Yes
No
Insert the visitor into waiting queue
Assign inter-arrival time for the visitor
Insert the visitor into service queue
Assign inter-arrival time for the visitor
Is service queue full?
No
Yes
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Cancel current Load Event
Return

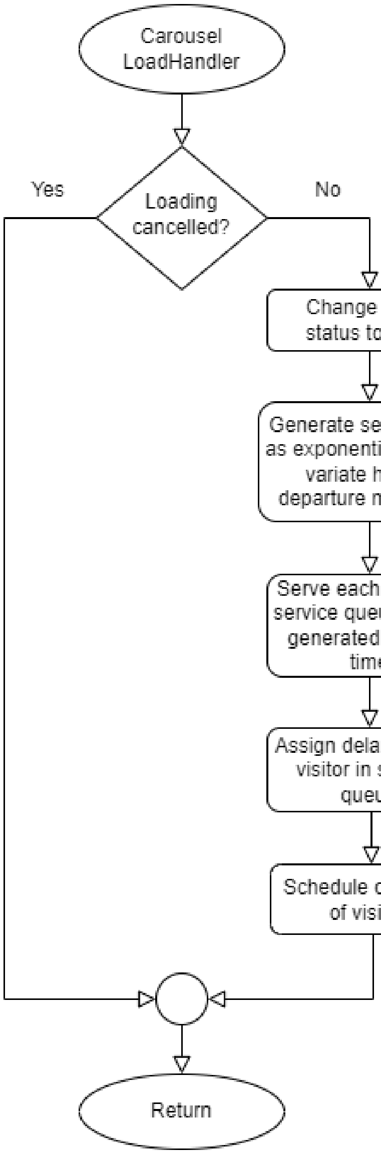
ii) Departure Event Handler:

As riders joyfully exit the ride, the departure handler gracefully ushers them on. It diligently marks their completion of the thrilling ride. Casting a watchful eye towards the waiting queue, it seeks the next eager souls to experience the excitement. If riders patiently await, the first in line is swiftly guided to the now-vacant vehicle. Their journey through the queue and service time are meticulously recorded. With a final flourish, their departure is scheduled, and the ride vehicle stands ready for the next wave of adventurers.


Carousel DepartureHandler
Increase number of visitors served by 1 for each visitor in service queue
Remove all visitors from service queue and send to route system
Is waiting queue empty?
Yes
No
Change ride status to idle
Schedule loading of visitors
Remove a visitor from waiting queue
Insert the visitor into service queue
Is service queue full?
No
Yes
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Return

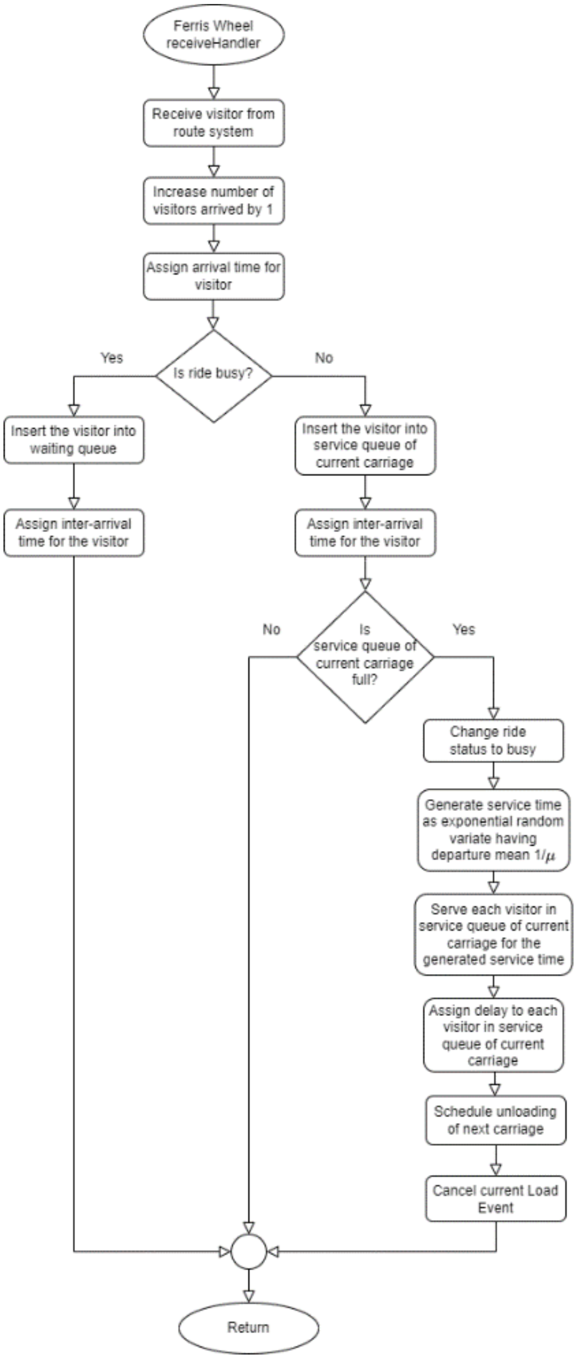
iii) Load Event Handler:

