Project 1: Project Enigma
Name: Sena Yevenyo
Section: CS-150-03

A. Introduction

The goal of this project is to simulate the operation of a restaurant that offers either of these choices: bagels, hoagies or pizzas. The main objective is to collect data about the operation of such restaurant that runs for 24 hours in Lafayette's College new student-building which would be completed Fall of 2020.

Objectives

The simulation is designed to collect data about:

1. Average wait-time of customers at the restaurant during normal operations and during peak hours.
2. Satisfaction of customers both at peak times and normal times
3. Order volumes
4. General revenue
5. General staff costs
6. Staff utilization

All these data should be utilized to conclude what restaurant is much appropriate for the school.

Design Constraints

1. The simulation is designed to simulate and generate data for 12 hour running time and not the full 24 hours.
2. Ingredient cost relative to selling price of items sold are not considered in this simulation.
3. Prize of items should be the about the same prize of the same items sold by shops around Lafayette College and Easton so as to not price items out of market, and also skewing expected revenue data.
4. Peak hours are not constant time range and should vary.
5. Customers arrival are also at irregular times.
6. Customers can order multiple numbers of items.
7. Employees are assumed to take breaks but posts are assumed to be fully staffed as the restaurant runs.
8. Employee's salaries should be similar to the salaries of employees of eateries in and around Lafayette College.
9. For items sold, bagels should be the fastest to cook, but the cheapest of the three items sold.
10. Pizza should be the most expensive, but slowest to cook.
11. Hoagies are midway with pricing and rate of production.
12. A customer can order 2-6 bagels, 1-2 pizzas and 1-3 hoagies per order.

B. Design Approach

This simulation is an event driven simulation. Classes used in this simulation are eleven in all. The classes are:

1. Food
2. Bagel

3. Hoagie
4. Pizza
5. GlobalClock
6. Cook
7. Customer
8. Cashier
9. Order
10. Restaurant
11. SimulationController

All methods, instance and static variables explained in the following chapters can be obtained and explored in detail in the JavaDoc documentation.

1. Food Class

The Food class is a class whose object instantiation is to represent a generic food or meal made by a restaurant or eatery. The properties of an object instantiation of this class is the price of this meal. This is a static and immutable variable that can be accessed outside of the class. Another property is the time taken for the meal to cook which is a private variable and can be accessed by a method on the class' object.

The simulation design constraints required the order number by order by customer in the restaurant should vary based on food type ordered. Thus, the value of range of number of meals per customer order is an attribute of this class' object.

2. Bagel Class

The Bagel class inherits all its properties from the Food class. Thus, all methods applicable to Food class is applicable to Bagel class.

The price of the Bagel is assumed to be $0.99. Prices of bagels in supermarkets are often lower than this price, but breakfast eateries sell them at a markup price. Wawa sells bagels at $0.95 but other eateries such as Panera and Starbucks sell them for just above $1. This seems to be priced within market range.

The time taken to prepare a Bagel is assumed to be 2 minutes of real world time. Toasting and spreading condiments over bagels and delivering them to customers do not tend to take a lot of time.

3. Hoagie Class

The Hoagie class inherits all its properties from the Food class. Thus, all methods applicable to Food class is applicable to Hoagie class.

The price of a Hoagie is assumed to be $4.20. This price was compared to the price of a full Shorti Wawa Hoagie is $4.39 dollars. Wawa is the best competitor to the new restaurant so it would be a smart business decision to undercut the competition to win patrons.

Some recipe websites put the cook time of a Hoagie around 10 to 15 minutes. Logically, an eatery would have all the ingredients and filling elements cut up and cooked before the customer makes an order. Filling up bread with condiments and toasting the bread should not take an overwhelming amount of time. Ballpark estimate would place it 3 to 5 minutes of preparation. The time taken to make a hoagie is therefore assumed to be 5 minutes so as to make observable difference between a Bagel restaurant and a Hoagie restaurant much more clear.

4. **Pizza Class**

The Pizza class inherits all its properties from the Food class. Thus, all methods applicable to Food class is applicable to Pizza class.

