Optimizing Resource Utilization and Time Efficiency in Fast-Food Restaurant Operations Using Discrete Petri-Net Models

Project Report

In the DAT530 Course of the master program Computer Science at

Universitetet i Stavanger

Written by

B M Nafis Fuad

Md Safin Sarker

Course Supervisor: Prof. Reggie Davidrajuh

Submission Date: **10th November,2023**

Abstract

This report presents the utilization of GPenSIM for conducting discrete simulations of a fast-food restaurant, aimed at gaining valuable insights into the key determinants of restaurant optimization. The model of the fast-food restaurant is constructed through the implementation of a flexible, modular, timed, and colored PetriNet, with the staffs represented as resources. To enable and disable specific functionalities and scenarios, capacity constraints are employed as firing conditions across various PRE files. The study covers the calculation of total costs, income, and overall profit under diverse scenarios, providing a comprehensive understanding of the fast-food workflow and the profitability associated with each scenario. Additionally, the report offers visualizations that depict the usage of different modules at various time intervals. By manipulating different variables to create distinct scenarios, this model offers a comprehensive overview of the fast-food restaurant's operational workflow and the profitability associated with each specific scenario.

List of Figures

List of Tables

Table of Contents

1. Introduction

This project is a component of the course DAT530: Discrete Simulation and Performance Analysis at Universitetet i Stavanger. The GPenSIM MATLAB toolbox is employed for the purpose of optimizing resource utilization and time management within the fast-food restaurant industry. This optimization is achieved through the utilization of a PetriNet-based modeling approach

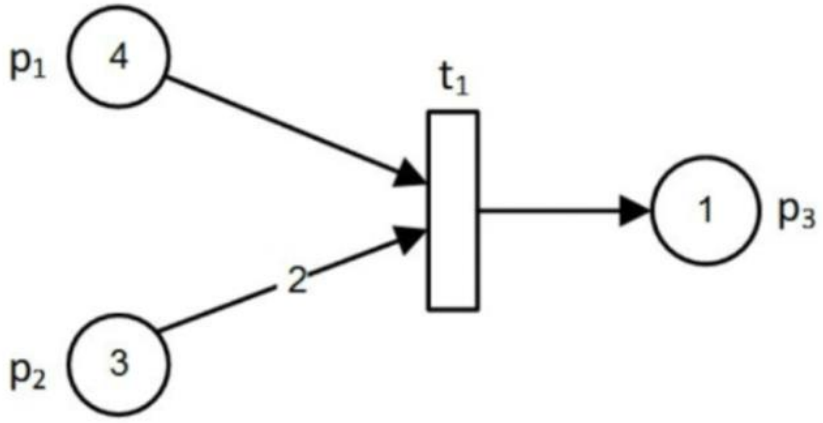

Figure 1: Basic Petri-Net Model (Image Credit: Devidrajuh, R) [1]

1.1 Problem Definition

In the bustling environment of a fast-food restaurant, the primary goal is to ensure optimal utilization of resources and time, thereby enhancing overall operational efficiency and customer satisfaction. However, a prevalent issue faced by many restaurant owners is the lack of efficient resource management. Often, schedules are created without meticulous consideration of time and resource allocation, resulting in ineffective service delivery to customers. [2]

Mismanagement leads to various operational challenges, including insufficient staffing, which forces overworked employees to hastily attend to customers, potentially leading to service delays and dissatisfaction. Furthermore, effective inventory management is identified as a critical component for optimizing customer satisfaction, supplier capabilities, and production scheduling. Inadequate

control over inventory often results in shortages of essential raw materials, particularly during peak hours, compromising the restaurant's ability to promptly fulfill customer orders. [3]

To address these pressing challenges, this project focuses on devising a comprehensive strategy for the efficient allocation of staff during regular and peak hours, as well as timely restocking of inventory. The objective is to determine the required resources and time for every operation, ranging from food preparation to the timely replenishment of ingredients. The development of this project is facilitated using the GPenSIM toolbox in MATLAB, enabling a systematic approach to optimizing resources and time management in the fast-food restaurant industry.

2. Method and Design

2.1 Overall Design

In this modernized fast food restaurant development project, we utilize a Petri Net model, depicting transitions as rectangle shapes, places as circle shapes, and arcs as arrows. The primary objective of this model is to showcase how customers are serviced in various sections of the restaurant and how efficiently resources are used. Customers move between different areas of the restaurant through transitions that are activated based on predefined times and conditions. The implementation of this project is facilitated by the GPenSIM toolbox, a discrete-event system modeling tool for MATLAB, developed by Reggie Davidrajuh. [1]

2.2 Modular Design

The primary focus of this project is to revolutionize the customer experience within the fast-food restaurant, emphasizing both the order placement process and the food preparation stages and efficiently resource uses. With the aim of modernizing the restaurant, we have introduced additional sections such as a charging station and a dedicated kids' zone. To ensure a comprehensive and efficient design, we adopted a modular approach, enabling the integration of diverse functionalities seamlessly. These modules are interconnected via an Inter-Module Connection (IMC) framework, allowing customers to navigate freely between different sections or exit the restaurant as needed.

2.3 Techniques

2.3.1 Using Timed Petri-Net

This project employs a time-based Petri Net framework where both customers and resources are actively involved in various modules within predefined time intervals. Both resources and customers engage in activities for specific durations,

representing the firing time of transitions. Several key scenarios from this project are outlined below:

- To ensure the availability of ingredients for burger preparation, a system monitors ingredient levels over time. When the quantity falls below a predefined threshold, a refill process is initiated, calling upon the Extra Staff of the restaurant. The Extra Staff requires a specific timeframe to complete the ingredient replenishment task the purpose of taking order resource is used here. At this time the resource is counter staff which can complete the task according to the given time.
- The order placement process involves the use of a counter staff resource. This staff member handles customer orders within a designated time frame.
- The Timed Petri Net model also facilitates customer engagement in various activities across different sections of the restaurant, such as device charging, restroom usage, and table occupancy. These time-bound customer activities are integrated into the Timed Petri Net model, enhancing the overall functionality of the system.

2.3.2 Using Resources

In the development of this project, we have integrated several resources that require external input. Notably, the utilization of Kitchen staff, Counter staff, and Extra staff is essential, as they perform various assigned tasks within the system. It is crucial to ensure that resources utilized by transitions within modules are released in the module's modular post-processor or the transition's specialized post-processor, as they cannot be released in the COMMON_POST processor.

Several tasks within our report that necessitate the involvement of resources are highlighted below:

- Kitchen Staff is deployed as a resource for burger and drink preparation
- Extra Staff is designated as the resource for all types of refilling tasks
- Counter Staff serves as the resource for order-taking procedures.

2.3.3 Using Colors

Color is employed to attribute specific characteristics to tokens, enabling them to traverse designated transitions based on their assigned color, facilitating the execution of particular activities. The utilization of color is elaborated in various segments of this project:

- Color is utilized to differentiate between customers opting for either Dine-in or Takeaway services during the order placement process.
- The use of color is instrumental in the selection of games for kids, enabling differentiation between those choosing to play on the jumping or sliding equipment, based on their respective colors.

2.3.4 Capacity

In this project, we have implemented diverse capacities within different sections. In specific scenarios, when the capacity threshold is reached, the preceding transition is disabled with predetermined conditions outlined in the pre-file of that transition. Additionally, in certain instances, when the places attain their maximum value, bypass transitions are activated, allowing customers to bypass specific modules without availing their services. This technique is applied across various sections of the project.

2.3.5 Distribution

Customers are randomly assigned to various modules, with certain factors considered to adjust the probability of a client entering or exiting a specific module. This approach enables the modeling of diverse customer behaviors, accommodating the manipulation of these limiting factors.

2.3.6 Cost Calculation

The cost or income associated with the firing of a transition is calculated by multiplying the respective value with the total number of times the transition has been fired during the operational hours, allowing for the determination of costs and revenues over the entire duration. Additionally, daily labor and fixed costs are factored into the calculations.

3. Implementation

In this project, the system comprises six distinct modules: Entrance, Counter, Table, Kids Zone, Toilet, and Charging Station, all interconnected through the Integrated Management Console (IMC).

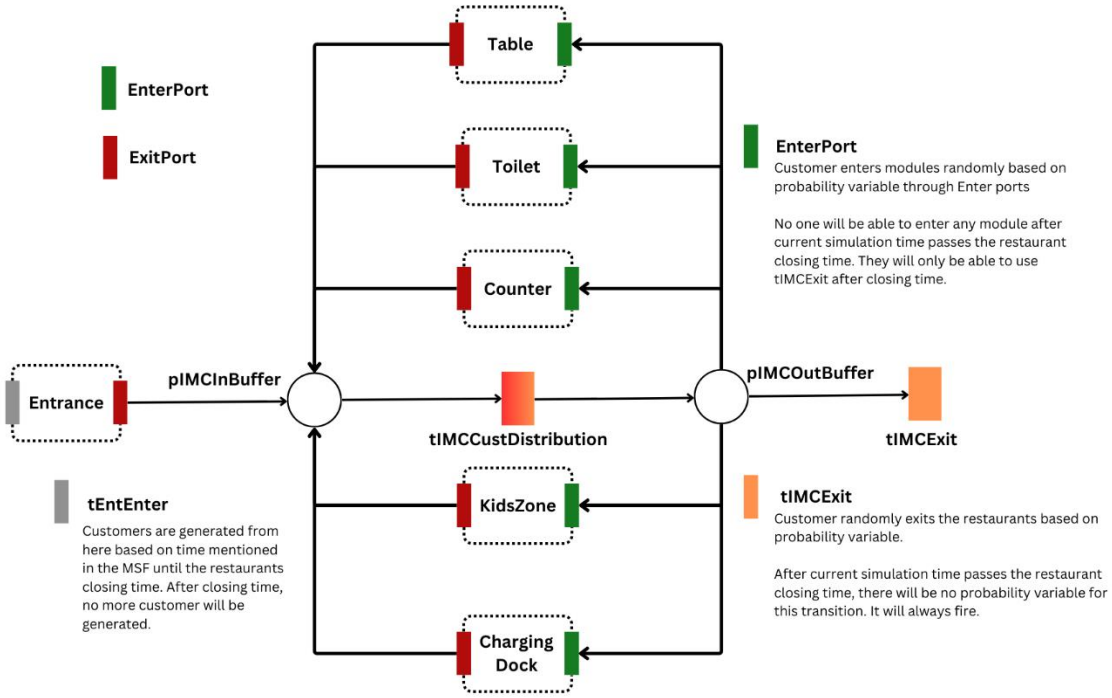

Figure 2: Global Petri Net

Figure 2 illustrates the interplay among these modules within the overarching structure of the global PetriNet.

3.1 Module

In all module there are ports for enter and exit from the module. But there is an exception in the entrance module that entrance module only generates customers and customers can go IMC module. After exiting nobody can enter entrance module. All transitions name is initiated with “t” and place name “p”.

We have created some temporary places and transitions, for better graphical view of analysis and calculations, but these are not core part of our model. That is why we didn't include those places and transitions in the model diagrams.

3.1.1 Entrance

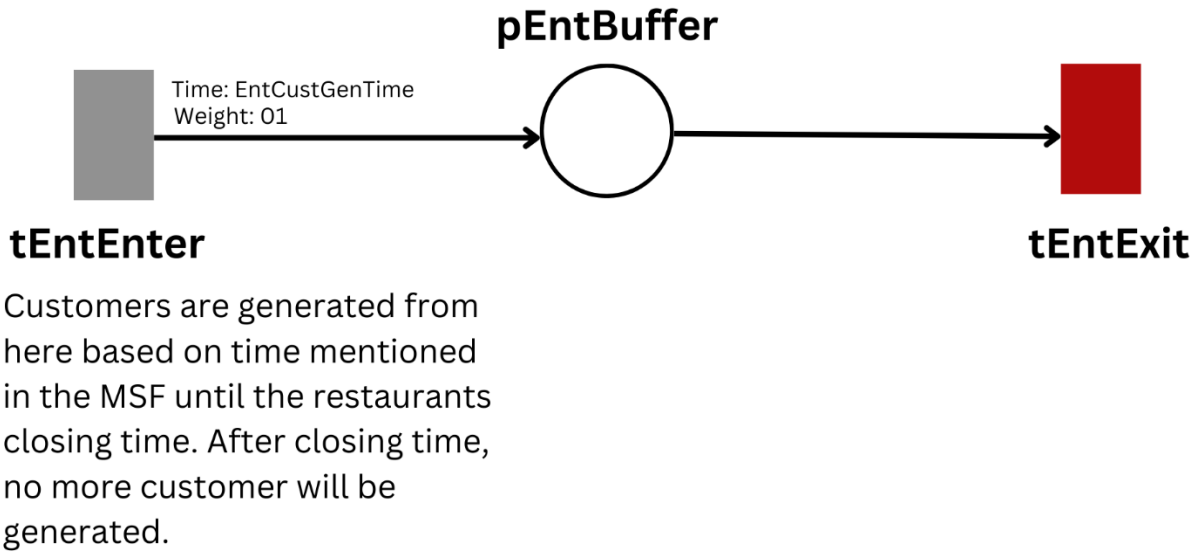

Figure 3: Entrance Module

Entrance module generates all the customers. **tEntenter** generates customers in every predefined time mentioned in global variable **EntCustGenTime** in MSF. But customers are only generated till the restaurant closing time. After closing time, **tEntenter** will stop firing, thus no more customer will be generated. After being genererated, customer enters **pEntBuffer** and then exit the module through **tEntExit**.

3.1.2 Counter

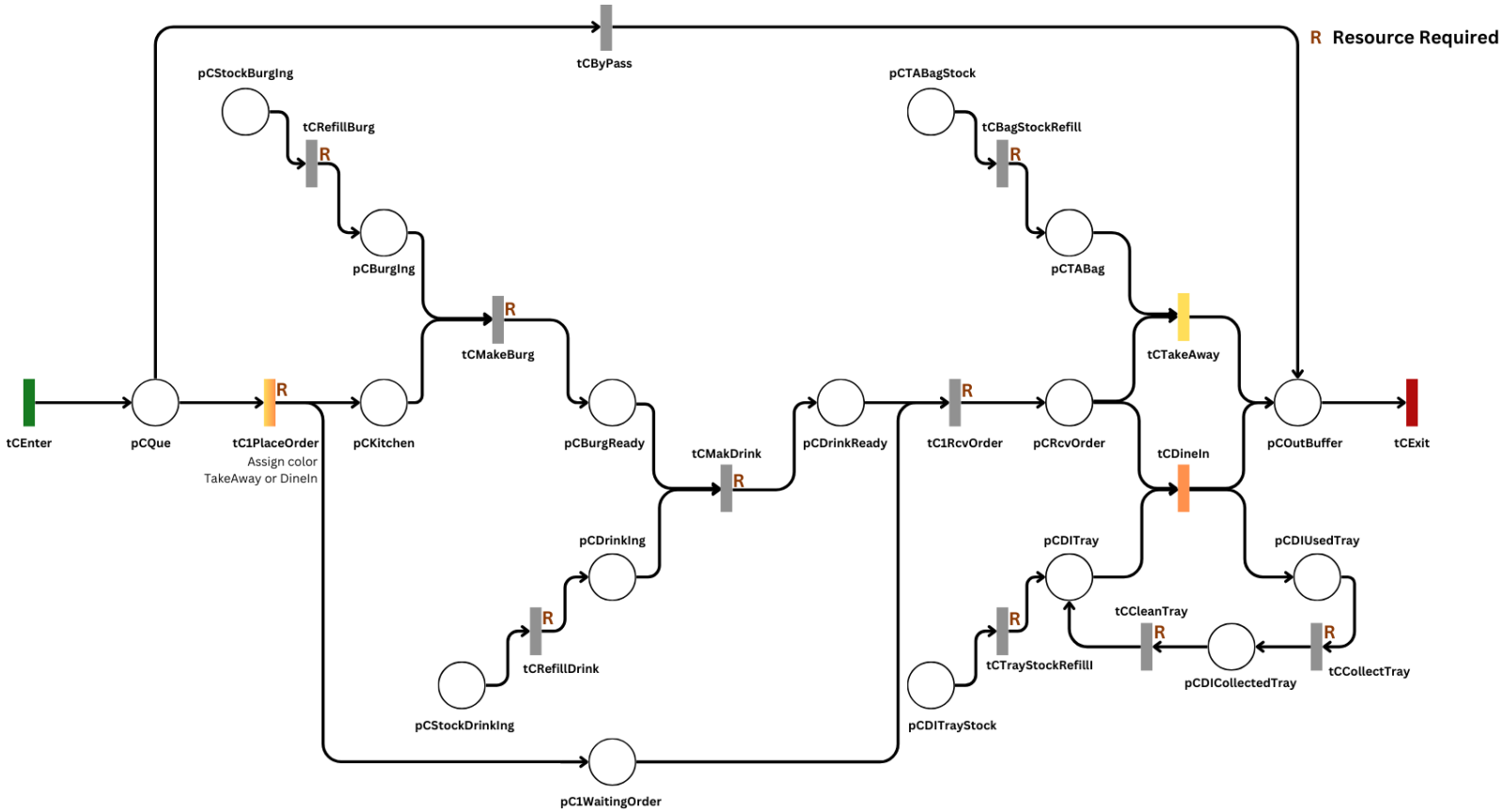

Figure 4: Counter Module

We have set up some initial conditions as predefined variable in the simulation file to run this module. Based on the number of order counters are being used, our Counter module has two different models.

In this module, we are setting how many order-counters should be used based on customer arrival time. We have set up a threshold value, if the customer arrival time is less than that, then it means too many customers are coming in within a short period of time and the restaurant is getting busy. So, one more counter will be used. If customer arrival time is higher than threshold value, then it means it is taking long time for a customer to come in, so only one counter is enough. Based

on order-counters, we are deciding how many resources should be used and the firing time of different transition in Counter module.

We are using Counter Staff, Burger Staff, Drink Staff, Extra Staff as resources for the execution of different transitions in this module. Customers, customer orders, order ingredients, order delivery equipment (takeaway bags or dine in trays) are used as tokens here.

The work flow of counter module is described below,

A customer enters the module through tCEnter and goes to the waiting queue pCQue. This waiting queue has a maximum capacity which has been defined in the global variable CQueCap in MSF. If the customer number in pCQue exceed this maximum capacity, exceeded customer will be removed from waiting queue to the output buffer pCOutBuffer through a bypass channel tCByPass. Otherwise, customer proceeds forward from pCQue to place their food order.