As riders prepare to embark on the roller coaster adventure, the load event handler steps in. It carefully checks if the ride is cancelled, ensuring everyone's safety. If the coast is clear, the handler assigns each rider a unique ID and tracks their arrival time. It then scans the available cars, directing riders to the shortest queue for a swift boarding experience. If all queues are full, riders join the main line and patiently wait their turn. This meticulous process ensures a smooth and efficient loading sequence for all eager adventurers.


Carousel LoadHandler
Loading cancelled?
Yes
No
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue for the generated service time
Assign delay to each visitor in service queue
Schedule departure of visitors
Return

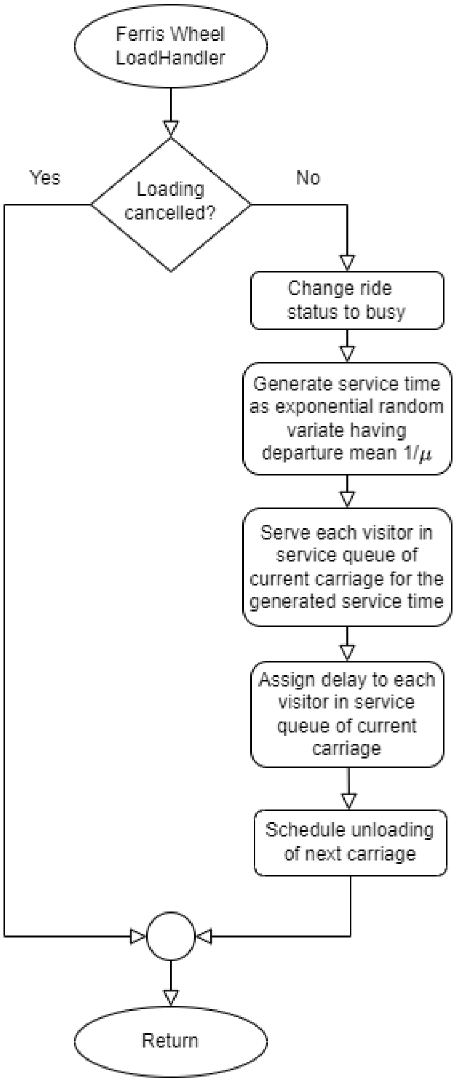
C.3 Ferris Wheel System

i) Receive Event Handler:
It diligently counts each new arrival, ensuring no one is overlooked. It then assigns a unique identifier to each visitor, tracking their journey through the system. The handler gracefully checks the ride's status, determining if it's ready to receive guests. If the ride is busy, visitors are gently guided to a waiting queue, where they can anticipate their moment of flight. If a carriage stands empty, the handler swiftly ushers visitors aboard, setting the stage for their ascent.


Ferris Wheel receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign arrival time for visitor
Is ride busy?
Yes
No
Insert the visitor into waiting queue
Assign inter-arrival time for the visitor
Insert the visitor into service queue of current carriage
Assign inter-arrival time for the visitor
Is service queue of current carriage full?
No
Yes
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Cancel current Load Event
Return

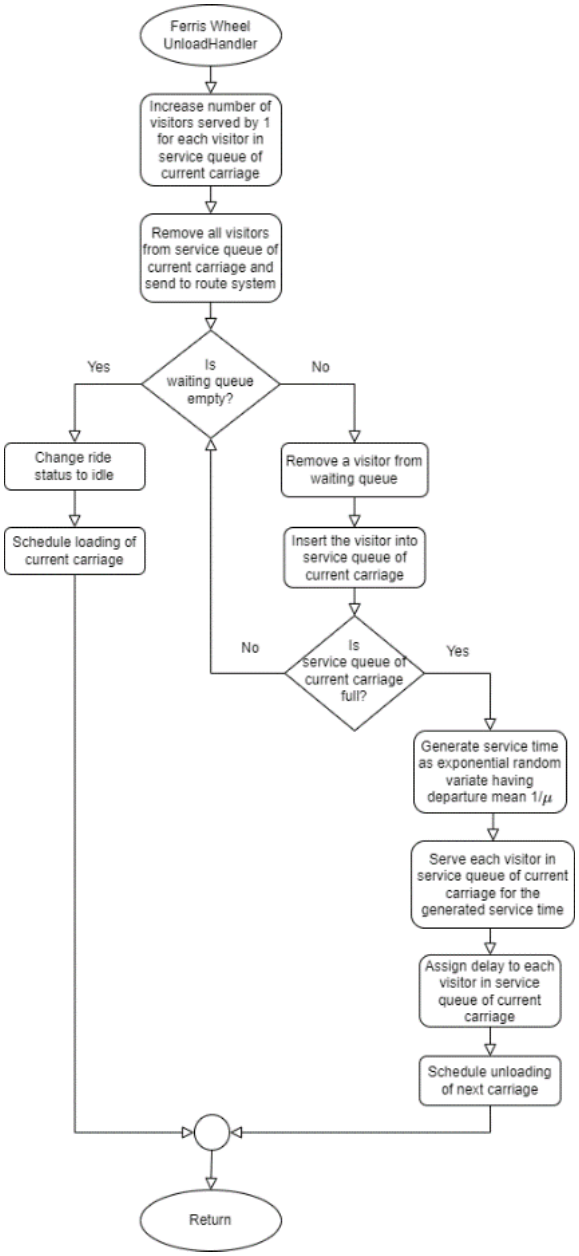
ii) Load Event Handler:
Ferris Wheel's load event handler initiates its task. It first checks for any cancellations, ensuring the ride's smooth operation. If the journey is ready to commence, the handler diligently changes the ride status to "busy," signalling its engagement. It then expertly generates a unique service time for each carriage, ensuring a balanced experience for all passengers. With precision, it guides those in the waiting queue to their designated carriages, where they'll embark on a captivating ascent. The handler meticulously assigns a delay value to each visitor, capturing their unique experience within the

system. Finally, with a soft whisper, it schedules the unloading of the next carriage


Ferris Wheel LoadHandler
Loading cancelled?
Yes
No
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Return

iii) Unload Event Handler:

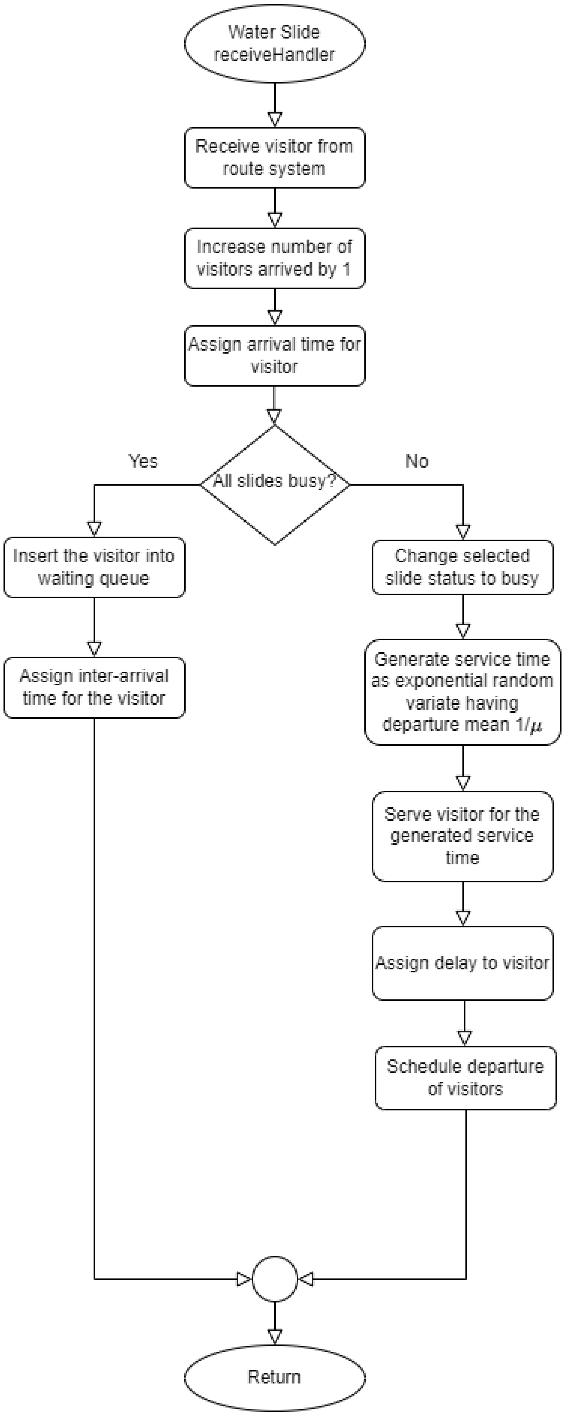
Ferris Wheel gracefully completes its rotation, the unload event handler steps in to orchestrate a seamless transition. It diligently records the number of visitors served, keeping track of the ride's impact. With a gentle hand, it guides visitors from the carriages, ensuring their safe return to solid ground. The handler then efficiently directs them towards the route system, where they can continue their journey through the park's delights. It meticulously checks for those who await their turn, ready to ascend to new heights. If eager riders remain, the handler swiftly assigns them to the now-empty carriage, ensuring the Ferris Wheel's enchanting dance continues without pause.