Now, based on customer generation time, we are deciding how many order-counters should be used. If customer generation time which has been defined in the global variable EntCustGenTime, is higher than a predefined new counter threshold CNewCounterThreshold, then there will be only one counter. Otherwise, two counters will be used. For example, if EntCustGenTime is 5 minutes and CNewCounterThreshold is 8 minutes, that means in every 5 minutes one customer is entering the restaurant and we need two order-counters as our new counter threshold was 8 minutes. If customer generation time was 8 minutes or higher, only then one order-terminal would have been used. Now based on order-counters, customer will go to tC1PlaceOrder if there is only one counter or either in tC1PlaceOrder or tC2PlaceOrder if there are two counters to place food order with

the help of Counter Staffs. But this transition will not fire if burger ingredients stocks in pCStockBurgIng become low. This low value has been defined in the global variable CustRedirectBOBurgStock. This condition was set so that there is no customer with pending order while burger making ingredients are out of stock. In this case, they will be redirected from pCQue to tCByPass to pCOutBuffer. This condition is also mentioned in tCByPass.

Now, after coming to tC1PlaceOrder or tC2PlaceOrder, a color will be assigned randomly to decide whether the order is for TakeAway or DineIn. After ordering food, customer goes to waiting state which is marked as pC1WaitingOrder or pC2WaitingOrder based on which order counter has been used. While customer is waiting, order is moved from tC1PlaceOrder or tC2PlaceOrder to pCKitchen.

Every order has a burger and a drink by default. So, kitchen staff makes the burger in tCMakeBurg transition which takes a certain amount of time mentioned in the global variable CBurgMakeTime and after the burger is ready drink is being prepared in tCMakeDrink transition by kitchen staff which takes CDrinkMakeTime.

Each burger and each drink require a certain amount of ingredients mentioned in CBurgIngReq and CDrinkIngReq respectively. These ingredients are being used from pCBurgIng and pCDrinkIng. These places have a limited stock and this stock amount have been mentioned in the global variable CBurgIngSmlStock and CDrinkIngSmlStock. Whenever these stock value, decreases up to a certain trigger point CBurgRefillTriggerMark and CDrinkRefillTriggerMark, extra staff will refill this stock from pCStockBurgIng and pCStockDrinkIng by using the transitions tCBurgStockRefill and tCDrinkStockRefill. We have set the refill time for all kind of refilling in CRefillTime variable.

Now, when drink is ready, customer who is in waiting state will receive the order with the help of counter staff from tC1RcvOrder or tC2RcvOrder based on which order counter has been used. If the order is TakeAway, it will go through the tCTakeAway transition and will receive a TakeAway Bag. If the order is DineIn, then it will go through tCDineIn transition and will receive a DineIn Tray. This takeaway bag and dinein tray stock follows the same stock and refill functionality as burger and drinks ingredients stocks. But for DineIn tray, whenever it is used it will be collected by extra staff after used tray reaches the amount CDITrayCollect. Collected dirty trays will be kept stacking until it reaches CDITrayClean point and then it will be sent for cleaning. After cleaning, it will be stocked again in pCDITray. After receiving order from tCTakeAway or tCDineIn, customer will go to the pCOutBuffer and from there they will exit the module through tCExit.

3.1.3 Toilet

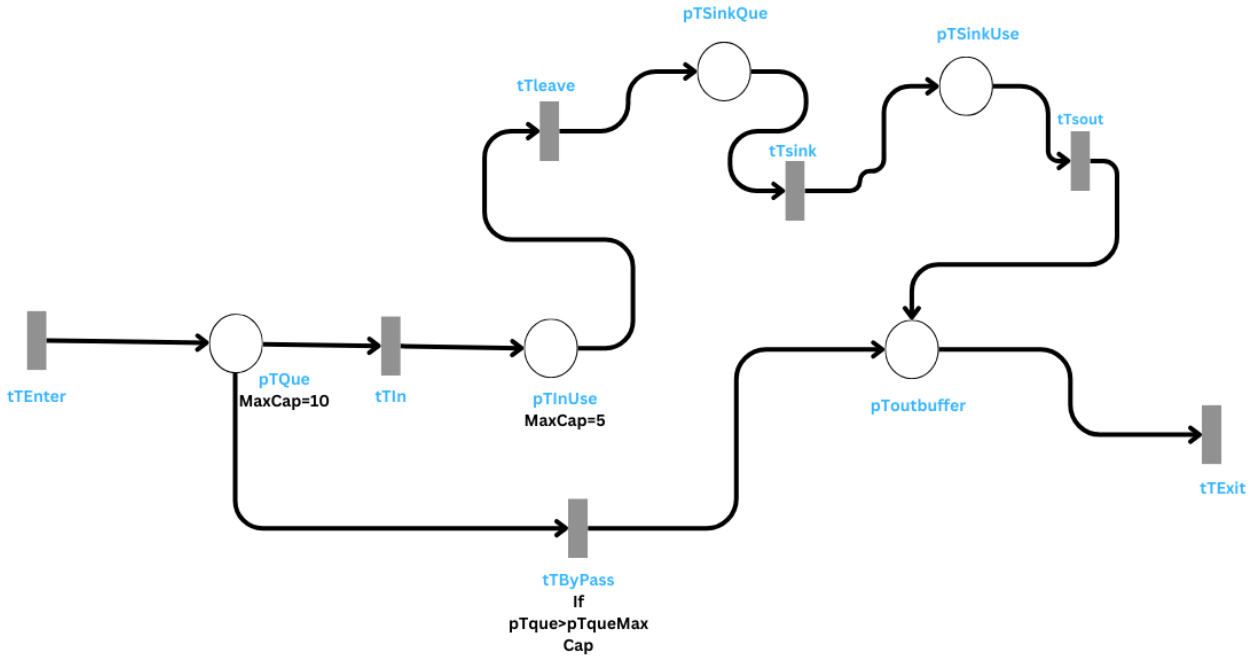

Figure 5: Toilet Module

Customers enter the toilet module through the designated "tTEnter" port, gaining access to the facility. Upon entry, they are directed to the waiting area known as "pTQue" where they await their turn to use the toilet facilities. If the number of customers in the "pTQue" area reaches its maximum capacity, the system activates the "tTBypass" mechanism, diverting additional customers to the designated "pToutbuffer" area to prevent overcrowding.

On the other hand, if the "pTQue" area has not reached its maximum capacity, customers are permitted to transition to the designated "pTInUse" area through the specified "tTIn" pathway. However, once the "pTInUSe" area reaches its maximum capacity, the "tTIn" transition is automatically disabled in accordance with the predefined conditions.

Upon entering the "pTInUse" area, customers are allowed to leave every 5 minutes via the "tTleave" transition, enabling them to join the queue for the sink in the "pTSinkQue" area.Once inside the "pTSinQue" area, customers can access the sink via the "tTSinkIn" transition. Each customer is allotted a two-minute interval for

handwashing before they proceed to the "pTOutBuffer" area through the "tTSinkOut" transition. Finally, customers can exit the entire toilet module using the "tTExit" transition.

3.1.4 Table

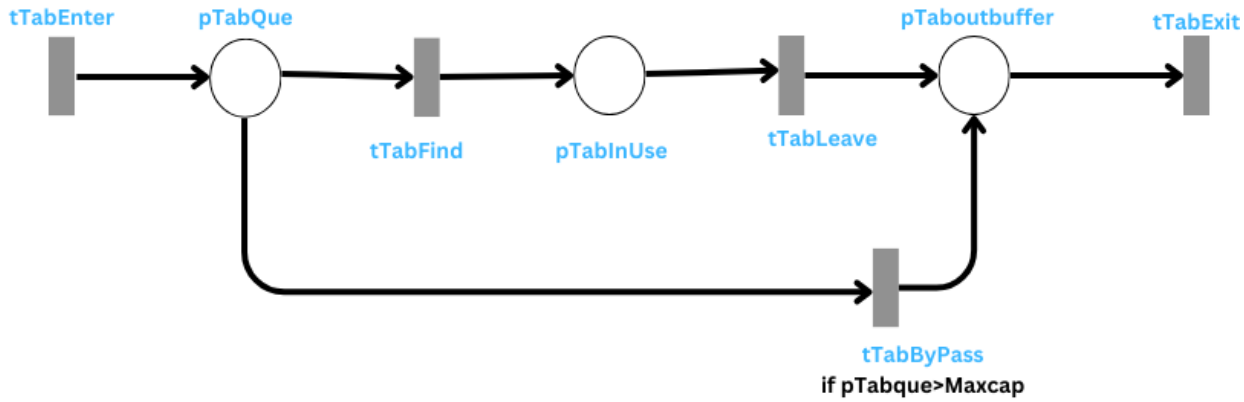

Figure 6: Table Module

Within the table module, customers can access the facility through the tTabEnter port and proceed to the designated pTabQue area. Once the pTabQue area reaches its maximum capacity, the tTabBypass mechanism is activated, allowing customers to be redirected to the pTabOutBuffer section. Conversely, if the pTabQue area is not at its maximum capacity, customers can utilize the tTabFind service to locate their designated table within the pTabInUse area. Customers are allowed to spend up to ten minutes in the pTabInUse area, after which they will be directed to leave through the tTabLeave exit and proceed to the pTaboutbuffer section. From the pTabOutbuffer area, customers can ultimately exit the Table module through the designated tTabExit port.

3.1.5 Kids Zone

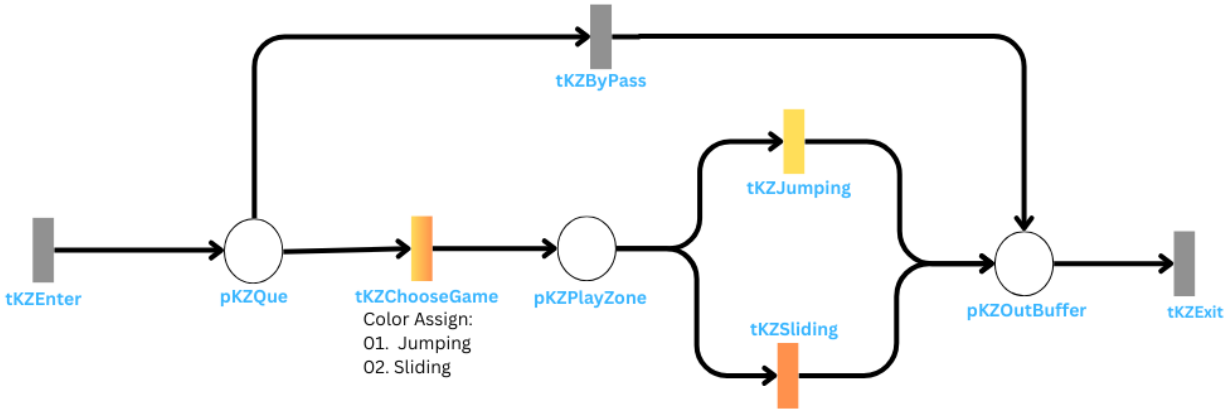

Figure 7: KidsZone Module

Children can access the Kidszone module through the designated tKZEnter port. Upon entry, they are directed to the pkZQue where they can wait for their turn to enter the playing zone. The pkZQue place has a predefined capacity, and when this capacity is exceeded, children are redirected to the pKZOutBuffer place as specified by the preprocessor conditions within the module. Conversely, if the capacity limit is not reached, the children proceed to the tKZchooseGame section where they can select their preferred game, either jumping or sliding. The tKZchooseGame transition also assigns a specific color to each child based on their game preference. After selecting a game, the children enter the pKZPlayzone to engage in the chosen activity. From the pKZPlayzone, children follow the designated pathways, tKZJumping or tKZsliding, based on their assigned color, before eventually reaching the pKZOutbuffer. Finally, the children can exit the Kidszone module through tKZExit port.

3.1.6 Charging Dock

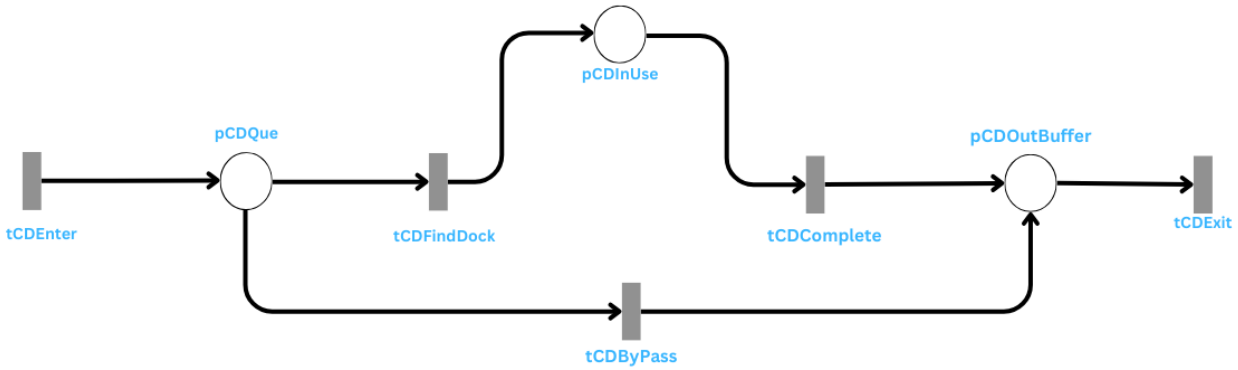

Figure 8: Charging Dock Module

The process at the Device Charging Station involves several key steps to ensure a smooth and efficient experience for customers. Upon arrival, customers can enter the facility through the designated entrance, tCDEnter, and are directed to the charging area. If the charging queue, pCDQue, has not reached its maximum capacity, customers can wait in line for their turn at a charging port. If the pCDQue is at full capacity, customers are redirected to the pCDOutBuffer place via the tCDBypass route. Once customers reach an available charging port, they are guided to find the appropriate charger port by following instructions at the tCDFindDock point. Following successful connection, customers can charge their devices at the pCDInUse area for up to 10 minutes. After the charging period, customers are instructed to proceed to the pCDOutBuffer area via the tCDComplete transitions. Finally, customers can complete their visit by exiting through the tCDExit.

3.1.7 IMC

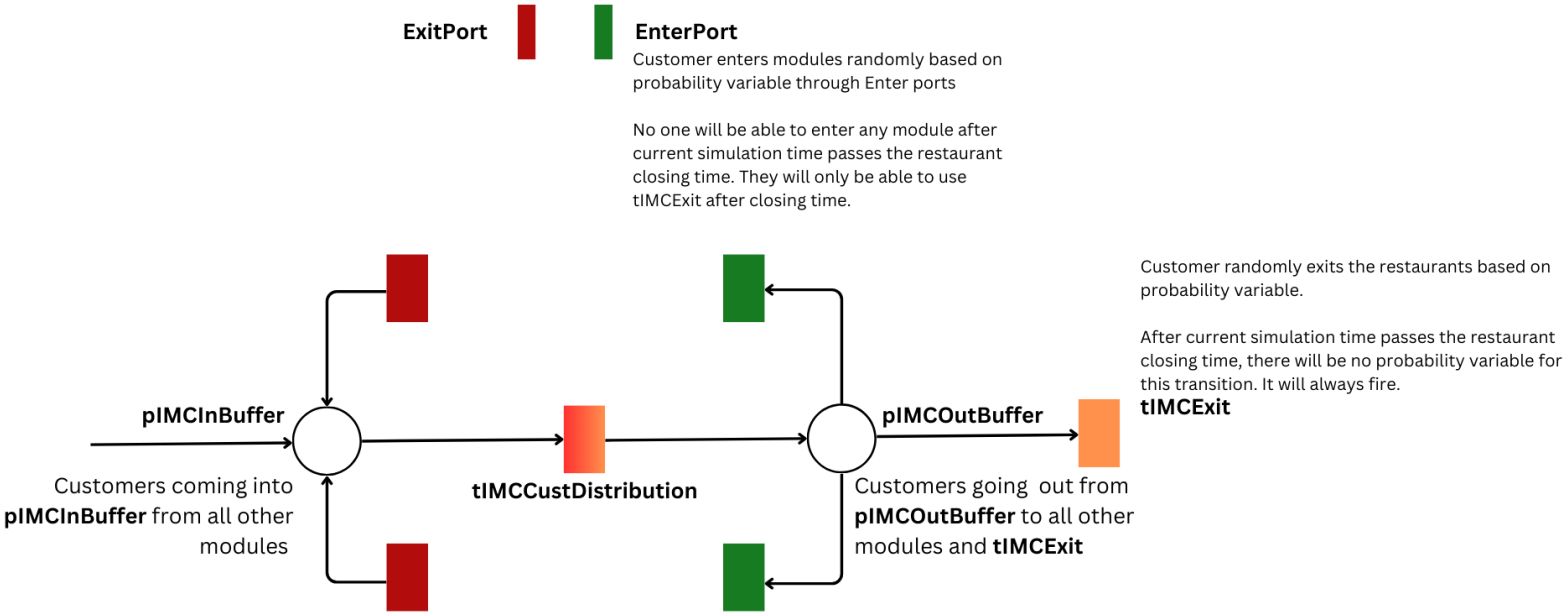

Figure 9: IMC Module

The Integrated Module Controller (IMC) serves as the central hub connecting all other modules within the system. Customers arrive at the system through various modules, including the entrance, counter, table, toilet, kidszone, or charging dock. They enter the IMC via the input buffer called pIMCInBuffer.

Subsequently, customers transition through the tIMCCustDistribution process, eventually reaching the IMC's output buffer, pIMCOutBuff. From there, they have the option to access other modules, such as the counter, table, toilet, kidszone, or charging dock, through their respective input ports.

However, there is a closing hour in place, beyond which customers are no longer allowed to enter any other module from pIMCOutBuff. Instead, they can only exit

the system through the tIMCExit transition. This rule is enforced by the common pre-processor.

To influence customer behavior and determine the likelihood of them choosing a particular module, a ProbabilityVariable system has been implemented. This system controls the probability of a customer entering a specific module based on predefined settings. These probability variables can be adjusted in the main simulation file (MSF), allowing for flexibility in managing customer flow and module usage.

4. Testing, Analysis, and Results

4.1 User Manual

To customize different scenarios, one can modify the parameters in the initial section of the main simulation file (MSF) from their default values, as shown in the code snippets below, to reflect a specific case. In the accompanying figure 10, the starting time, simulation stop time, and closing time are illustrated. The closing time signifies that customers cannot generate after it, and even those present in the restaurant at that time cannot transition to a new module; they must exit through the IMC.

```
clear all; clc;
global global_info

%Simulation Run Time
global_info.START_AT = 0;
global_info.STOP_AT = 300;

%Closing time of Restaurant, after which no new customer will be generated
%and no one will be allowed to enter any other module; but customers who
%are already inside different modules will finish their functionality as
%usual
global_info.ClosingTime = 240;
```

Figure 10: Code Snippet of Run Time

Figure 11 shows how customers will enter different module. Here, we are using randi([1,global_info.VARIABLE]) to determine probability. So, the higher the number, the lesser the probability.

```
%-------------------------Initial IMC Module Conditions----------------------------
% The probability of which module should be used from IMC
% We are using randi([1,global_info.VARIABLE]) to determine probability
% So, the higher the number, the lesser the probability
global_info.CounterEnterProbability = 1;
global_info.TableEnterProbability = 2;
global_info.ToiletEnterProbability = 3;
global_info.KidsZoneEnterProbability = 5;
global_info.ChargDockEnterProbability = 5;
global_info.ExitProbability = 3;
```

Figure 11: Code Snippet of Probability Variable

Initially Customers are generated in every 3 minutes. But it can be changed according to need.

```
%-------------------------Initial Entrance Module Conditions-------------------------
global_info.EntCustGenTime = 3;             %One Customer is being generated in every # Minutes
```

Figure 12: Code Snippet of Customer Generation

Based on customer generation time, order counters can be adjusted and there is maximum capacity in the queue. Burger main stock value can be adjusted based on which we can control customer movement in counter module.

```
%-------------------------Initial Counter Module Conditions-------------------------
%Customer
global_info.CNewCounterThreshold = 5;       %Threshold for when one more counter should be used
global_info.CQueCap = 5;                    %Maximum Capacity of Counter Queue
global_info.CustRedirectBOBurgStock = 250;  %Burger Main stock's critical value based on which customer will be redirected to bypass
```

Figure 13: Code Snippet of Counter Module Conditions

Initial stock conditions can be adjusted from here,

```
%Burger
global_info.CBurgIngReq = 50;              %Amount of ingredients required for a Burger
global_info.CBurgIngSmlStock = global_info.CBurgIngReq * 10;       %Bruger Ingredients Kithcen Stock
global_info.CBurgIngBigStock = global_info.CBurgIngReq * 170;      %Burger Ingredients Main Stock
global_info.CBurgRefillAmnt = global_info.CBurgIngReq * 7;        %Burger Ingredients refill amount from Main to Kitchen Stock
global_info.CBurgRefillTriggerMark = global_info.CBurgIngReq * 3;  %Trigger Point When Burger Ingredients in Kitchen should be refilled from Main Storage

%Drink
global_info.CDrinkIngReq = 50;             %Amount of ingredients required for a Drink
global_info.CDrinkIngSmlStock = global_info.CDrinkIngReq * 10;      %Drink Ingredients Kithcen Stock
global_info.CDrinkIngBigStock = global_info.CDrinkIngReq * 170;     %Drink Ingredients Main Stock
global_info.CDrinkRefillAmnt = global_info.CDrinkIngReq * 7;       %Drink Ingredients refill amount from Main to Kitchen Stock
global_info.CDrinkRefillTriggerMark = global_info.CDrinkIngReq * 3;  %Trigger Point When Drink Ingredients in Kitchen should be refilled from Main Storage

%TakeAway Paper Bag (in pcs)
global_info.CTABagSmlStock = 40;            %TakeAway Bag Kithcen Stock
global_info.CTABagBigStock = 200;           %TakeAway Bag Main Stock
global_info.CTABagRefillAmnt = 35;          %TakeAway Bag refill amount from Main to Kitchen Stock
global_info.CTABagRefillTriggerMark = 5;    %Trigger Point When TakeAway Bags in Kitchen should be refilled from Main Storage

%DineIn Tray (in pcs)
global_info.CDITraySmlStock = 25;           %DineIn Tray Kithcen Stock
global_info.CDITrayBigStock = 50;          %DineIn Tray Main Stock
global_info.CDITrayRefillAmnt = 10;         %DineIn Tray refill amount from Main to Kitchen Stock
global_info.CDITrayRefillTriggerMark = 5;   %Trigger Point When DineIn Trays in Kitchen should be refilled from Main Storage
global_info.CDITrayCollect = 5;             %Trigger point when DineIn Trays should be collected from Used Tray
global_info.CTrayColTime = 1;               %Tray collection time
global_info.CDITrayClean = 10;              %Trigger point when Dirty Trays should be sent for cleaning
global_info.CTrayCleanTime = 2;             %Tray Clean Time
```

Figure 14: Code snippet of Stock Conditions

Figurer 15 shows the resources management based on order counters

```
%Resource Management
if global_info.OrderCounter == 1
    nCounter1Staff = 1;
    nBurgStaff = 1;
    nDrinkStaff = 1;
    nExtraStaff = 1;
else
    nCounter1Staff = 1;
    nCounter2Staff = 1;
    nBurgStaff = 2;
    nDrinkStaff = 2;
    nExtraStaff = 2;
end
```

Figure 15: Code Snippet of Resource Management

Figure 16 shows the queue capacity and availability and usage time of different modules.

```
%-------------------------Initial Table Module Conditions-------------------------
global_info.TabQueCap = 3;                  %Maximum Capacity of Table Queue
global_info.TabAvaialble = 2;               %Total number of Tables
global_info.TabUsageTime = 5;               %Table usage time for a customer

%-------------------------Initial Toilet Module Conditions-------------------------
global_info.TQueCap = 3;                    %Maximum Capacity of Table Queue
global_info.TAvaialble = 2;                 %Total number of Toilets
global_info.TUsageTime = 5;                 %Toitlet usage time for a customer

%-------------------------Initial KidsZone Module Conditions-------------------------
global_info.KZQueCap = 3;                   %Maximum Capacity of KidsZone Queue
global_info.KZAvaialble = 2;                %Total number of Empty Slots
global_info.KZUsageTime = 5;                %KidsZone usage time for a customer

%-------------------------Initial ChargingDock Module Conditions-------------------------
global_info.CDQueCap = 5;                  %Maximum Capacity of ChargingDock Queue
global_info.CDAvaialble = 10;              %Total number of Charging Dock
global_info.CDUsageTime = 5;               %Charging Dock usage time for a customer
```

Figure 16:Code Snippet of Initial Conditions of Different Modules

```
%----------------Cost Variables----------------
% Ingredients Cost
Costs.Burger = 35;
Costs.Drink = 5;

%Employee Cost (Per Minute)
if global_info.OrderCounter == 1
    Costs.EmpCounter1Staff = 5;
else
    Costs.EmpCounter1Staff = 5;
    Costs.EmpCounter2Staff = 5;
end
Costs.EmpBurgStaff = 5;
Costs.EmpDrinkStaff = 5;
Costs.EmpExtraStaff = 5;

%Fixed Cost (Per Minute)
Costs.FixedCosts = 30;

%----------Income Variables-------------------
BurgerPrice = 250;
DrinkPrice = 80;
```

Figure 17: Code Snippet of Cost & Income Variables

Figure 17 shows the cost and income variables which can be adjusted according to our needs

4.2 Sample Run

Our sample run shows one of the profitable simulations run, where we have enough customers coming into the restaurant, forcing us to use two order counters and lead to profitability.

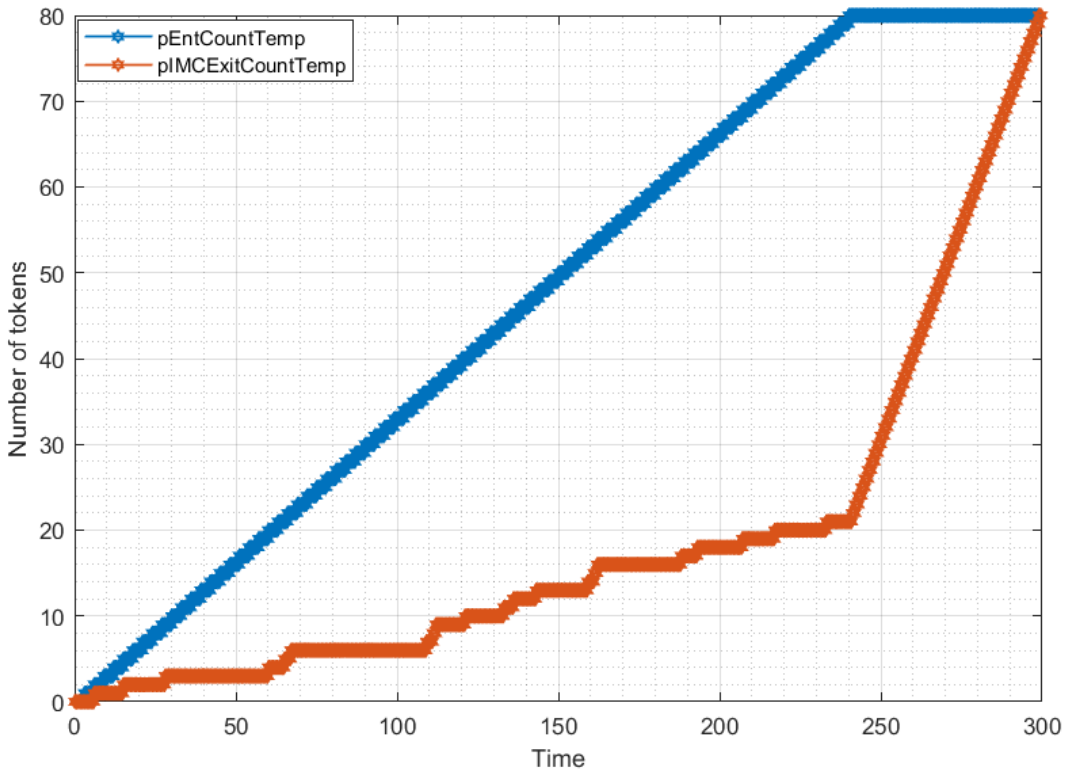

Figure 18: Total Customer Generated and Exited

Figure 18 shows how many total customers have been generated during the whole simulation time and how many have exited the module. Here, we can see after time reaches 240 which is mentioned as closing time in MSF, no more new customer is being generated and all the other remaining customers starts exiting as no one can enter any other module after closing time. The simulation runtime is from 0 to 300 as per this test run. Total generated customer number is 80 and all of them have exited eventually.

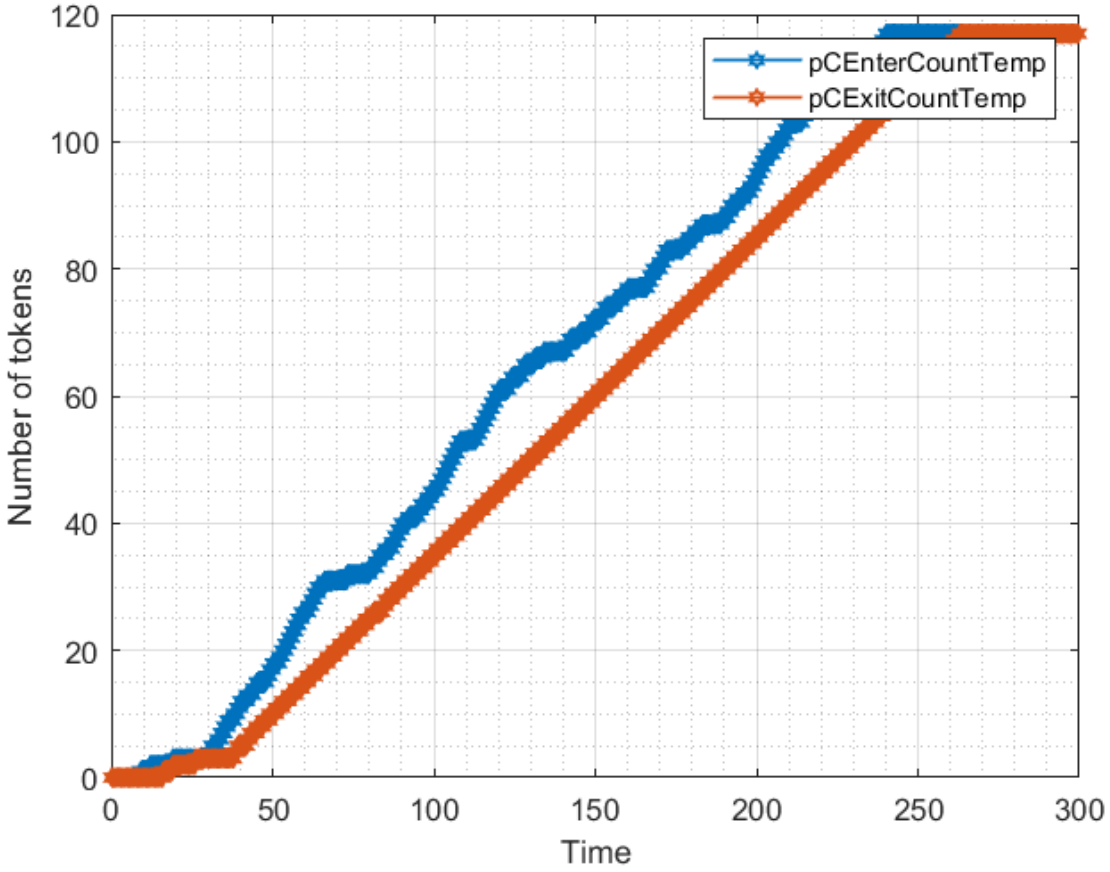

Figure 19: Total Number of Customer in Counter Module

Figure 19 shows the total number of customers who have entered the counter module. From figure 18 we can see that our total generated customers were 80, but here our total in and out customer number is 117. Because after being generated from entrance module, they can enter other module multiple times based on probability variables. Here, customers enter the module until 240 as mentioned in the previous figure and after this closing no one enter the module. But those who are already in this module, will be served accordingly and will eventually exit the module. This closing time scenario applies to rest of the modules as well (Table, Toilet, KidsZone and Charging Dock).

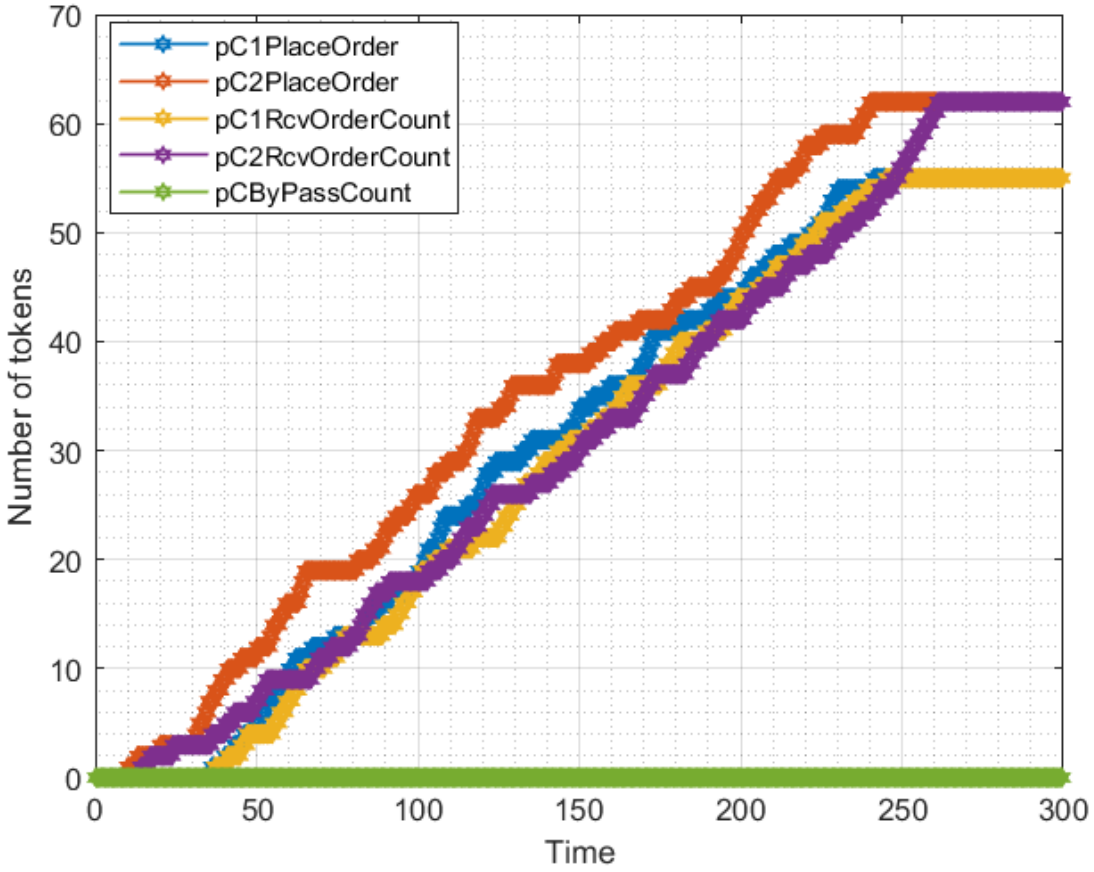

Figure 20: Total Order Placed and Received in different counters

This figure shows how many orders have been placed in different order counters and how many have been received from their respective receive counters. We can see total 55 orders have been placed and received in counter 01 and total 62 orders in counter 02. So, the total number of orders is 117 including both counters which matches the total number who entered counter module as mentioned in figure 19. We can also see that total by pass count is 0, that means there was not enough customer in queue than our maximum queue capacity. It also means we had enough burger ingredients in stock to serve 117 people. If we had not, customers would have gone through bypass to output buffer. But as per our test run, we have enough ingredients to serve 170 customers.