Ferris Wheel UnloadHandler
Increase number of visitors served by 1 for each visitor in service queue of current carriage
Remove all visitors from service queue of current carriage and send to route system
Is waiting queue empty?
Yes
No
Change ride status to idle
Schedule loading of current carriage
Remove a visitor from waiting queue
Insert the visitor into service queue of current carriage
Is service queue of current carriage full?
No
Yes
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Return

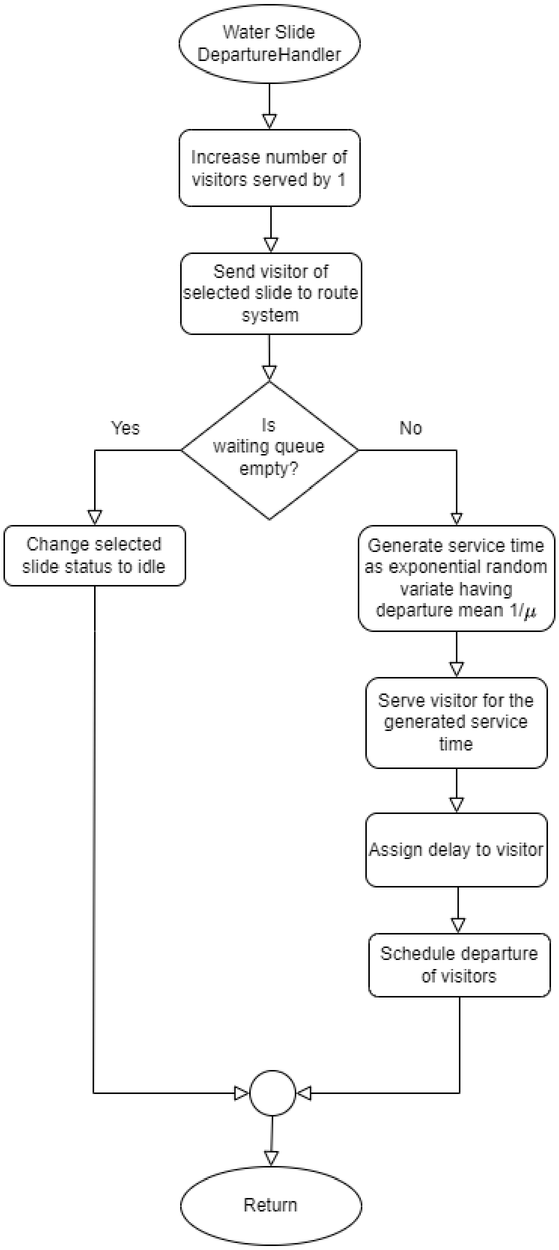
C.4 Water Slide System

i) Receive Handler:

As eager riders approach the watery plunge, the receive handler welcomes them, counting each arrival and assigning unique IDs. It scans the available slides, seeking the shortest queue for a swift descent. If a slide stands empty, the rider glides gracefully towards its dedicated line, wait time calculated for a refreshing dip. If all slides are occupied, the rider patiently joins the main queue, anticipation building with each new arrival. The handler diligently records each arrival time, capturing the flow of thrill-seekers eager to embrace the watery rush.