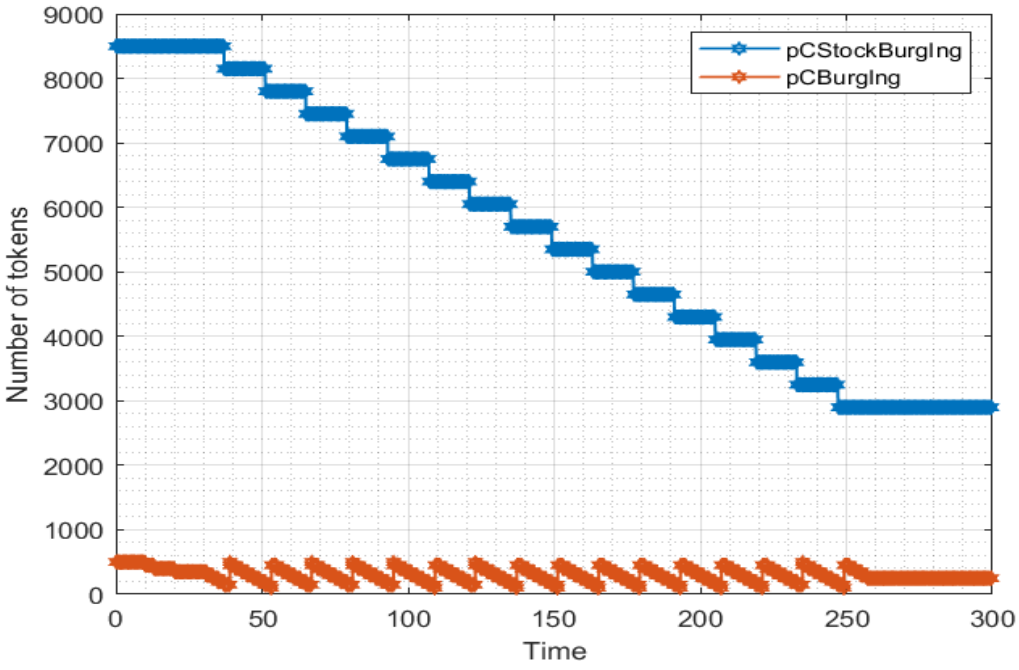

Figure 21: Burger Ingredients Stock

Figure 21 shows the usage of Burger ingredients stock and their refilling time and amount. This same thing has been done for drinks ingredients stock as well as shown in figure 22.

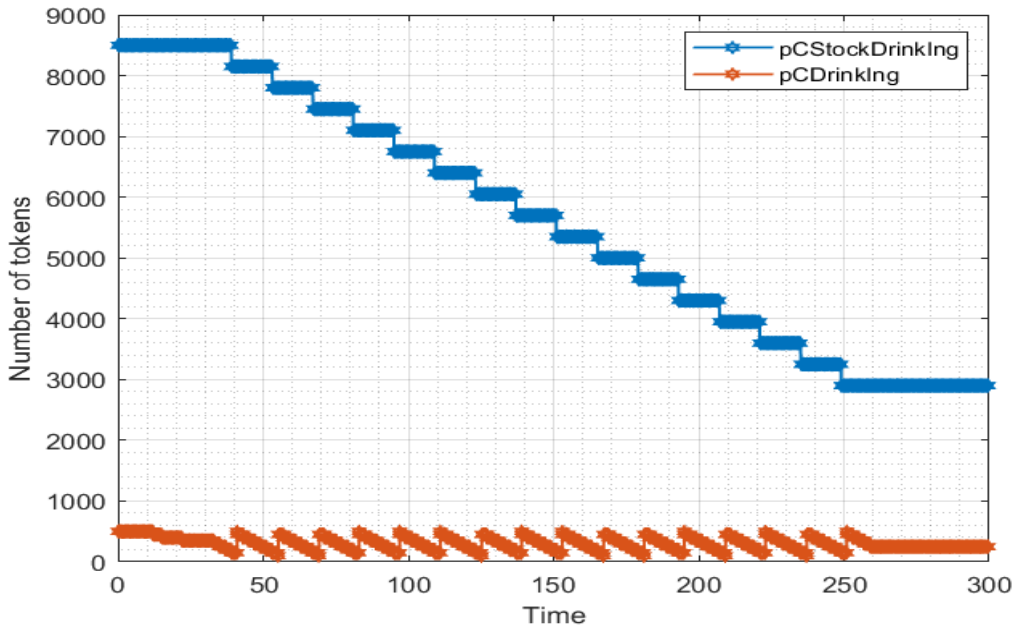

Figure 22: Drinks Ingredients Stock

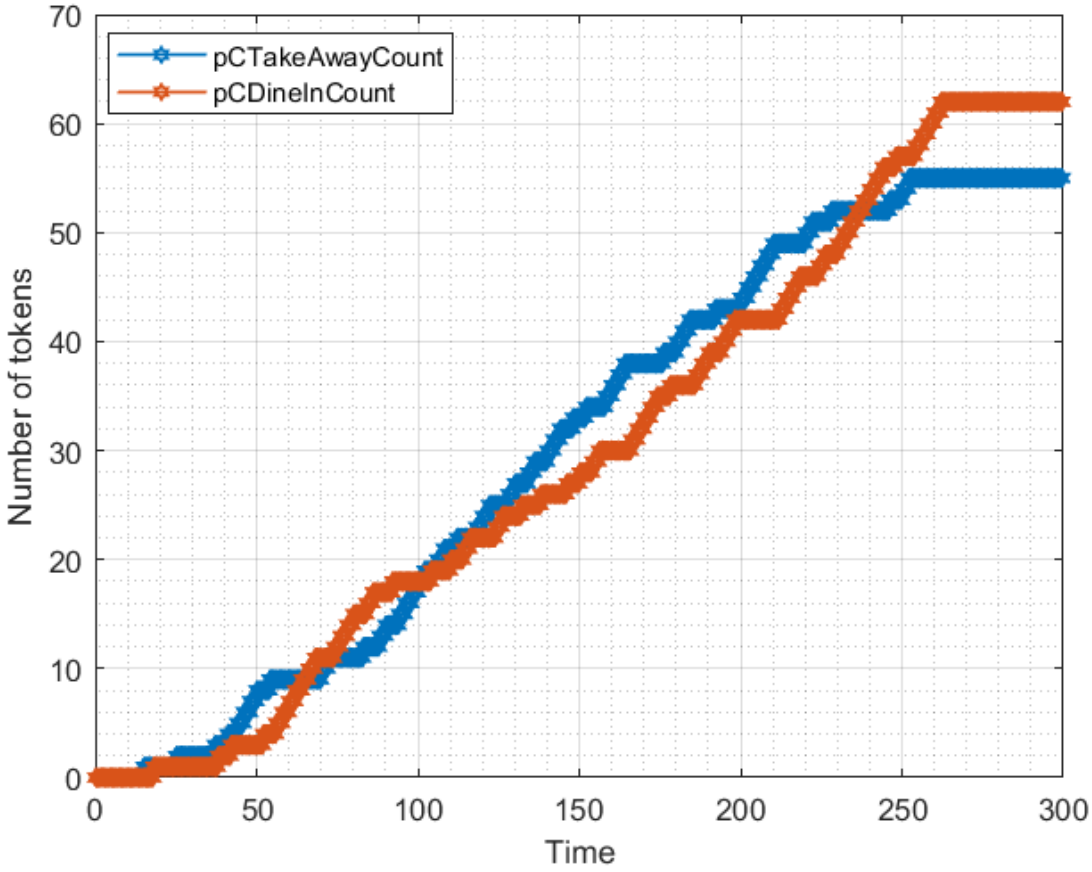

Figure 23: Total Number of TakeAway & DineIn Orders

From figure 23, we can see that out of 117 orders, 62 orders have been colored as dine in and rest of the 55 orders have been colored as take away.

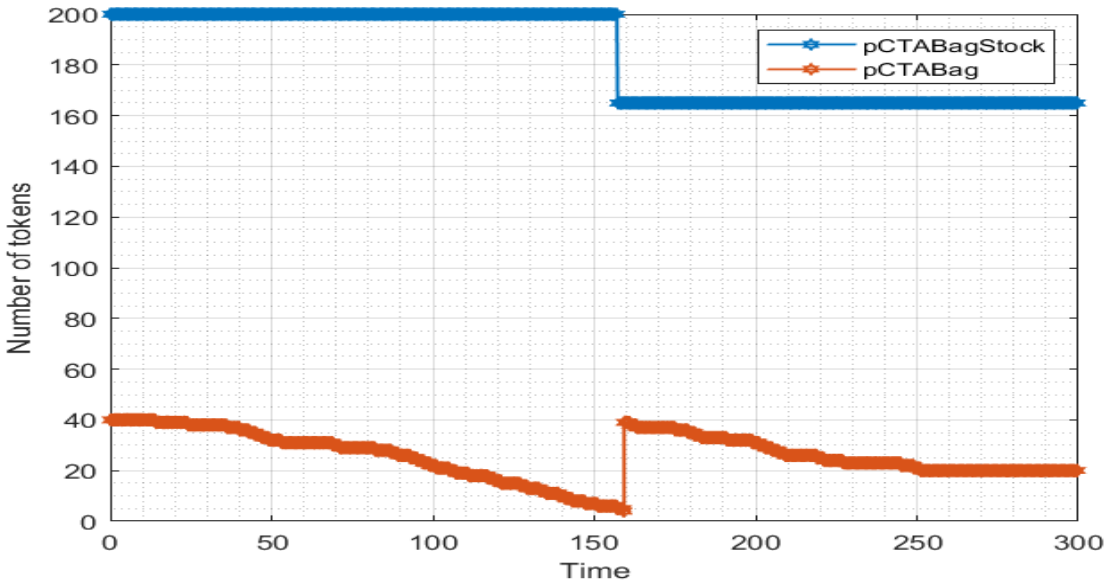

Figure 24: TakeAway Bag Stock

From figure 24 and 27, we can see the stock status of takeaway bags and dine in trays. In dine in tray stock, nothing was refilled from there as we are reusing used dirty trays after cleaning and putting it back in the PCDITray. That is why only PCDITray value is changing over time but not the main stock value.

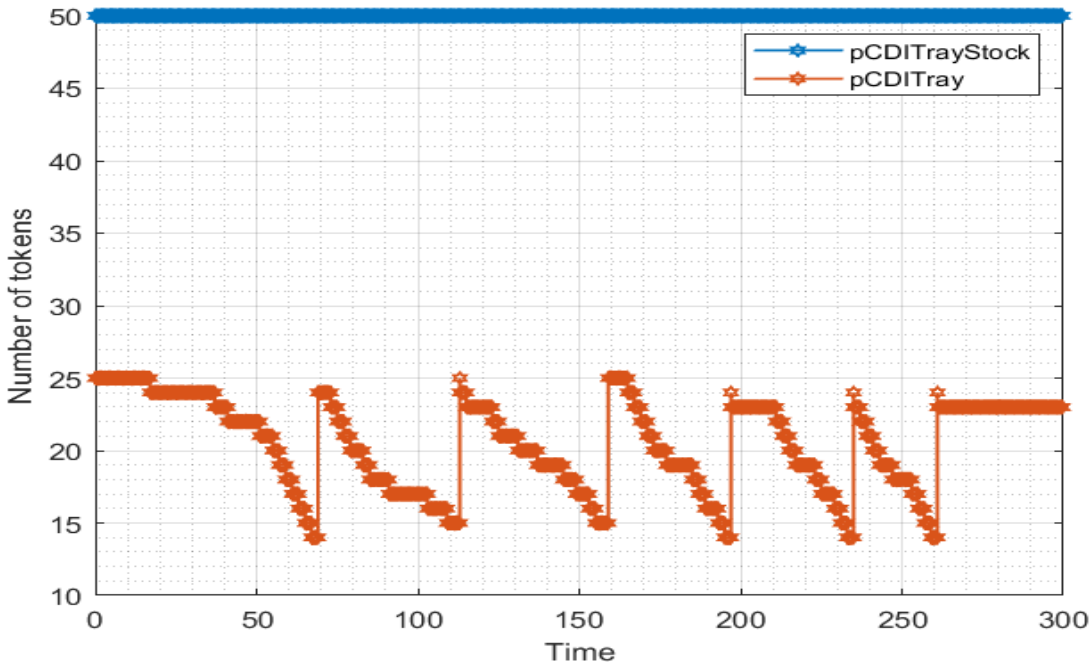

Figure 25: Dine-In Tray Stock

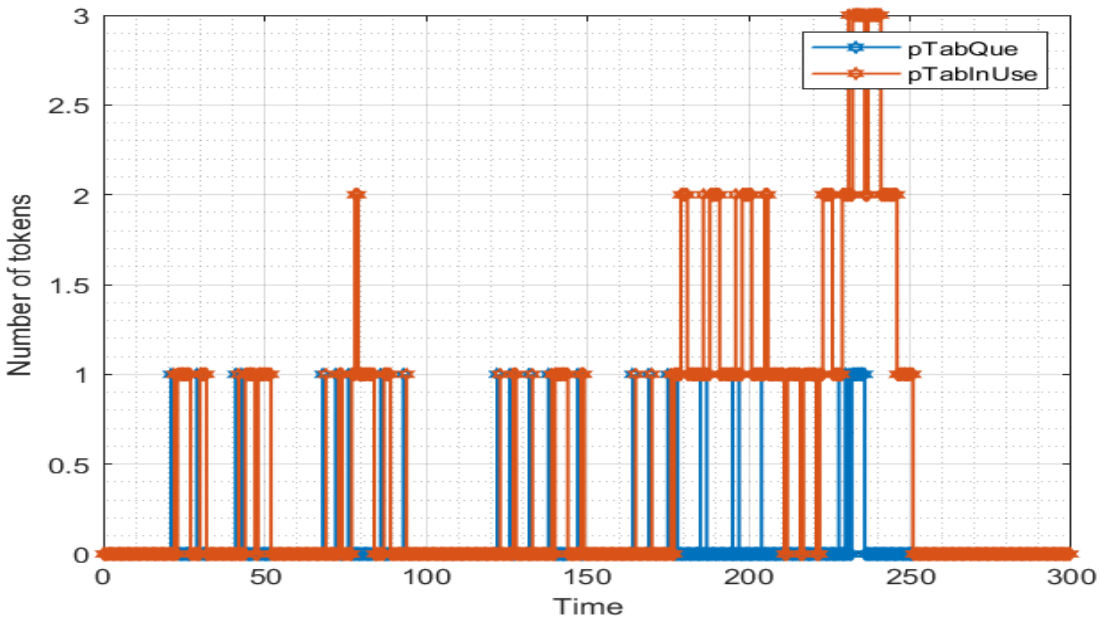

Figure 26: Number of Tokens in Table Module

Figure 26 shows how many customers are in queue and how many customers are using tables.

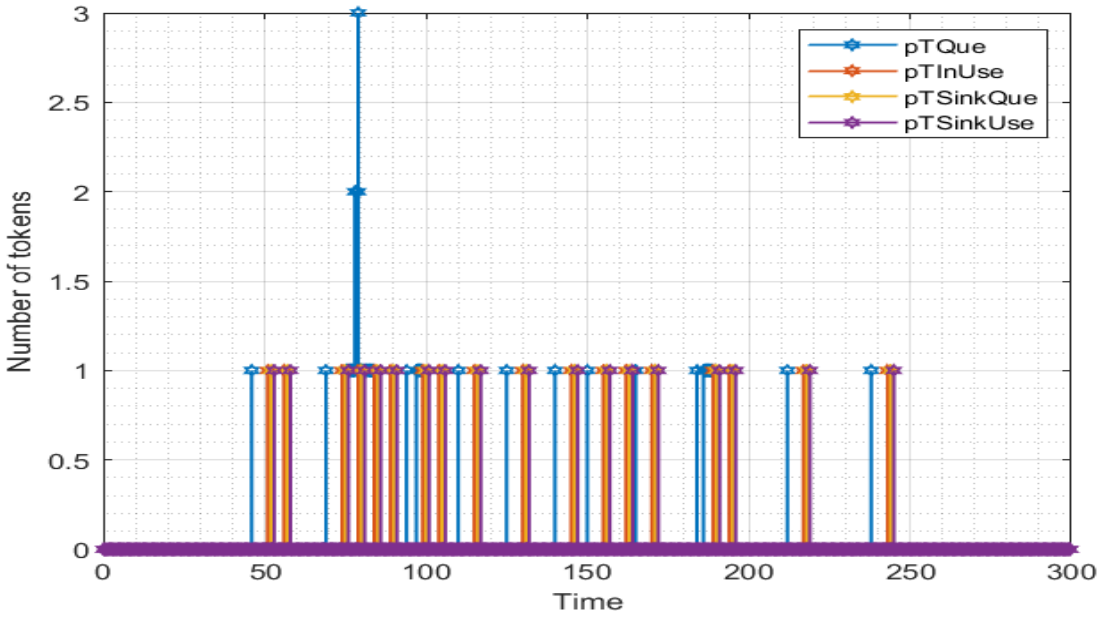

Figure 27: Number of Token in Toilet Module

Figure 27 shows how many customers are in toilet queue and sink queue and how many of them are using those facilities.

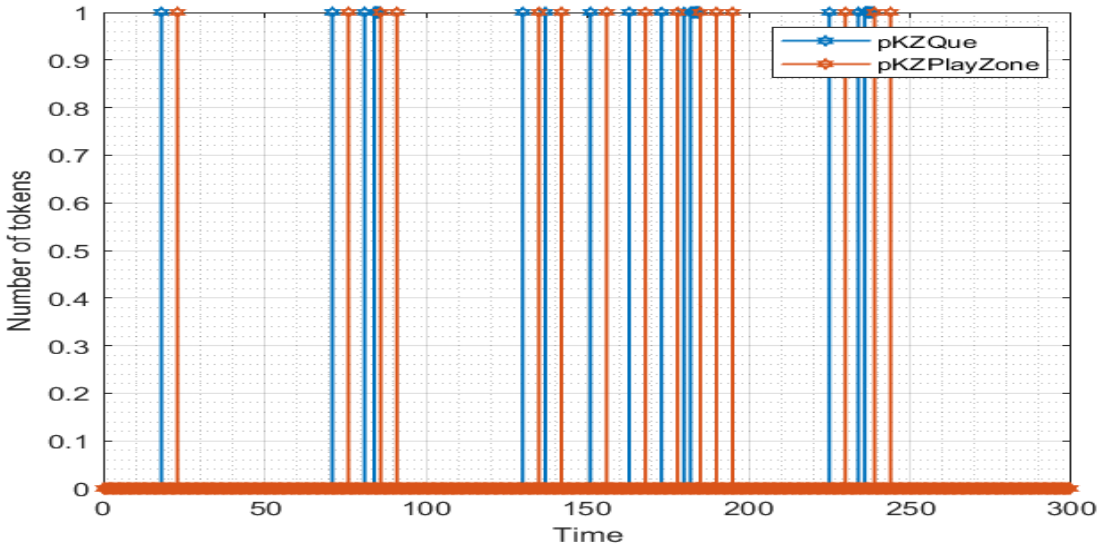

Figure 28: Number of Tokens in KidsZone Module

Figure 28 shows how many customers are in kids zone queue and how many customers are using them.

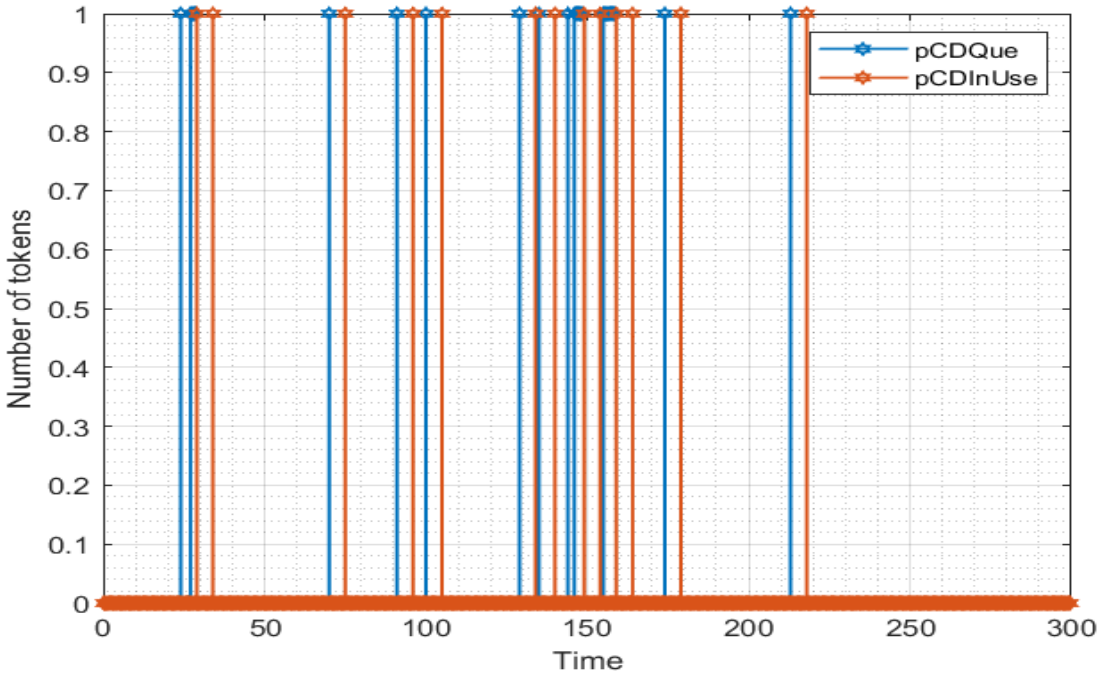

Figure 29: Number of Tokens in Charging Dock Module

Figure 29 shows how many customers are in charging dock queue and how many customers are using them.

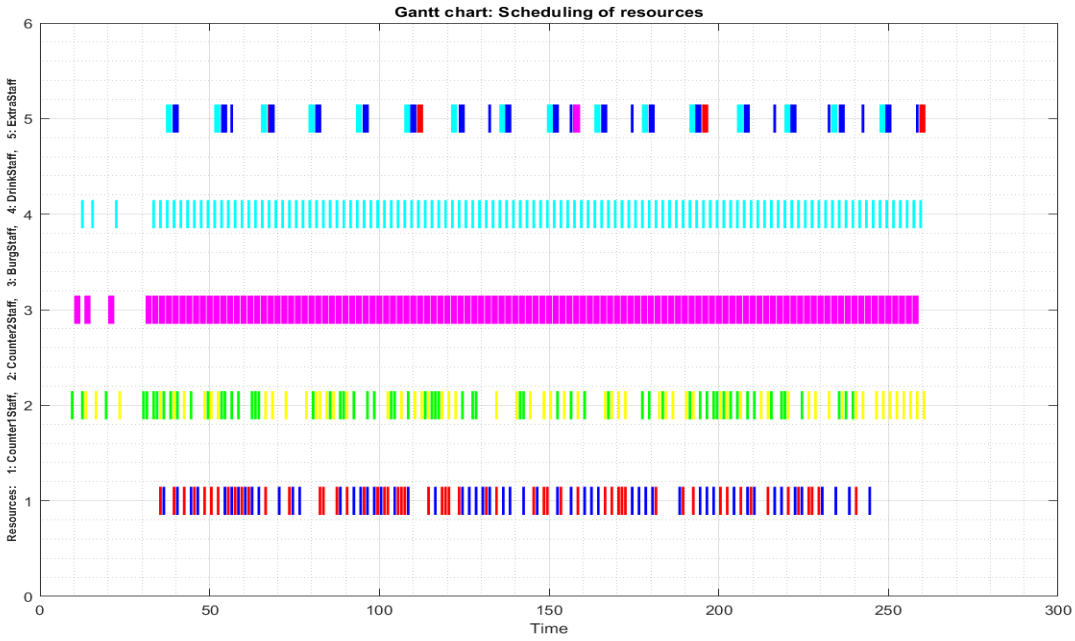

Figure 30: Resource Usage according to Gant Chart

Figure 30 shows the usage of different resources at different times. In our module, based on counter number we have used different kind of resources.

```
RESOURCE USAGE SUMMARY:
Counter1Staff:  Total occasions: 110   Total Time spent: 110
Counter2Staff:  Total occasions: 124   Total Time spent: 124
BurgStaff:  Total occasions: 117   Total Time spent: 234
DrinkStaff:  Total occasions: 117   Total Time spent: 117
ExtraStaff:  Total occasions: 51   Total Time spent: 90
```

Figure 31: Resource Usage Summary from GPenSim

We can see from the resource usage summary that our Counter1Staff has been used 110 times and Counter2Staff for 124 times. If we look at our figure 20, we can see that total 55 orders were placed in Counter 01 and 62 in Counter 02. So from counter 01, Counter1Staff was used while placing order 55 times and while receiving order 55 times, so in total 110 times. Same goes for Counter2Staff. And we had total 117 orders, and that's why BurgerStaff and DrinkStaff were used for 117 times each. ExtraStaff were used in all kind of refilling purposes and to collect and clean dirty trays.

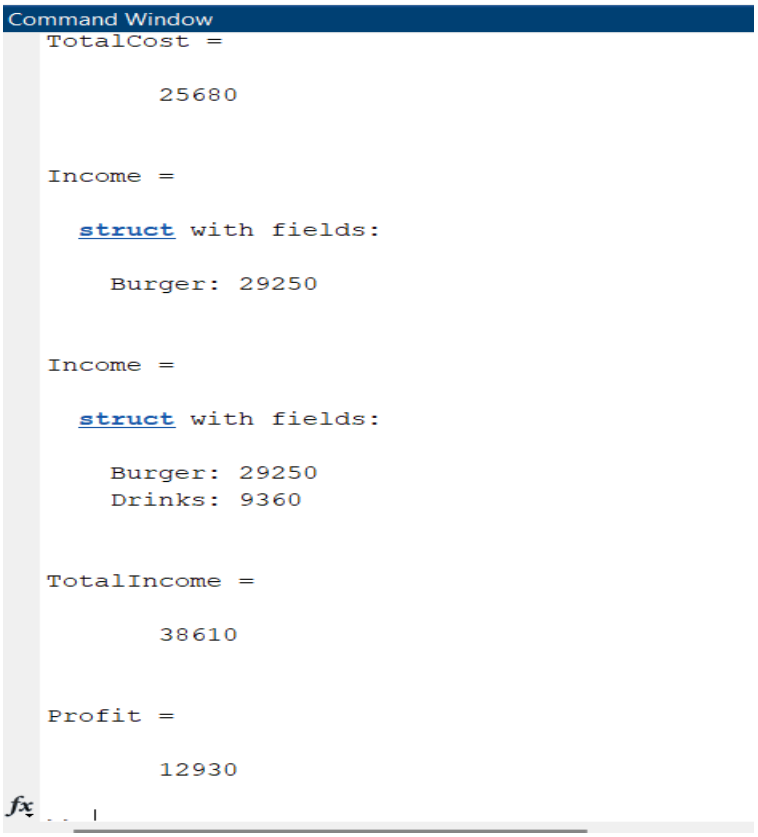

Figure 32: Profit Calculations

Based on our pre-defined cost and income variables and our simulation run time, we can that this test run made a profit of 12930 in currency.

4.3 Data Used for Testing

Our simulation data is derived from a source associated with Burger King in Stavanger. The information was provided by an employee who expressed concerns about potential work agreement conflicts, so he offered us estimated data based on his extensive experience in fast-food scenarios. In order to maintain data privacy and confidentiality, we used his expertise to create a representation of average fast-food situations in our simulation.

Based on that data, we created different scenarios to test our module. By changing the global variable, any kind of scenarios can be created easily and can be compared with other scenarios to take vital decisions.

Table 1: Customer Generation and Resource Management

Case	EntCustGenTime	nCounter1Staff	nCounter2Staff	nBurgStaff	nDrinkStaff	nExtraStaff
1	3	1	1	2	2	2
2	2	1	1	1	1	1
3	2	1	1	1	1	1

Table 2: Counter Queue cap, Burger Ingredients Stock & Order Placement Time

Case	CQueCap	CBurgIngReq	CBurgIngBigStock	COrderPlaceTime
1	5	50	50*170 = 8500	1
2	2	50	50*170 = 8500	5
3	5	50	50*70 = 3500	1

Case 01 is our sample run. Everything was running smoothly and there was no by pass as everything was optimized for optimal simulation.

Case 02 In this scenario we are creating more customers by decreasing generation time to 02 minutes and also reduced staff number. We also decreased Counter Queue Capacity and increased the order placement time. This will cause by pass to hit as we are not able to serve too many customers efficiently with the given conditions.

A graph of total number of customers along with order placement along with by pass and receive amount has been showed in figure 33 and 34 for this specific scenario.

We can see that number of customers entered and exited the module is 125 but only 36 customers were able to place and receive order from counter 01 and 37 customers from counter 02. Rest of the 52 people were moved through bypass and was not able to place order.

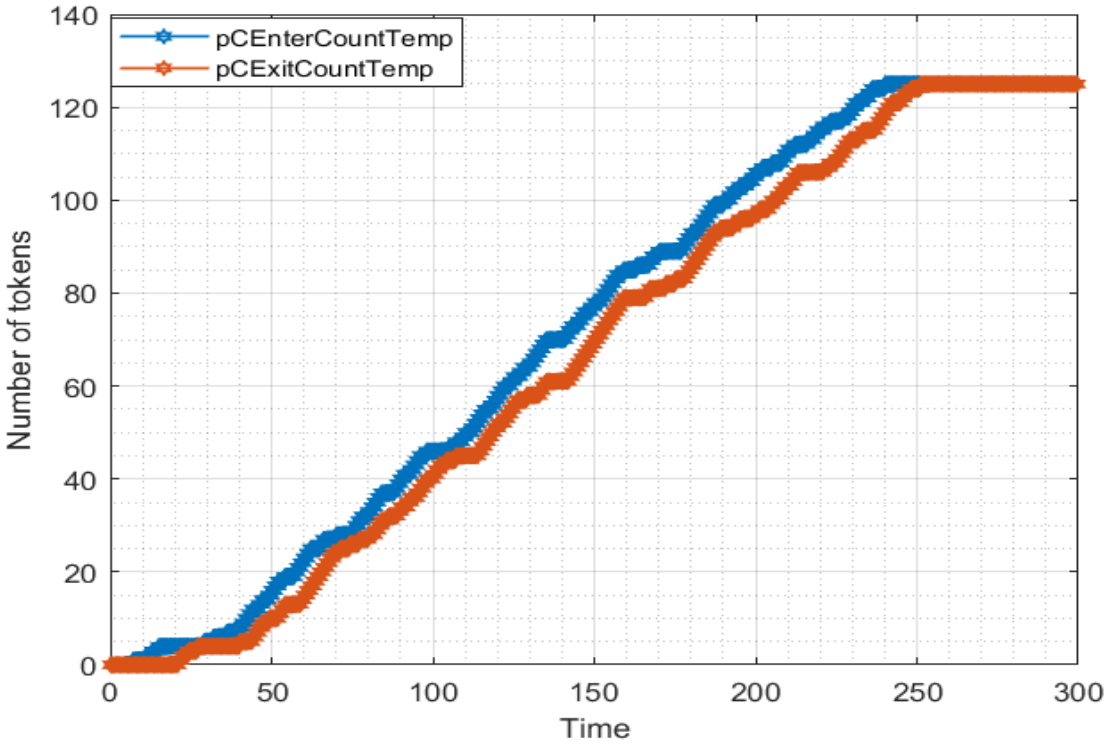

Figure 33: Total Number of Customers in Counter (CASE 02)

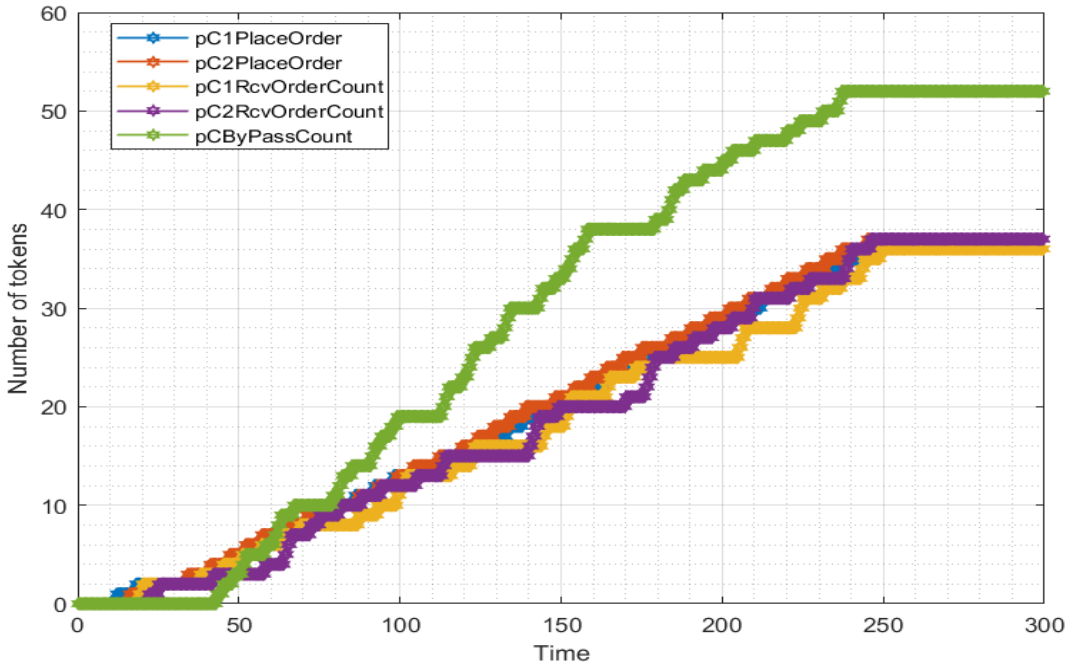

Figure 34: Total Number of Order Placed & Received along with ByPass (CASE 02)

5. Discussion

5.1 Originality of this work

As evidenced in the Testing, Analysis, and Results section, the primary advantage of our model lies in its adaptability, enabling the representation of diverse scenarios through the manipulation of a few key parameters. With each parameter being adjustable, the potential scenarios become practically limitless. Our model responds predictably to changes in these parameters, providing intricate and valuable insights into the dynamics of a fastfood operations. These insights have the potential to uncover unforeseen interdependencies between parameters and outcomes, which can then be leveraged to devise optimal business strategies for accommodating new circumstances.

5.2 Related Works

While existing studies developed by Gpensim do not specifically focus on fast-food workflow optimization, a relevant study on workforce optimization in fast-food is identified. This study employs a mixed-method approach, incorporating surveys, interviews, and CCTV data collection. Monte Carlo simulation is utilized to gain insights into crew performance under varying customer arrivals. Furthermore, integer programming optimizes labor costs by considering hourly customer influx, wait times, and workstation limitations. [4]

Additionally, insights from a prior project in the same course, focusing on bar logistics and resource management, contribute to understanding service-related aspects.

5.3 Limitation

While our project has proven satisfactory, there are notable limitations stemming from a lack of advanced knowledge in GPenSim system capabilities. Specifically, our current model lacks the functionality to enable customers to make menu choices based on their preferences. In the existing framework, customers are generated at fixed intervals, which deviates from the randomness that would more realistically reflect dynamic situations.

This limitation is particularly evident in scenarios where customer preferences play a crucial role, such as in fast-food establishments where menu choices are diverse. The inability to incorporate customer choice dynamics may impact the accuracy of

our simulation in replicating real-world scenarios, where customers' decisions are often influenced by various factors.

Recognizing this limitation is essential, as it acknowledges a constraint in the model's ability to fully capture the intricacies of customer behavior and adapt to the dynamic nature of fast-food service environments

5.4 Future Work

While our current project has provided valuable insights into resource optimization within a fast-food restaurant using the GPenSim system, there are exciting avenues for future enhancements and expansions. The following areas present opportunities for further development:

- Introducing a color receipt system could be explored as a means for customers to access different facilities within the restaurant. This enhancement could simulate a more dynamic and interactive customer experience, allowing for varied service paths based on the color-coded receipts.
- Future iterations could include the implementation of a random timing generation system, simulating more dynamic customer arrival patterns and providing a more accurate representation of the uncertainties in a fast-food environment.
- Exploring the integration of resource utilization across different modules presents an exciting avenue. For instance, optimizing resource sharing or allocation between the kitchen and dining area could lead to a more comprehensive understanding of operational efficiency and potentially uncover further areas for improvement.

These proposed future works aim to address current limitations and elevate the simulation model's capabilities to better emulate the complexities of a dynamic fast-food restaurant environment. By implementing these enhancements, the project can achieve a higher level of accuracy and provide even more valuable insights for the optimization of resources and customer service in the fast-food industry.

5.5 Learning Experience

While certain constraints within GPenSIM, like the absence of built-in support for variable timing and the capacity to remove designated colors or release specific resources, posed challenges during the development of our model, the overall learning experience proved to be rewarding. The satisfaction derived from witnessing our model perform as anticipated outweighed these minor inconveniences.