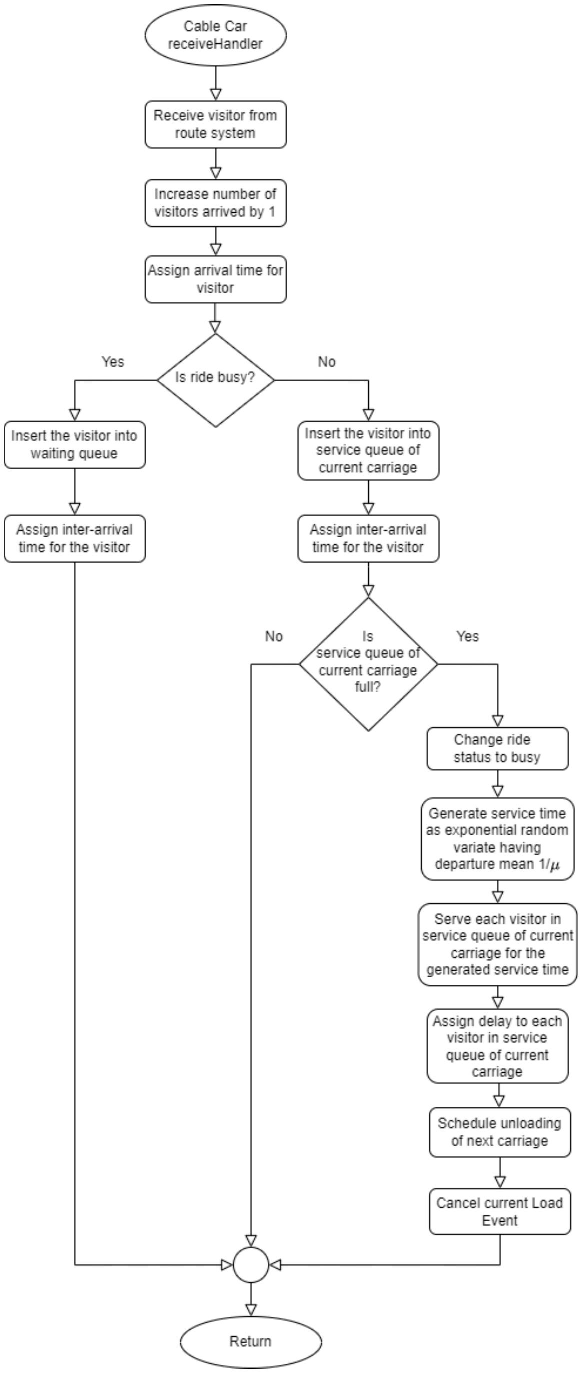

Water Slide receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign arrival time for visitor
All slides busy?
Yes
No
Insert the visitor into waiting queue
Assign inter-arrival time for the visitor
Change selected slide status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve visitor for the generated service time
Assign delay to visitor
Schedule departure of visitors
Return

ii) Departure Handler:
The departure handler, ever vigilant, marks their journey complete. Scanning the waiting queue, it seeks the next souls eager for a thrilling plunge. If impatient adventurers await, the first in line is swiftly ushered to the now-vacant slide. Their service time, a delicate blend of anticipation and chance, is meticulously recorded. As their watery adventure concludes, the handler captures their journey, from splashdown to departure. With a final flourish, their exit is orchestrated, and the slide, refreshed and eager, stands ready to welcome the next wave of thrill-seekers.


Water Slide DepartureHandler
Increase number of visitors served by 1
Send visitor of selected slide to route system
Is waiting queue empty?
Yes
No
Change selected slide status to idle
Generate service time as exponential random variate having departure mean 1/μ
Serve visitor for the generated service time
Assign delay to visitor
Schedule departure of visitors
Return

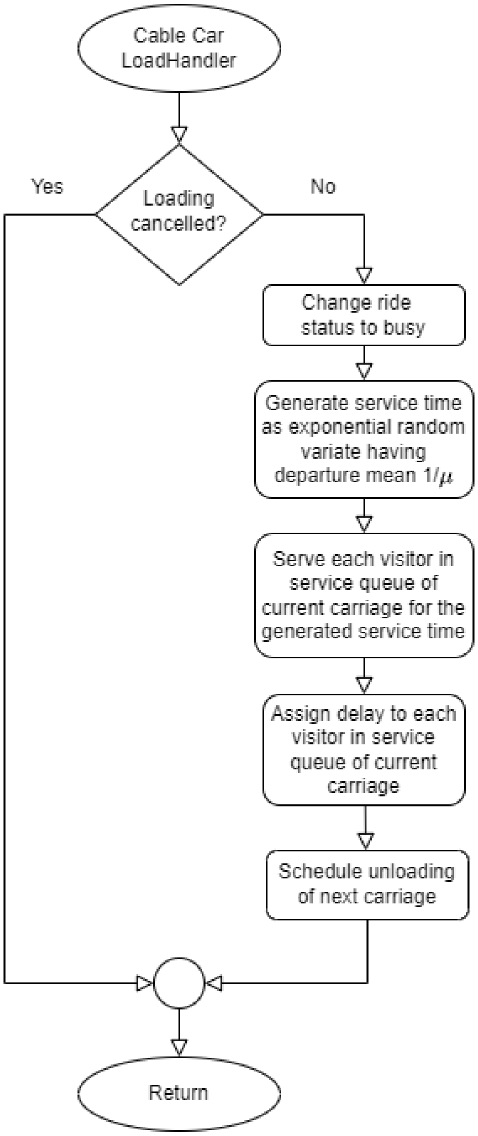
C.5 Cable Car System

i) Receive Handler:
It diligently counts each new arrival, ensuring no one is overlooked. It then assigns a unique identifier to each visitor, tracking their journey through the system. The handler gracefully checks the ride's status, determining if it's ready to receive guests. If the ride is busy, visitors are gently guided to a waiting queue, where they can anticipate their moment of flight. If a carriage stands empty, the handler swiftly pushes visitors aboard, setting the stage for their ascent.