References

1. R. Davidrajuh, 'Modelling Discrete-Event Systems with GPenSIM: An Introduction', Stavanger: Springer, 2017

2. Zouhar, & Havlová. (n.d.). Are fast food chains really that efficient? A case study on crew optimization. Retrieved November 10, 2023, from http://mme2012.opf.slu.cz/proceedings/pdf/177_Zouhar.pdf

3. Kanyan, A., Ngana, L., & Voon, B. H. (2016, June 1). Improving the Service Operations of Fast-food Restaurants. Procedia - Social and Behavioral Sciences; Elsevier BV. https://doi.org/10.1016/j.sbspro.2016.05.439

4. Liggayu, David, Lubguban, & Perez. (n.d.). A Mixed-Method Approach Workforce Optimization for Fast-Food Restaurants. Retrieved November 10, 2023, from http://www.e-jikei.org/Conf/ICTSI2018/proceedings/materials/proc_files/GS_papers/GS_A012_/CameraReady_ICTSI2018_GS-A012.pdf

Appendix-A

A1: Complete Code

Only the main simulation file (MSF) will change based on case 01 and case 02. Rest of the files remain same for all the scenarios.

Case-01: ff.m (Main Simulation File)

```
clear all; clc;
global global_info
%--------------------------------------CASE ID: 01

%Simulation Run Time
global_info.START_AT = 0;
global_info.STOP_AT = 300;

%Closing time of Restaurant, after which no new customer will be generated
%and no one will be allowed to enter any other module; but customers who
%are already inside different modules will finish their functionality as
%usual
global_info.ClosingTime = 240;
global_info.TotalRunTime = global_info.STOP_AT - global_info.START_AT;

%-------------------------Initial IMC Module Conditions-------------------------
% The probability of which module should be used from IMC
% We are using randi([1,global_info.VARIABLE]) to determine probability
% So, the higher the number, the lesser the probability
global_info.CounterEnterProbability = 1;
global_info.TableEnterProbability = 2;
global_info.ToiletEnterProbability = 3;
global_info.KidsZoneEnterProbability = 5;
global_info.ChargDockEnterProbability = 5;
global_info.ExitProbability = 3;


%-------------------------Initial Entrance Module Conditions-----------------------
--
global_info.EntCustGenTime = 3;             %One Customer is being generated in every
# Minutes


%-------------------------Initial Counter Module Conditions-----------------------
-
%Customer
global_info.CNewCounterThreshold = 5;       %Threshold for when one more counter
should be used
global_info.CQueCap = 5;                    %Maximum Capacity of Counter Queue
global_info.CustRedirectBOBurgStock = 250;  %Burger Main stock's critical value based
on which customer will be redirected to bypass


%Burger
```

```
global_info.CBurgIngReq = 50;              %Amount of ingredients required for a
Burger
global_info.CBurgIngSmlStock = global_info.CBurgIngReq * 10;       %Bruger
Ingredients Kithcen Stock
global_info.CBurgIngBigStock = global_info.CBurgIngReq * 170;      %Burger
Ingredients Main Stock
global_info.CBurgRefillAmnt = global_info.CBurgIngReq * 7;        %Burger Ingredients
refill amount from Main to Kitchen Stock
global_info.CBurgRefillTriggerMark = global_info.CBurgIngReq * 3;  %Trigger Point
When Burger Ingredients in Kitchen should be refilled from Main Storage

%Drink
global_info.CDrinkIngReq = 50;             %Amount of ingredients required for a
Drink
global_info.CDrinkIngSmlStock = global_info.CDrinkIngReq * 10;      %Drink
Ingredients Kithcen Stock
global_info.CDrinkIngBigStock = global_info.CDrinkIngReq * 170;     %Drink
Ingredients Main Stock
global_info.CDrinkRefillAmnt = global_info.CDrinkIngReq * 7;       %Drink Ingredients
refill amount from Main to Kitchen Stock
global_info.CDrinkRefillTriggerMark = global_info.CDrinkIngReq * 3;  %Trigger Point
When Drink Ingredients in Kitchen should be refilled from Main Storage

%TakeAway Paper Bag (in pcs)
global_info.CTABagSmlStock = 40;           %TakeAway Bag Kithcen Stock
global_info.CTABagBigStock = 200;          %TakeAway Bag Main Stock
global_info.CTABagRefillAmnt = 35;         %TakeAway Bag refill amount from Main to
Kitchen Stock
global_info.CTABagRefillTriggerMark = 5;    %Trigger Point When TakeAway Bags in
Kitchen should be refilled from Main Storage

%DineIn Tray (in pcs)
global_info.CDITraySmlStock = 25;          %DineIn Tray Kithcen Stock
global_info.CDITrayBigStock = 50;          %DineIn Tray Main Stock
global_info.CDITrayRefillAmnt = 10;        %DineIn Tray refill amount from Main to
Kitchen Stock
global_info.CDITrayRefillTriggerMark = 5;   %Trigger Point When DineIn Trays in
Kitchen should be refilled from Main Storage
global_info.CDITrayCollect = 5;            %Trigger point when DineIn Trays should
be collected from Used Tray
global_info.CTrayColTime = 1;              %Tray collection time
global_info.CDITrayClean = 10;             %Trigger point when Dirty Trays should be
sent for cleaning
global_info.CTrayCleanTime = 2;            %Tray Clean Time

%Refill Time
global_info.CRefillTime = 2;               %Stock Refill Time

%Number of Counters based on Customer Arrival
if global_info.EntCustGenTime > global_info.CNewCounterThreshold
    global_info.OrderCounter = 1;
else
    global_info.OrderCounter = 2;
end
```

```
%Time Required for different Works
if global_info.OrderCounter == 1
    global_info.COrderPlaceTime = 2;        %Order Placement Time
    global_info.CBurgMakeTime = 4;          %Burger Making Time
    global_info.CDrinkMakeTime = 2;         %Drink Making Time
    global_info.CRcvOrderTime = 2;          %Order Delivery Time
else
    global_info.COrderPlaceTime = 1;        %Order Placement Time
    global_info.CBurgMakeTime = 2;          %Burger Making Time
    global_info.CDrinkMakeTime = 1;         %Drink Making Time
    global_info.CRcvOrderTime = 1;          %Order Delivery Time
end

%Resource Management
if global_info.OrderCounter == 1
    nCounter1Staff = 1;
    nBurgStaff = 1;
    nDrinkStaff = 1;
    nExtraStaff = 1;
else
    nCounter1Staff = 1;
    nCounter2Staff = 1;
    nBurgStaff = 2;
    nDrinkStaff = 2;
    nExtraStaff = 2;
end

%-------------------------Initial Table Module Conditions-------------------------
global_info.TabQueCap = 3;                  %Maximum Capacity of Table Queue
global_info.TabAvaialble = 2;               %Total number of Tables
global_info.TabUsageTime = 5;               %Table usage time for a customer

%-------------------------Initial Toilet Module Conditions-------------------------
global_info.TQueCap = 3;                    %Maximum Capacity of Table Queue
global_info.TAvaialble = 2;                 %Total number of Toilets
global_info.TUsageTime = 5;                 %Toitlet usage time for a customer

%-------------------------Initial KidsZone Module Conditions----------------------
--
global_info.KZQueCap = 3;                   %Maximum Capacity of KidsZone Queue
global_info.KZAvaialble = 2;                %Total number of Empty Slots
global_info.KZUsageTime = 5;                %KidsZone usage time for a customer

%-------------------------Initial ChargingDock Module Conditions------------------
------
global_info.CDQueCap = 5;                  %Maximum Capacity of ChargingDock Queue
global_info.CDAvaialble = 10;              %Total number of Charging Dock
global_info.CDUsageTime = 5;               %Charging Dock usage time for a customer


%----------------Cost Variables----------------
% Ingredients Cost
Costs.Burger = 35;
Costs.Drink = 5;
```

```
%Employee Cost (Per Minute)
if global_info.OrderCounter == 1
    Costs.EmpCounter1Staff = 5;
else
    Costs.EmpCounter1Staff = 5;
    Costs.EmpCounter2Staff = 5;
end
Costs.EmpBurgStaff = 5;
Costs.EmpDrinkStaff = 5;
Costs.EmpExtraStaff = 5;

%Fixed Cost (Per Minute)
Costs.FixedCosts = 30;

%----------Income Variables------------------
BurgerPrice = 250;
DrinkPrice = 80;

pns =
pnstruct({'Entrance_pdf','IMC_pdf','Counter_pdf','Table_pdf','Toilet_pdf','Kidszone_p
df','Chargingdock_pdf'});

dyn.m0 = {'pCBurgIng', global_info.CBurgIngSmlStock, 'pCStockBurgIng',
global_info.CBurgIngBigStock,...     %Burger Ingredients
    'pCDrinkIng', global_info.CDrinkIngSmlStock, 'pCStockDrinkIng',
global_info.CDrinkIngBigStock,...       %Drinks Ingredients
    'pCTABag', global_info.CTABagSmlStock, 'pCTABagStock',
global_info.CTABagBigStock,...                  %TakeAway Bags
    'pCDITray', global_info.CDITraySmlStock, 'pCDITrayStock',
global_info.CDITrayBigStock,...                %DineIn Tray
    };

if global_info.OrderCounter == 1
    dyn.ft = {'tEntEnter',
global_info.EntCustGenTime,'tC1PlaceOrder',global_info.COrderPlaceTime,...

'tCMakeBurg',global_info.CBurgMakeTime,'tCMakeDrink',global_info.CDrinkMakeTime,...

'tCBurgStockRefill',global_info.CRefillTime,'tCDrinkStockRefill',global_info.CRefillT
ime,...

'tCBagStockRefill',global_info.CRefillTime,'tCTrayStockRefill',global_info.CRefillTim
e,...
        'tCCollectTray',global_info.CTrayColTime,'tCCleanTray',
global_info.CTrayCleanTime,...
        'tC1RcvOrder',global_info.CRcvOrderTime,...
        'tTabLeave',global_info.TabUsageTime,'tTIn',global_info.TUsageTime,...

'tKZChooseGame',global_info.KZUsageTime,'tCDFindDock',global_info.CDUsageTime,...
        'allothers', 1};
else
    dyn.ft = {'tEntEnter',
global_info.EntCustGenTime,'tC1PlaceOrder',global_info.COrderPlaceTime,'tC2PlaceOrder
',global_info.COrderPlaceTime,...
```

```
'tCMakeBurg',global_info.CBurgMakeTime,'tCMakeDrink',global_info.CDrinkMakeTime,...

'tCBurgStockRefill',global_info.CRefillTime,'tCDrinkStockRefill',global_info.CRefillT
ime,...

'tCBagStockRefill',global_info.CRefillTime,'tCTrayStockRefill',global_info.CRefillTim
e,...
        'tCCollectTray',global_info.CTrayColTime,'tCCleanTray',
global_info.CTrayCleanTime,...

'tC1RcvOrder',global_info.CRcvOrderTime,'tC2RcvOrder',global_info.CRcvOrderTime,...
        'tTabLeave',global_info.TabUsageTime,'tTIn',global_info.TUsageTime,...

'tKZChooseGame',global_info.KZUsageTime,'tCDFindDock',global_info.CDUsageTime,...
        'allothers', 1};
end

if global_info.OrderCounter == 1
    dyn.re = {'Counter1Staff',nCounter1Staff,inf,...
        'BurgStaff',nBurgStaff,inf,...
        'DrinkStaff',nDrinkStaff,inf,...
        'ExtraStaff',nExtraStaff,inf};
else
    dyn.re =
{'Counter1Staff',nCounter1Staff,inf,'Counter2Staff',nCounter2Staff,inf,...
        'BurgStaff',nBurgStaff,inf,...
        'DrinkStaff',nDrinkStaff,inf,...
        'ExtraStaff',nExtraStaff,inf};
end

pni = initialdynamics(pns, dyn);
results = gpensim(pni);
prnschedule(results)
%prnss(results);

% -------------------Count Firing Time of Different Transition------------
nBurgCount = timesfired('tCMakeBurg')
nDrinkCount = timesfired('tCMakeDrink')

%----------------------Cost Calculations----------------------
TotalBurgCost = nBurgCount*Costs.Burger
TotalDrinkCost = nDrinkCount*Costs.Drink
if global_info.OrderCounter == 1
    TotalEmpCost = global_info.TotalRunTime*(nCounter1Staff*Costs.EmpCounter1Staff +
nBurgStaff*Costs.EmpBurgStaff + nDrinkStaff*Costs.EmpDrinkStaff +
nExtraStaff*Costs.EmpExtraStaff)
else
    TotalEmpCost = global_info.TotalRunTime*(nCounter1Staff*Costs.EmpCounter1Staff +
nCounter2Staff*Costs.EmpCounter2Staff + nBurgStaff*Costs.EmpBurgStaff +
nDrinkStaff*Costs.EmpDrinkStaff + nExtraStaff*Costs.EmpExtraStaff)
end
TotalFixCost = global_info.TotalRunTime*Costs.FixedCosts
TotalCost = TotalBurgCost + TotalDrinkCost + TotalEmpCost + TotalFixCost
```

```
%-----------------Income Calculations-------------------
Income.Burger = nBurgCount*BurgerPrice
Income.Drinks = nDrinkCount*DrinkPrice
TotalIncome = Income.Burger + Income.Drinks

%-------------------Profit Calculations------------------
Profit = TotalIncome - TotalCost

%---------------------------Plotting Graphs-------------------------

%%% Total Number of customers in different modules
figure(1), plotp(results,{'pEntCountTemp','pIMCExitCountTemp'});        %Total
Customer Generate, Total Customer Exit
figure(2), plotp(results,{'pCEnterCountTemp','pCExitCountTemp'});       %Total Number
of Customer In and Out from Counter
figure(3), plotp(results,{'pTabEnterCountTemp','pTabExitCountTemp'});   %Total Number
of Customer In and Out from Table
figure(4), plotp(results,{'pTEnterCountTemp','pTExitCountTemp'});       %Total Number
of Customer In and Out from Toilet
figure(5), plotp(results,{'pKZEnterCountTemp','pKZExitCountTemp'});     %Total Number
of Customer In and Out from KidsZone
figure(6), plotp(results,{'pCDEnterCountTemp','pCDExitCountTemp'});     %Total Number
of Customer In and Out from ChargingDock

%%%------------------Visulisation of Counter Module
%Current number of customers in Queue and waiting for orders
if global_info.OrderCounter == 1
    figure(7), plotp(results,{'pCQue','pC1WaitingOrder'});
else
    figure(7), plotp(results,{'pCQue','pC1WaitingOrder','pC2WaitingOrder'});
end
%Total Number of Order placed, received and bypassed customer
if global_info.OrderCounter == 1
    figure(8), plotp(results,{'pC1PlaceOrder','pC1RcvOrderCount','pCByPassCount'});
else
    figure(8),
plotp(results,{'pC1PlaceOrder','pC2PlaceOrder','pC1RcvOrderCount','pC2RcvOrderCount',
'pCByPassCount'});
end
figure(9), plotp(results,{'pCStockBurgIng','pCBurgIng'});               %Burger
Ingredients Stock
figure(10), plotp(results,{'pCStockDrinkIng','pCDrinkIng'});            %Drinks
Ingredients Stock
figure(11), plotp(results,{'pCTakeAwayCount','pCDineInCount'});         %Total Number
of TakeAway and DineIn orders
figure(12), plotp(results,{'pCTABagStock','pCTABag'});                  %Take Away
Paper Bag Stock
figure(13), plotp(results,{'pCDITrayStock','pCDITray'});                %Dine In Tray
Stock

%%%------------------Visulisation of current stats of other Modules
figure(14), plotp(results,{'pTabQue','pTabInUse'});
figure(15), plotp(results,{'pTQue','pTInUse','pTSinkQue','pTSinkUse'});
figure(16), plotp(results,{'pKZQue','pKZPlayZone'});
figure(17), plotp(results,{'pCDQue','pCDInUse'});
```

```
plotGC(results)
```

Case-02: ff.m (Main Simulation File)

```
clear all; clc;
global global_info
%---------------------------------------CASE ID: 02

%Simulation Run Time
global_info.START_AT = 0;
global_info.STOP_AT = 300;

%Closing time of Restaurant, after which no new customer will be generated
%and no one will be allowed to enter any other module; but customers who
%are already inside different modules will finish their functionality as
%usual
global_info.ClosingTime = 240;
global_info.TotalRunTime = global_info.STOP_AT - global_info.START_AT;

%-------------------------Initial IMC Module Conditions-------------------------
% The probability of which module should be used from IMC
% We are using randi([1,global_info.VARIABLE]) to determine probability
% So, the higher the number, the lesser the probability
global_info.CounterEnterProbability = 1;
global_info.TableEnterProbability = 2;
global_info.ToiletEnterProbability = 3;
global_info.KidsZoneEnterProbability = 5;
global_info.ChargDockEnterProbability = 5;
global_info.ExitProbability = 3;


%-------------------------Initial Entrance Module Conditions-----------------------
--
global_info.EntCustGenTime = 2;             %One Customer is being generated in every
# Minutes


%-------------------------Initial Counter Module Conditions-----------------------
-
%Customer
global_info.CNewCounterThreshold = 5;       %Threshold for when one more counter
should be used
global_info.CQueCap = 2;                    %Maximum Capacity of Counter Queue
global_info.CustRedirectBOBurgStock = 250;  %Burger Main stock's critical value based
on which customer will be redirected to bypass


%Burger
global_info.CBurgIngReq = 50;              %Amount of ingredients required for a
Burger
global_info.CBurgIngSmlStock = global_info.CBurgIngReq * 10;       %Bruger
Ingredients Kithcen Stock
global_info.CBurgIngBigStock = global_info.CBurgIngReq * 170;      %Burger
Ingredients Main Stock
```

```
global_info.CBurgRefillAmnt = global_info.CBurgIngReq * 7;        %Burger Ingredients
refill amount from Main to Kitchen Stock
global_info.CBurgRefillTriggerMark = global_info.CBurgIngReq * 3;  %Trigger Point
When Burger Ingredients in Kitchen should be refilled from Main Storage

%Drink
global_info.CDrinkIngReq = 50;             %Amount of ingredients required for a
Drink
global_info.CDrinkIngSmlStock = global_info.CDrinkIngReq * 10;      %Drink
Ingredients Kithcen Stock
global_info.CDrinkIngBigStock = global_info.CDrinkIngReq * 170;     %Drink
Ingredients Main Stock
global_info.CDrinkRefillAmnt = global_info.CDrinkIngReq * 7;       %Drink Ingredients
refill amount from Main to Kitchen Stock
global_info.CDrinkRefillTriggerMark = global_info.CDrinkIngReq * 3;  %Trigger Point
When Drink Ingredients in Kitchen should be refilled from Main Storage

%TakeAway Paper Bag (in pcs)
global_info.CTABagSmlStock = 40;            %TakeAway Bag Kithcen Stock
global_info.CTABagBigStock = 200;           %TakeAway Bag Main Stock
global_info.CTABagRefillAmnt = 35;          %TakeAway Bag refill amount from Main to
Kitchen Stock
global_info.CTABagRefillTriggerMark = 5;    %Trigger Point When TakeAway Bags in
Kitchen should be refilled from Main Storage

%DineIn Tray (in pcs)
global_info.CDITraySmlStock = 25;           %DineIn Tray Kithcen Stock
global_info.CDITrayBigStock = 50;          %DineIn Tray Main Stock
global_info.CDITrayRefillAmnt = 10;         %DineIn Tray refill amount from Main to
Kitchen Stock
global_info.CDITrayRefillTriggerMark = 5;   %Trigger Point When DineIn Trays in
Kitchen should be refilled from Main Storage
global_info.CDITrayCollect = 5;             %Trigger point when DineIn Trays should
be collected from Used Tray
global_info.CTrayColTime = 1;               %Tray collection time
global_info.CDITrayClean = 10;              %Trigger point when Dirty Trays should be
sent for cleaning
global_info.CTrayCleanTime = 2;             %Tray Clean Time

%Refill Time
global_info.CRefillTime = 2;                %Stock Refill Time

%Number of Counters based on Customer Arrival
if global_info.EntCustGenTime > global_info.CNewCounterThreshold
    global_info.OrderCounter = 1;
else
    global_info.OrderCounter = 2;
end

%Time Required for different Works
if global_info.OrderCounter == 1
    global_info.COrderPlaceTime = 5;        %Order Placement Time
    global_info.CBurgMakeTime = 4;          %Burger Making Time
    global_info.CDrinkMakeTime = 2;         %Drink Making Time
    global_info.CRcvOrderTime = 2;          %Order Delivery Time
```

```
else
    global_info.COrderPlaceTime = 5;        %Order Placement Time
    global_info.CBurgMakeTime = 2;          %Burger Making Time
    global_info.CDrinkMakeTime = 1;         %Drink Making Time
    global_info.CRcvOrderTime = 1;          %Order Delivery Time
end

%Resource Management
if global_info.OrderCounter == 1
    nCounter1Staff = 1;
    nBurgStaff = 1;
    nDrinkStaff = 1;
    nExtraStaff = 1;
else
    nCounter1Staff = 1;
    nCounter2Staff = 1;
    nBurgStaff = 1;
    nDrinkStaff = 1;
    nExtraStaff = 1;
end

%-------------------------Initial Table Module Conditions-------------------------
global_info.TabQueCap = 3;                  %Maximum Capacity of Table Queue
global_info.TabAvaialble = 2;               %Total number of Tables
global_info.TabUsageTime = 5;               %Table usage time for a customer

%-------------------------Initial Toilet Module Conditions-------------------------
global_info.TQueCap = 3;                    %Maximum Capacity of Table Queue
global_info.TAvaialble = 2;                 %Total number of Toilets
global_info.TUsageTime = 5;                 %Toitlet usage time for a customer

%-------------------------Initial KidsZone Module Conditions-----------------------
--
global_info.KZQueCap = 3;                   %Maximum Capacity of KidsZone Queue
global_info.KZAvaialble = 2;                %Total number of Empty Slots
global_info.KZUsageTime = 5;                %KidsZone usage time for a customer

%-------------------------Initial ChargingDock Module Conditions-------------------
------
global_info.CDQueCap = 5;                  %Maximum Capacity of ChargingDock Queue
global_info.CDAvaialble = 10;              %Total number of Charging Dock
global_info.CDUsageTime = 5;               %Charging Dock usage time for a customer


%----------------Cost Variables---------------
% Ingredients Cost
Costs.Burger = 35;
Costs.Drink = 5;

%Employee Cost (Per Minute)
if global_info.OrderCounter == 1
    Costs.EmpCounter1Staff = 5;
else
    Costs.EmpCounter1Staff = 5;
    Costs.EmpCounter2Staff = 5;
```

```
end
Costs.EmpBurgStaff = 5;
Costs.EmpDrinkStaff = 5;
Costs.EmpExtraStaff = 5;

%Fixed Cost (Per Minute)
Costs.FixedCosts = 30;

%----------Income Variables-------------------
BurgerPrice = 250;
DrinkPrice = 80;

pns =
pnstruct({'Entrance_pdf','IMC_pdf','Counter_pdf','Table_pdf','Toilet_pdf','Kidszone_p
df','Chargingdock_pdf'});

dyn.m0 = {'pCBurgIng', global_info.CBurgIngSmlStock, 'pCStockBurgIng',
global_info.CBurgIngBigStock,...     %Burger Ingredients
    'pCDrinkIng', global_info.CDrinkIngSmlStock, 'pCStockDrinkIng',
global_info.CDrinkIngBigStock,...       %Drinks Ingredients
    'pCTABag', global_info.CTABagSmlStock, 'pCTABagStock',
global_info.CTABagBigStock,...                    %TakeAway Bags
    'pCDITray', global_info.CDITraySmlStock, 'pCDITrayStock',
global_info.CDITrayBigStock,...                %DineIn Tray
    };

if global_info.OrderCounter == 1
    dyn.ft = {'tEntEnter',
global_info.EntCustGenTime,'tC1PlaceOrder',global_info.COrderPlaceTime,...

'tCMakeBurg',global_info.CBurgMakeTime,'tCMakeDrink',global_info.CDrinkMakeTime,...

'tCBurgStockRefill',global_info.CRefillTime,'tCDrinkStockRefill',global_info.CRefillT
ime,...

'tCBagStockRefill',global_info.CRefillTime,'tCTrayStockRefill',global_info.CRefillTim
e,...
        'tCCollectTray',global_info.CTrayColTime,'tCCleanTray',
global_info.CTrayCleanTime,...
        'tC1RcvOrder',global_info.CRcvOrderTime,...
        'tTabLeave',global_info.TabUsageTime,'tTIn',global_info.TUsageTime,...

'tKZChooseGame',global_info.KZUsageTime,'tCDFindDock',global_info.CDUsageTime,...
        'allothers', 1};
else
    dyn.ft = {'tEntEnter',
global_info.EntCustGenTime,'tC1PlaceOrder',global_info.COrderPlaceTime,'tC2PlaceOrder
',global_info.COrderPlaceTime,...

'tCMakeBurg',global_info.CBurgMakeTime,'tCMakeDrink',global_info.CDrinkMakeTime,...

'tCBurgStockRefill',global_info.CRefillTime,'tCDrinkStockRefill',global_info.CRefillT
ime,...
```

```
'tCBagStockRefill',global_info.CRefillTime,'tCTrayStockRefill',global_info.CRefillTim
e,...
        'tCCollectTray',global_info.CTrayColTime,'tCCleanTray',
global_info.CTrayCleanTime,...

'tC1RcvOrder',global_info.CRcvOrderTime,'tC2RcvOrder',global_info.CRcvOrderTime,...
        'tTabLeave',global_info.TabUsageTime,'tTIn',global_info.TUsageTime,...

'tKZChooseGame',global_info.KZUsageTime,'tCDFindDock',global_info.CDUsageTime,...
        'allothers', 1};
end

if global_info.OrderCounter == 1
    dyn.re = {'Counter1Staff',nCounter1Staff,inf,...
        'BurgStaff',nBurgStaff,inf,...
        'DrinkStaff',nDrinkStaff,inf,...
        'ExtraStaff',nExtraStaff,inf};
else
    dyn.re =
{'Counter1Staff',nCounter1Staff,inf,'Counter2Staff',nCounter2Staff,inf,...
        'BurgStaff',nBurgStaff,inf,...
        'DrinkStaff',nDrinkStaff,inf,...
        'ExtraStaff',nExtraStaff,inf};
end

pni = initialdynamics(pns, dyn);
results = gpensim(pni);
prnschedule(results)
%prnss(results);

% -------------------Count Firing Time of Different Transition------------
nBurgCount = timesfired('tCMakeBurg')
nDrinkCount = timesfired('tCMakeDrink')

%----------------------Cost Calculations----------------------
TotalBurgCost = nBurgCount*Costs.Burger
TotalDrinkCost = nDrinkCount*Costs.Drink
if global_info.OrderCounter == 1
    TotalEmpCost = global_info.TotalRunTime*(nCounter1Staff*Costs.EmpCounter1Staff +
nBurgStaff*Costs.EmpBurgStaff + nDrinkStaff*Costs.EmpDrinkStaff +
nExtraStaff*Costs.EmpExtraStaff)
else
    TotalEmpCost = global_info.TotalRunTime*(nCounter1Staff*Costs.EmpCounter1Staff +
nCounter2Staff*Costs.EmpCounter2Staff + nBurgStaff*Costs.EmpBurgStaff +
nDrinkStaff*Costs.EmpDrinkStaff + nExtraStaff*Costs.EmpExtraStaff)
end
TotalFixCost = global_info.TotalRunTime*Costs.FixedCosts
TotalCost = TotalBurgCost + TotalDrinkCost + TotalEmpCost + TotalFixCost

%-----------------Income Calculations-------------------
Income.Burger = nBurgCount*BurgerPrice
Income.Drinks = nDrinkCount*DrinkPrice
TotalIncome = Income.Burger + Income.Drinks
```

```
%-------------------Profit Calculations-----------------
Profit = TotalIncome - TotalCost

%---------------------------Plotting Graphs-------------------------

%%% Total Number of customers in different modules
figure(1), plotp(results,{'pEntCountTemp','pIMCExitCountTemp'});        %Total
Customer Generate, Total Customer Exit
figure(2), plotp(results,{'pCEnterCountTemp','pCExitCountTemp'});       %Total Number
of Customer In and Out from Counter
figure(3), plotp(results,{'pTabEnterCountTemp','pTabExitCountTemp'});   %Total Number
of Customer In and Out from Table
figure(4), plotp(results,{'pTEnterCountTemp','pTExitCountTemp'});       %Total Number
of Customer In and Out from Toilet
figure(5), plotp(results,{'pKZEnterCountTemp','pKZExitCountTemp'});     %Total Number
of Customer In and Out from KidsZone
figure(6), plotp(results,{'pCDEnterCountTemp','pCDExitCountTemp'});     %Total Number
of Customer In and Out from ChargingDock

%%------------------Visulisation of Counter Module
%Current number of customers in Queue and waiting for orders
if global_info.OrderCounter == 1
    figure(7), plotp(results,{'pCQue','pC1WaitingOrder'});
else
    figure(7), plotp(results,{'pCQue','pC1WaitingOrder','pC2WaitingOrder'});
end
Total Number of Order placed, received and bypassed customer
if global_info.OrderCounter == 1
    figure(8), plotp(results,{'pC1PlaceOrder','pC1RcvOrderCount','pCByPassCount'});
else
    figure(8),
plotp(results,{'pC1PlaceOrder','pC2PlaceOrder','pC1RcvOrderCount','pC2RcvOrderCount',
'pCByPassCount'});
end
figure(9), plotp(results,{'pCStockBurgIng','pCBurgIng'});               %Burger
Ingredients Stock
figure(10), plotp(results,{'pCStockDrinkIng','pCDrinkIng'});            %Drinks
Ingredients Stock
figure(11), plotp(results,{'pCTakeAwayCount','pCDineInCount'});         %Total Number
of TakeAway and DineIn orders
figure(12), plotp(results,{'pCTABagStock','pCTABag'});                  %Take Away
Paper Bag Stock
figure(13), plotp(results,{'pCDITrayStock','pCDITray'});                %Dine In Tray
Stock

%%%------------------Visulisation of current stats of other Modules
figure(14), plotp(results,{'pTabQue','pTabInUse'});
figure(15), plotp(results,{'pTQue','pTInUse','pTSinkQue','pTSinkUse'});
figure(16), plotp(results,{'pKZQue','pKZPlayZone'});
figure(17), plotp(results,{'pCDQue','pCDInUse'});

plotGC(results)
```

Entrance_pdf.m

```
function [png] = Entrance_pdf()
global global_info

png.PN_name='Entrance';

png.set_of_Ps={'pEntBuffer',...
    'pEntCountTemp',... %temp
    };
png.set_of_Ts={'tEntEnter','tEntExit'};
png.set_of_As={'tEntEnter','pEntBuffer',1,'pEntBuffer','tEntExit',1,...
    'tEntEnter','pEntCountTemp',1,... %temp
    };
png.set_of_Ports = {'tEntEnter','tEntExit'};
```

Counter_pdf.m

```
function [png] = Counter_pdf()
global global_info

png.PN_name = 'Counter';

if global_info.OrderCounter == 1
    png.set_of_Ps = {'pCEnterCountTemp','pCExitCountTemp',... %temp

'pC1PlaceOrder','pCBurgCount','pCDrinkCount','pC1RcvOrderCount','pCTakeAwayCount','pC
DineInCount','pCByPassCount',... %temp places

'pCQue','pCKitchen','pCBurgReady','pCDrinkReady','pC1WaitingOrder','pCRcvOrder','pCOu
tBuffer',... %order flow

'pCBurgIng','pCStockBurgIng','pCDrinkIng','pCStockDrinkIng','pCTABag','pCTABagStock',
'pCDITray','pCDITrayStock',... %storage
        'pCDIUsedTray','pCDICollectedTray',... %reusability
        };
    png.set_of_Ts =
{'tCEnter','tC1PlaceOrder','tCMakeBurg','tCMakeDrink','tC1RcvOrder','tCTakeAway','tCD
ineIn','tCExit',... %order flow

'tCBurgStockRefill','tCDrinkStockRefill','tCBagStockRefill','tCTrayStockRefill',...
%storage
        'tCCollectTray','tCCleanTray',... %reusability
        'tCByPass',... %ByPass
        };
    png.set_of_As = {'tCEnter','pCEnterCountTemp',1,... %temp
        'tCEnter','pCQue',1,'pCQue','tC1PlaceOrder',1,... %order flow
        'tC1PlaceOrder','pC1PlaceOrder',1,... %temp
        'tC1PlaceOrder','pCKitchen',1,'tC1PlaceOrder','pC1WaitingOrder',1,... %order
flow

'pCKitchen','tCMakeBurg',1,'pCBurgIng','tCMakeBurg',global_info.CBurgIngReq,'tCMakeBu
rg','pCBurgReady',1,... %Burger Making
```

```
'pCStockBurgIng','tCBurgStockRefill',global_info.CBurgRefillAmnt,'tCBurgStockRefill',
'pCBurgIng',global_info.CBurgRefillAmnt,... %Burger Ingredients Storage
        'tCMakeBurg','pCBurgCount',1,... %temp

'pCBurgReady','tCMakeDrink',1,'pCDrinkIng','tCMakeDrink',global_info.CDrinkIngReq,'tC
MakeDrink','pCDrinkReady',1,... %Drink Making

'pCStockDrinkIng','tCDrinkStockRefill',global_info.CDrinkRefillAmnt,'tCDrinkStockRefi
ll','pCDrinkIng',global_info.CDrinkRefillAmnt,... %Drink Ingredients Storage
        'tCMakeDrink','pCDrinkCount',1,... %temp

'pCDrinkReady','tC1RcvOrder',1,'pC1WaitingOrder','tC1RcvOrder',1,'tC1RcvOrder','pCRcv
Order',1,... %order flow
        'tC1RcvOrder','pC1RcvOrderCount',1,... %temp
        'pCRcvOrder','tCTakeAway',1,'pCRcvOrder','tCDineIn',1,... %takeaway or dinein

'pCTABagStock','tCBagStockRefill',global_info.CTABagRefillAmnt,'tCBagStockRefill','pC
TABag',global_info.CTABagRefillAmnt,... %TakeAway Bag Storage
        'tCTakeAway','pCTakeAwayCount',1,... %temp

'pCDITrayStock','tCTrayStockRefill',global_info.CDITrayRefillAmnt,'tCTrayStockRefill'
,'pCDITray',global_info.CDITrayRefillAmnt,... %DineIn Tray Storage
        'tCDineIn','pCDineInCount',1,... %temp

'tCTakeAway','pCOutBuffer',1,'tCDineIn','pCOutBuffer',1,'pCOutBuffer','tCExit',1,...
%order flow

'tCDineIn','pCDIUsedTray',1,'pCDIUsedTray','tCCollectTray',1,'tCCollectTray','pCDICol
lectedTray',1,... %DineIn Tray Flow
        'pCDICollectedTray','tCCleanTray',1,'tCCleanTray','pCDITray',1,... %DineIn
Tray Flow

'pCQue','tCByPass',1,'tCByPass','pCOutBuffer',1,'tCByPass','pCByPassCount',1,...
%ByPass
        'tCExit','pCExitCountTemp',1,... %temp
        };

elseif global_info.OrderCounter == 2
    png.set_of_Ps = {'pCEnterCountTemp','pCExitCountTemp',... %temp

'pC1PlaceOrder','pC2PlaceOrder','pCBurgCount','pCDrinkCount','pC1RcvOrderCount','pC2R
cvOrderCount','pCTakeAwayCount','pCDineInCount','pCByPassCount',... %temp places

'pCQue','pCKitchen','pCBurgReady','pCDrinkReady','pC1WaitingOrder','pC2WaitingOrder',
'pCRcvOrder','pCOutBuffer',... %order flow

'pCBurgIng','pCStockBurgIng','pCDrinkIng','pCStockDrinkIng','pCTABag','pCTABagStock',
'pCDITray','pCDITrayStock',... %storage
        'pCDIUsedTray','pCDICollectedTray',... %reusability
        };
    png.set_of_Ts =
{'tCEnter','tC1PlaceOrder','tC2PlaceOrder','tCMakeBurg','tCMakeDrink','tC1RcvOrder','
tC2RcvOrder','tCTakeAway','tCDineIn','tCExit',... %order flow
```

```
'tCBurgStockRefill','tCDrinkStockRefill','tCBagStockRefill','tCTrayStockRefill',...
%storage
        'tCCollectTray','tCCleanTray',... %reusability
        'tCByPass',... %ByPass
        };
    png.set_of_As = {'tCEnter','pCEnterCountTemp',1,... %temp
        'tCEnter','pCQue',1,'pCQue','tC1PlaceOrder',1,'pCQue','tC2PlaceOrder',1,...
%order flow
        'tC1PlaceOrder','pC1PlaceOrder',1,'tC2PlaceOrder','pC2PlaceOrder',1,... %temp
        'tC1PlaceOrder','pCKitchen',1,'tC1PlaceOrder','pC1WaitingOrder',1,... %order
flow
        'tC2PlaceOrder','pCKitchen',1,'tC2PlaceOrder','pC2WaitingOrder',1,... %order
flow

'pCKitchen','tCMakeBurg',1,'pCBurgIng','tCMakeBurg',global_info.CBurgIngReq,'tCMakeBu
rg','pCBurgReady',1,... %Burger Making

'pCStockBurgIng','tCBurgStockRefill',global_info.CBurgRefillAmnt,'tCBurgStockRefill',
'pCBurgIng',global_info.CBurgRefillAmnt,... %Burger Ingredients Storage
        'tCMakeBurg','pCBurgCount',1,... %temp

'pCBurgReady','tCMakeDrink',1,'pCDrinkIng','tCMakeDrink',global_info.CDrinkIngReq,'tC
MakeDrink','pCDrinkReady',1,... %Drink Making

'pCStockDrinkIng','tCDrinkStockRefill',global_info.CDrinkRefillAmnt,'tCDrinkStockRefi
ll','pCDrinkIng',global_info.CDrinkRefillAmnt,... %Drink Ingredients Storage
        'tCMakeDrink','pCDrinkCount',1,... %temp

'pCDrinkReady','tC1RcvOrder',1,'pC1WaitingOrder','tC1RcvOrder',1,'tC1RcvOrder','pCRcv
Order',1,... %order flow

'pCDrinkReady','tC2RcvOrder',1,'pC2WaitingOrder','tC2RcvOrder',1,'tC2RcvOrder','pCRcv
Order',1,... %order flow
        'tC1RcvOrder','pC1RcvOrderCount',1,'tC2RcvOrder','pC2RcvOrderCount',1,...
%temp

'pCRcvOrder','tCTakeAway',1,'pCTABag','tCTakeAway',1,'tCTakeAway','pCOutBuffer',1,...
%TakeAway

'pCTABagStock','tCBagStockRefill',global_info.CTABagRefillAmnt,'tCBagStockRefill','pC
TABag',global_info.CTABagRefillAmnt,... %TakeAway Bag Storage
        'tCTakeAway','pCTakeAwayCount',1,... %temp

'pCRcvOrder','tCDineIn',1,'pCDITray','tCDineIn',1,'tCDineIn','pCOutBuffer',1,...
%DneIn

'pCDITrayStock','tCTrayStockRefill',global_info.CDITrayRefillAmnt,'tCTrayStockRefill'
,'pCDITray',global_info.CDITrayRefillAmnt,... %DineIn Tray Storage
        'tCDineIn','pCDineInCount',1,... %temp
        'pCOutBuffer','tCExit',1,... %Order Flow

'tCDineIn','pCDIUsedTray',1,'pCDIUsedTray','tCCollectTray',global_info.CDITrayCollect
,'tCCollectTray','pCDICollectedTray',global_info.CDITrayCollect,... %DineIn Tray Flow
```

```
'pCDICollectedTray','tCCleanTray',global_info.CDITrayClean,'tCCleanTray','pCDITray',g
lobal_info.CDITrayClean,... %DineIn Tray Flow

'pCQue','tCByPass',1,'tCByPass','pCOutBuffer',1,'tCByPass','pCByPassCount',1,...
%ByPass
        'tCExit','pCExitCountTemp',1,... %temp
        };
end

png.set_of_Ports = {'tCEnter','tCExit'};
```