Cable Car receiveHandler
Receive visitor from route system
Increase number of visitors arrived by 1
Assign arrival time for visitor
Is ride busy?
Yes
No
Insert the visitor into waiting queue
Assign inter-arrival time for the visitor
Insert the visitor into service queue of current carriage
Assign inter-arrival time for the visitor
Is service queue of current carriage full?
No
Yes
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Cancel current Load Event
Return

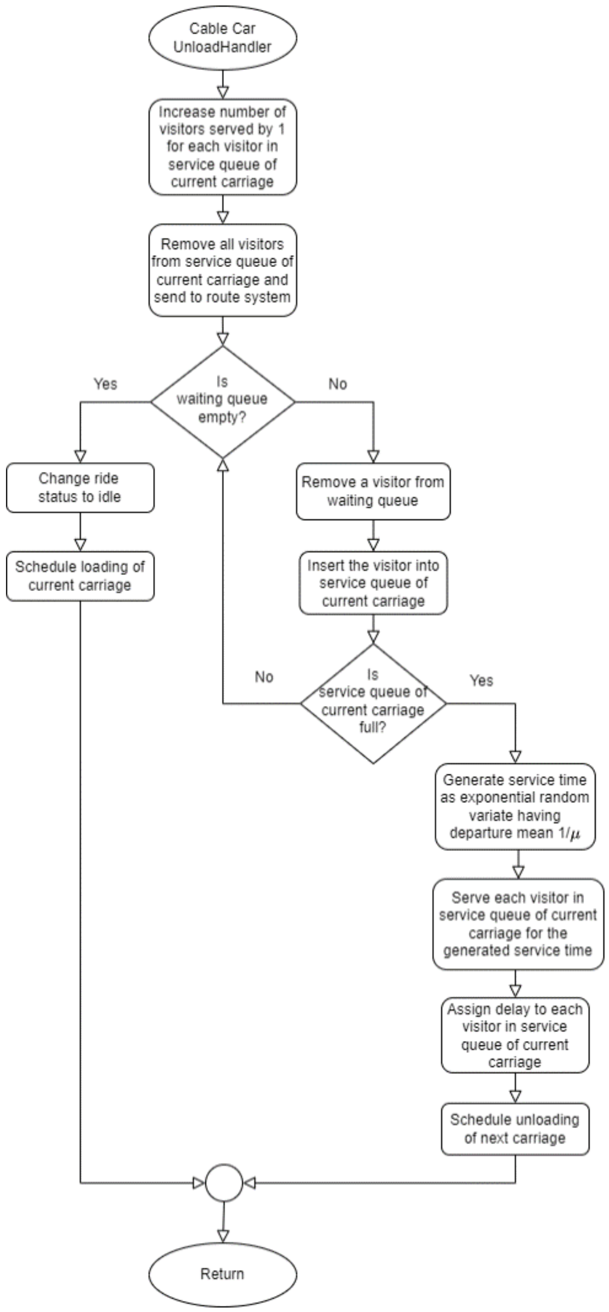
ii) Load Handler:

Cable car’s load event handler initiates its task. It first checks for any cancellations, ensuring the ride's smooth operation. If the journey is ready to commence, the handler diligently changes the ride status to "busy," signalling its engagement. It then expertly generates a unique service time for each carriage, ensuring a balanced experience for all passengers. With precision, it guides those in the waiting queue to their designated carriages, where they'll embark on a captivating ascent. The handler meticulously assigns a delay value to each visitor, capturing their unique experience within the system. Finally, with a soft whisper, it schedules the unloading of the next carriage


Cable Car LoadHandler
Loading cancelled?
Yes
No
Change ride status to busy
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Return

iii) Unload Handler:

Cable Car gracefully completes its rotation, the unload event handler steps in to orchestrate a seamless transition. It diligently records the number of visitors served, keeping track of the ride's impact. With a gentle hand, it guides visitors from the carriages, ensuring their safe return to solid ground. The handler then efficiently directs them towards the route system, where they can continue their journey through the park's delights. It meticulously checks for those who await their turn, ready to ascend to new heights. If eager riders remain, the handler swiftly assigns them to the now-empty carriage.