Toilet_pdf.m

```
function [png] = Toilet_pdf()
global global_info

png.PN_name='Toilet';

png.set_of_Ps ={'pTQue','pTInUse','pTSinkQue','pTSinkUse','pTOutBuffer',...
    'pTEnterCountTemp','pTExitCountTemp',... %temp
    };
png.set_of_Ts={'tTEnter','tTIn','tTleave','tTSinkIn','tTSinkOut','tTExit','tTByPass'}
;
png.set_of_As={'tTEnter','pTEnterCountTemp',1,... %temp
    'tTEnter','pTQue',1,'pTQue','tTIn',1,'tTIn','pTInUse',1,'pTInUse','tTleave',1,...

'tTleave','pTSinkQue',1,'pTSinkQue','tTSinkIn',1,'tTSinkIn','pTSinkUse',1,'pTSinkUse'
,'tTSinkOut',1,...
    'tTSinkOut','pTOutBuffer',1,'pTOutBuffer','tTExit',1,...
    'tTExit','pTExitCountTemp',1,... %temp
    'pTQue','tTByPass',1,'tTByPass','pTOutBuffer',1};
png.set_of_Ports = {'tTEnter','tTExit'};
```

Table_pdf.m

```
function [png] = Table_pdf()

global global_info

png.PN_name='Table';

png.set_of_Ps={'pTabQue','pTabInUse','pTabOutBuffer',...
    'pTabEnterCountTemp','pTabExitCountTemp'};
png.set_of_Ts={'tTabEnter','tTabFind','tTabLeave','tTabByPass','tTabExit'};
png.set_of_As={'tTabEnter','pTabEnterCountTemp',1,... %temp
    'tTabEnter','pTabQue',1,'pTabQue','tTabFind',1,'tTabFind','pTabInUse',1....

'pTabInUse','tTabLeave',1,'tTabLeave','pTabOutBuffer',1,'pTabOutBuffer','tTabExit',1,
...
    'tTabExit','pTabExitCountTemp',1,... %temp
    'pTabQue','tTabByPass',1,'tTabByPass','pTabOutBuffer',1};
```

```
png.set_of_Ports = {'tTabEnter','tTabExit'};
```

Kidszone_pdf.m

```
function [png] = Kidszone_pdf()
global global_info

png.PN_name='Kidszone';

png.set_of_Ps={'pKZQue','pKZPlayZone','pKZOutBuffer',...
    'pKZEnterCountTemp','pKZExitCountTemp'};
png.set_of_Ts={'tKZEnter','tKZChooseGame','tKZJumping','tKZSliding','tKZExit','tKZByP
ass'};
png.set_of_As= {'tKZEnter','pKZEnterCountTemp',1,... %temp

'tKZEnter','pKZQue',1,'pKZQue','tKZChooseGame',1,'tKZChooseGame','pKZPlayZone',1,...
    'pKZPlayZone','tKZJumping',1,'tKZJumping','pKZOutBuffer',1,...
    'pKZPlayZone','tKZSliding',1,'tKZSliding','pKZOutBuffer',1,...
    'pKZOutBuffer','tKZExit',1,...
    'tKZExit','pKZExitCountTemp',1,... %temp
    'pKZQue','tKZByPass',1,'tKZByPass','pKZOutBuffer',1,... %ByPass
    };
png.set_of_Ports = {'tKZEnter','tKZExit'};
```

Chargingdock_pdf.m

```
function [png] = Chargingdock_pdf()
global global_info

png.PN_name='Chargingdock';

png.set_of_Ps={'pCDQue','pCDInUse','pCDOutBuffer',...
    'pCDEnterCountTemp','pCDExitCountTemp',... %temp
    };
png.set_of_Ts={'tCDEnter','tCDFindDock','tCDComplete','tCDExit','tCDByPass'};
png.set_of_As= {'tCDEnter','pCDEnterCountTemp',1,...

'tCDEnter','pCDQue',1,'pCDQue','tCDFindDock',1,'tCDFindDock','pCDInUse',1,'pCDInUse',
'tCDComplete',1,...
    'tCDComplete','pCDOutBuffer',1,'pCDOutBuffer','tCDExit',1,...
    'tCDExit','pCDExitCountTemp',1,... %temp
    'pCDQue','tCDByPass',1,'tCDByPass','pCDOutBuffer',1};
png.set_of_Ports = {'tCDEnter','tCDExit'};
```

IMC_pdf.m

```
function [png] = IMC_pdf()

png.PN_name = 'IMC';
```

```
png.set_of_Ps = {'pIMCInBuffer', 'pIMCOutBuffer',...
    'pIMCExitCountTemp',... %temp
    };
png.set_of_Ts = {'tIMCCustDistribution', 'tIMCExit'};
png.set_of_As = {'tEntExit','pIMCInBuffer',1,...
    'pIMCInBuffer','tIMCCustDistribution',1,...
    'tIMCCustDistribution','pIMCOutBuffer',1,...
    'pIMCOutBuffer','tIMCExit',1,...
    'tIMCExit','pIMCExitCountTemp',1,... %temp
    'pIMCOutBuffer','tCEnter',1,'tCExit','pIMCInBuffer',1,... %Counter
    'pIMCOutBuffer','tTabEnter',1,'tTabExit','pIMCInBuffer',1,... %Table
    'pIMCOutBuffer','tTEnter',1,'tTExit','pIMCInBuffer',1,... %Toilet
    'pIMCOutBuffer','tKZEnter',1,'tKZExit','pIMCInBuffer',1,... %KidsZone
    'pIMCOutBuffer','tCDEnter',1,'tCDExit','pIMCInBuffer',1,... %ChargingDock
    };
```

MOD_Entrance_PRE.m

```
function [fire, transition] = MOD_Entrance_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In Entrance PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));
CurrentTime = current_time();
disp(transition.name)

if strcmp(transition.name, 'tEntEnter')
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        fire= 1;
    end
end
```

MOD_Counter_PRE.m

```
function [fire, transition] = MOD_Counter_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In Counter PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));


% Assignining "Service Type" Color in tCounterEnter transition
if strcmp(transition.name, 'tC1PlaceOrder')
    if ntokens('pCStockBurgIng') > global_info.CustRedirectBOBurgStock
```

```
        %randomly generate service type: TakeAway or DineIn?
        randomService = randi([1,2]);
        if randomService == 1
            ServiceChoice = 'TakeAway';
        else
            ServiceChoice = 'DineIn';
        end
        %Assign the randomly generated color to the token
        transition.new_color = {ServiceChoice};
        fire = requestSR({'Counter1Staff',1});
    else
        fire = 0;
    end

% Assignining "Service Type" Color in tCounterEnter transition
elseif strcmp(transition.name, 'tC2PlaceOrder')
    if ntokens('pCStockBurgIng') > global_info.CustRedirectBOBurgStock
        %randomly generate service type: TakeAway or DineIn?
        randomService = randi([1,2]);
        if randomService == 1
            ServiceChoice = 'TakeAway';
        else
            ServiceChoice = 'DineIn';
        end
        %Assign the randomly generated color to the token
        transition.new_color = {ServiceChoice};
        fire = requestSR({'Counter2Staff',1});
    else
        fire = 0;
    end

elseif strcmp(transition.name, 'tCMakeBurg')
    fire = requestSR({'BurgStaff',1});

elseif strcmp(transition.name, 'tCBurgStockRefill')
    fire = requestSR({'ExtraStaff',1}) &&
le(ntokens('pCBurgIng'),global_info.CBurgRefillTriggerMark);

elseif strcmp(transition.name, 'tCMakeDrink')
    fire = requestSR({'DrinkStaff',1});

elseif strcmp(transition.name, 'tCDrinkStockRefill')
    fire = requestSR({'ExtraStaff',1}) &&
le(ntokens('pCDrinkIng'),global_info.CDrinkRefillTriggerMark);

elseif strcmp(transition.name, 'tC1RcvOrder')
    fire = requestSR({'Counter1Staff',1});

elseif strcmp(transition.name, 'tC2RcvOrder')
    fire = requestSR({'Counter2Staff',1});

elseif strcmp(transition.name, 'tCBagStockRefill')
    fire = requestSR({'ExtraStaff',1}) &&
le(ntokens('pCTABag'),global_info.CTABagRefillTriggerMark);
```

```
elseif strcmp(transition.name, 'tCTrayStockRefill')
    fire = requestSR({'ExtraStaff',1}) &&
le(ntokens('pCDITray'),global_info.CDITrayRefillTriggerMark);

elseif strcmp(transition.name, 'tCCollectTray')
    fire = requestSR({'ExtraStaff',1}) &&
ge(ntokens('pCDIUsedTray'),global_info.CDITrayCollect);

elseif strcmp(transition.name, 'tCCleanTray')
    fire = requestSR({'ExtraStaff',1}) &&
ge(ntokens('pCDICollectedTray'),global_info.CDITrayClean);

% Checking if the service type is TakeAway
elseif strcmp(transition.name, 'tCTakeAway')
    tokIDOrderAtt = tokenAnyColor('pCRcvOrder',1,{'TakeAway'});
    if tokIDOrderAtt == 0
        fire=0;
    else
        transition.selected_tokens = tokIDOrderAtt;
        transition.override = 1;
        transition.new_color = {};
        fire = tokIDOrderAtt;
    end

% Checking if the service type is DineIn
elseif strcmp(transition.name, 'tCDineIn')
    tokIDOrderAtt = tokenAnyColor('pCRcvOrder',1,{'DineIn'});
    if tokIDOrderAtt == 0
        fire=0;
    else
        transition.selected_tokens = tokIDOrderAtt;
        transition.override = 1;
        transition.new_color = {'Tray'};
        fire = tokIDOrderAtt;
    end

%Checking Condition for ByPass to fire or not
elseif strcmp(transition.name, 'tCByPass')
    fire = ge(ntokens('pCQue'),global_info.CQueCap) ||
le(ntokens('pCStockBurgIng'),global_info.CustRedirectBOBurgStock);

end
```

MOD_Counter_POST.m

```
function [] = MOD_Counter_POST (transition)
global PN
%disp(transition.name);
disp(['In Counter Post enabled trans is: ', transition.name]);
disp(markings_string(PN.X));
release(); %Release all resources after firing
```

MOD_Toilet_PRE.m

```
function [fire, transition] = MOD_Toilet_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In Toilet PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));

if strcmp(transition.name, 'tTIn')
    fire = le(ntokens('pTInUse'),global_info.TAvaialble);
elseif strcmp(transition.name, 'tTByPass')
    fire= ge(ntokens('pTQue'),global_info.TQueCap);
end
```

MOD_Table_PRE.m

```
function [fire, transition] = MOD_Table_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In Table PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));

if strcmp(transition.name, 'tTabByPass')
    fire= ge(ntokens('pTabQue'),global_info.TabQueCap);
elseif strcmp(transition.name, 'tTabFind')
    fire = le(ntokens('pTabInUse'),global_info.TabAvaialble);
end
```

MOD_Kidszone_PRE.m

```
function [fire, transition] = MOD_Kidszone_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In KZ PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));

if strcmp(transition.name, 'tKZChooseGame')
    %randomly generate service type: TakeAway or DineIn?
    Choosegame = randi([1,2]);
    if Choosegame == 1
        game = 'jumping';
    else
        game = 'sliding';
    end
    %Assign the randomly generated color to the token
    transition.new_color = {game};
```

```
    fire = le(ntokens('pKZPlayZone'),global_info.KZAvaialble);

elseif strcmp(transition.name, 'tKZJumping')
    tokIDOrderAtt = tokenAnyColor('pKZPlayZone',1,{'jumping'});
    if tokIDOrderAtt == 0
        fire=0;
    else
        transition.selected_tokens = tokIDOrderAtt;
        transition.override = 1;
        transition.new_color = {};
        fire = tokIDOrderAtt;
    end

elseif strcmp(transition.name, 'tKZSliding')
    tokIDOrderAtt = tokenAnyColor('pKZPlayZone',1,{'sliding'});
    if tokIDOrderAtt == 0
        fire=0;
    else
        transition.selected_tokens = tokIDOrderAtt;
        transition.override = 1;
        transition.new_color = {};
        fire = tokIDOrderAtt;
    end

elseif strcmp(transition.name, 'tKZByPass')
    fire= ge(ntokens('pKZQue'),global_info.KZQueCap);
end
```

MOD_Chargingdock_PRE.m

```
function [fire, transition] = MOD_Chargingdock_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
disp(['In CD PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));

% Assignining "Service Type" Color in tCounterEnter transition
if strcmp(transition.name, 'tCDFindDock')
    fire = le(ntokens('pCDInUse'),global_info.CDAvaialble);
elseif strcmp(transition.name, 'tCDByPass')
    fire= ge(ntokens('pCDQue'),global_info.CDQueCap);
end
```

COMMON_PRE.m

```
function [fire, transition] = COMMON_PRE (transition)
global global_info
global PN
fire = 1;
%disp(transition.name);
```

```
disp(['In Common PRE enabled trans is: ', transition.name]);
disp(markings_string(PN.X));

CurrentTime = current_time();

if strcmp(transition.name, 'tIMCExit')
    if ge(CurrentTime, global_info.ClosingTime) %Transition will not fire if Current
Time passes Closing Time
        fire = 1;
    else
        probabilityVariable = randi([1,global_info.ExitProbability]); %Probability of
Exiting FastFood Restaurant from IMC
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end

elseif strcmp(transition.name, 'tCEnter')   %Probability of Entering Counter Module
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        probabilityVariable = randi([1,global_info.CounterEnterProbability]);
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end

elseif strcmp(transition.name, 'tTabEnter')   %Probability of Entering Table Module
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        probabilityVariable = randi([1,global_info.TableEnterProbability]);
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end

elseif strcmp(transition.name, 'tTEnter')   %Probability of Entering Toilet Module
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        probabilityVariable = randi([1,global_info.ToiletEnterProbability]);
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end
```

```
elseif strcmp(transition.name, 'tKZEnter')   %Probability of Entering KidsZone Module
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        probabilityVariable = randi([1,global_info.KidsZoneEnterProbability]);
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end

elseif strcmp(transition.name, 'tCDEnter')   %Probability of Entering ChargingDock
Module
    if ge(CurrentTime, global_info.ClosingTime)
        fire = 0;
    else
        probabilityVariable = randi([1,global_info.ChargDockEnterProbability]);
        if probabilityVariable == 1
            fire = 1;
        else
            fire = 0;
        end
    end

end
```

COMMON_POST.m

```
function [] = COMMON_POST (transition)
global PN
%disp(transition.name);
disp(['In Common POST enabled trans is: ', transition.name]);
disp(markings_string(PN.X));
release(); %Release all resources after firing
```