Cable Car UnloadHandler
Increase number of visitors served by 1 for each visitor in service queue of current carriage
Remove all visitors from service queue of current carriage and send to route system
Is waiting queue empty?
Yes
No
Change ride status to idle
Schedule loading of current carriage
Remove a visitor from waiting queue
Insert the visitor into service queue of current carriage
Is service queue of current carriage full?
No
Yes
Generate service time as exponential random variate having departure mean 1/μ
Serve each visitor in service queue of current carriage for the generated service time
Assign delay to each visitor in service queue of current carriage
Schedule unloading of next carriage
Return

V. Simulation Program Description

The simulation program consists of three types of classes:

- System Component Classes: These are the classes that work as main components of the system.
  - Visitor Class: This is structure of visitor of the amusement park. Each visitor has an id to uniquely identify each visitor, a path of destinations that the visitor will visit during the simulation and other statistical variables to keep track of the waiting time at different destinations.
  - Server Class: This works as serving component at each destination and it serves the visitors at that destination. It contains a service queue to serve a particular amount of visitor specified at each destination setup.
  - Terminal Class: This works as a intermediate place for visitor before entering server. It maintains the queue(s) for the server(s) at that destination.
- System Utility Classes: These are utility classes that assist the system to run.
  - Sim Object Class: This is the parent class for most of the component and structure classes to provide communication between classes.
  - Event Class: This resembles the events the happening in the system and handled by the system management classes.
  - Handler Class: This is used to handle the events that are triggered by system management classes.
  - Queue Class: This is used to maintain queue at different destinations.
  - Policy Class: This is used to define and apply policy for each destination in the system.
- System Structure Classes: These are the classes that forms the structure of the system.
  - Generator Class: This generates the visitor and sends to the system.
  - Destination Class: This is used to struct different destination for the visitor to visit using component classes.
  - Terminator Class: This terminates the leaving visitors and produces result for the simulation.
  - Router Class: This routes visitor from one destination to another destination.
- System Management Classes: These are the classes that controls operation among different classes and runs the system according to the design.
  - Scheduler Class: This class schedules events that happen in the system and executes them.
  - Park System Class: This class is where the formation of the system is built and provided a method to run the system.

The full source code of the simulation program along with the trace files can be found here: https://github.com/A6du11ah/Amusement-Park-Simulation/tree/main/code

VI. Result

TABLE 01

AMUSEMENT PARK SIMULATION RESULT

[GENERAL POLICY]

Destination	Average Waiting Time (hours)
Ticket Counter	0.013138932
Roller Coaster	0.875802809
Carousel	0.592264924
Ferris Wheel	0.427905346
Cable Car	0.268597777
Water Slide	0.001124347
Food Counter	0.020990691

TABLE 02

AMUSEMENT PARK SIMULATION RESULT

[BUSY DAY POLICY]

Destination	Average Waiting Time (hours)
Ticket Counter	0.060892453
Roller Coaster	0.788745058
Carousel	0.608817355
Ferris Wheel	33.64636638
Cable Car	0.25677968
Water Slide	0.002660071
Food Counter	0.133374492

TABLE 01

AMUSEMENT PARK SIMULATION RESULT

[ROUTE VAN POLICY]

Destination	Average Waiting Time (hours)
Ticket Counter	0.013139054
Roller Coaster	0.846196296
Carousel	0.543505538
Ferris Wheel	0.400271095
Cable Car	0.221180914
Water Slide	0.001094749
Food Counter	0.023567356

TABLE 01

AMUSEMENT PARK SIMULATION RESULT

[CAPACITY INCREASE POLICY]

Destination	Average Waiting Time (hours)
Ticket Counter	0.013138256
Roller Coaster	1.156857769
Carousel	0.842059835
Ferris Wheel	0.430947127
Cable Car	0.231241023
Water Slide	0.002309534
Food Counter	0.031357978

VII. Analysis And Comparison Between Policies

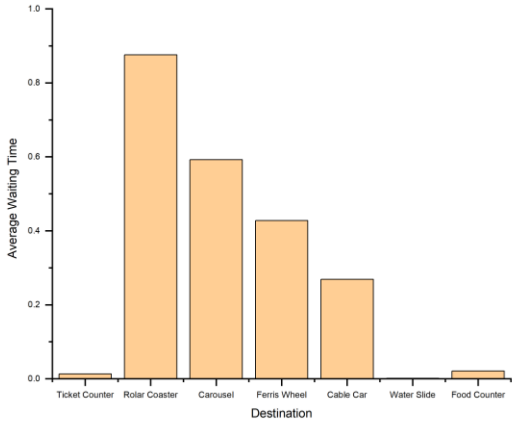


Fig-1A: Graph of average waiting time at each destination in general

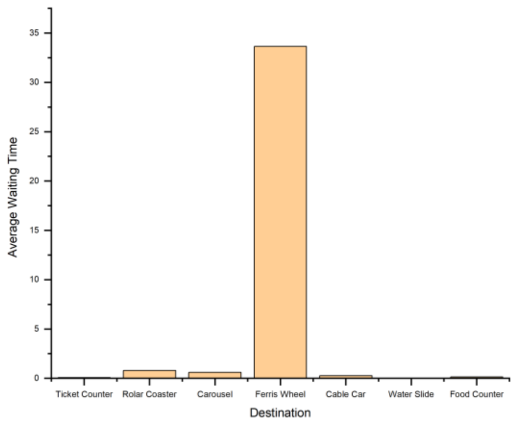


Fig-2A: Graph of average waiting time at each destination on a busy day

A. *Comparison of General Policy and Busy Day Policy:*

Busy Day Policy differs from General Policy on the context of arrival rate of visitors at the park. Busy Day Policy has higher arrival rate than General Policy. The Fig-1A and Fig-2A shows the statistics of waiting time at each destination in the Amusement Park System for both policies. Following the Busy Day Policy, the waiting time at each destination is increased vastly from the General Policy as the number of visitors increases in the system.

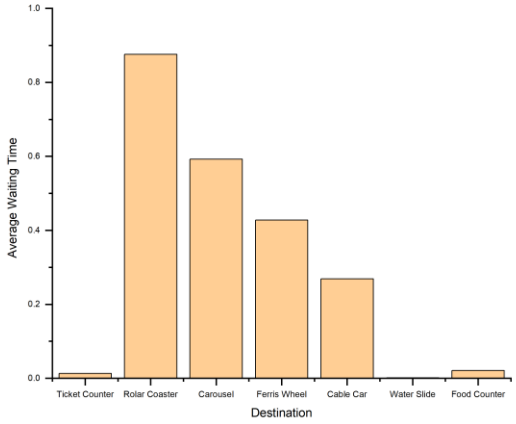


Fig-1B: Graph of average waiting time at each destination in general

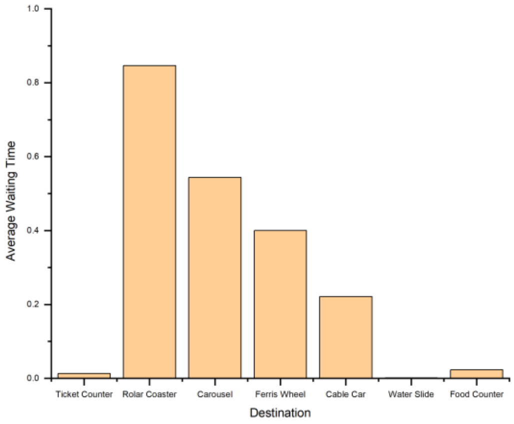
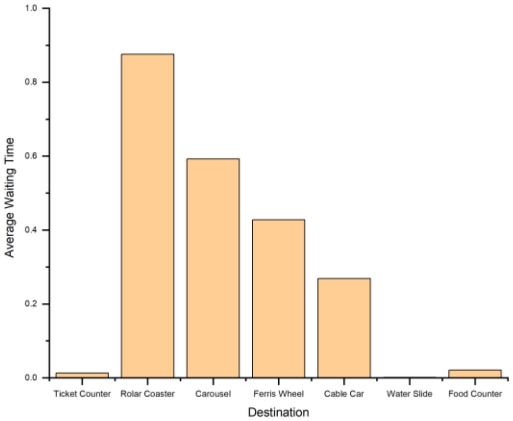


Fig-2B: Graph of average waiting time at each destination deploying route vans to travel from one destination to another

B. *Comparison of General Policy and Route Van Policy:*

Route Van Policy differs from General Policy on the context of routing time from one place to another. Route Van Policy has lower arrival rate than General Policy provided that route vans are deployed to make the routing comfortable as well as faster for visitors. The Fig-1B and Fig-2B shows the statistics of waiting time at each destination in the Amusement Park System for both policies. Following the Route Van Policy, the waiting time at each destination has decreased from the General Policy as it takes less time to visit from one place to another.



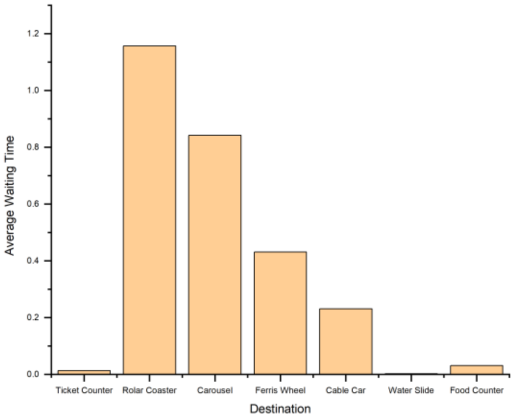


Fig-1C: Graph of average waiting time at each destination on a normal day

Fig-2C: Graph of average waiting time at each destination increasing serving capacity of each ride at different destination

C. Comparison of General Policy and Capacity Increase Policy:

Capacity Increase Policy differs from General Policy on the context of capacity of each ride at different destination. Capacity Increase Policy has higher capacity for each ride than General Policy. The Fig-1C and Fig-2C shows the statistics of waiting time at each destination in the Amusement Park System for both policies. Following the Capacity Increase Policy, the waiting time at each destination is increased slightly from the General Policy but serves more visitors than before.