The price of a Pizza meal is assumed to be $10.00. Price range of a medium sized pizza is between $9.99 to $10.99. Pricing the pizza at the lower end of the spectrum attracts customers much more.

The time taken to cook a Pizza meal is assumed to be 12 minutes. Some websites place the range between 7-15 minutes and 8-13 minutes. Thus, a 12-minute cook time would be a good candidate as the average cook time of a pizza meal.

5. **GlobalClock Class**

The GlobalClock class is the universal time in the simulation measured in ‘quants’ or ‘ticks’. The value of a single ‘quant’ in the simulation is assumed to be one minute in real world time.

All the methods and variables of this class are static and thus can be accessed by any object and class in the simulation. There is one static method to obtain the universal time and another to increase this time value.

6. **Cook Class**

The Cook class represents a chef or a cook at an eatery or restaurant.

The Cook class as a final static value of the salary of the Cook which was placed at $15.40 per hour. This is reportedly the national average by Indeed.com.

There is a class level function that outputs the total cost of Cook objects used in a simulation and this makes it easier to compute cost at the end of the simulation.

Function of a Cook

Cook objects are instantiated by giving an ID for easy identification, a Food variable which indicates the food prepared by the Cook object, a numeric value that indicates time left to complete meal cooked and boolean variables that indicate if a Cook object is done and when an object is available for a task.

Objects receive orders by a method that sets the time left to complete order and changes the status of the variables that indicate availability and task completeness.
A method prompts the object to continue making the order for each event in the simulation.

When done, the availability and task completeness indicators switch back on. There is another method that resets the task completeness indicator off and this function's significance would be made much clearer in the subsequent explanations.

7. Customer Class

The Customer class is a representation of a customer in a restaurant. As such, it has properties that are integral for data collection such as variables that store satisfaction and time spent in the restaurant. The class also contains variables that determine the range of number of meals ordered per customer and this is dependent on meal requested.

Function of a Customer

Customer objects are instantiated requiring an ID for comparison purposes, Food variable to indicate food about to be ordered. A third variable indicates if the Customer entered at a busy time or at a normal operating time. This variable is relevant for the Cashier class.

The satisfaction value is initially set to 100.0, the highest value. This decreases by 0.5 per time spent waiting to make the order and 0.25 per time waiting for an order where an order in this case means all the meals requested by the object bundled as one entity. If the waiting time for each range is greater than 15 minutes, the penalty satisfaction deducted per unit time is doubled. This computation felt logical based on the assumption that objects would be much more infuriated when they spent time trying to make an order, than waiting for one.

Methods that obtain instance variables are in place to facilitate interaction of customers with other objects in the simulation. A static variable records the number of customers instantiated in the simulation.

8. Cashier Class

The function of the Cashier class is to provide elements that make the behavior of object instances of this class similar to the operation of a Cashier at a restaurant. It has an ID for identification and a variable that indicates its availability. Similar to the Cook class,
there is a class level function that outputs the total cost of Cashier objects used in a simulation and this makes it easier to compute cost at the end of the simulation.

There are global values where the satisfaction of customers, time spent by customers at both busy and normal times are stored. Number of order volumes is stored as a global static value where all Cashier objects can log their data obtained from customer objects during checkout.

Function of a Cashier

A Cashier object accepts orders and checks out customers from the restaurant. Therefore the rate of entry and exit is dependent on their numbers. The rate of entry is equal to the rate of checking out. Therefore, a Cashier object is assumed to only accept orders and check out customer only once for each task for a single time event.

At the check out end of the simulation, the object obtains integral data such as satisfaction and time spent and logs it at the class level.

Averages of these values are computed and obtained by classes at the global level and can be accessed by methods for data collection.

9. Order Class

The Order class represents the receipt of obtained by a customer at a restaurant during checkout, but without the prices. Objects instantiated have an order ID which is same as the Customer object who made the order, the number of meals to be cooked for the order to be complete, a Food variable indicating the food to be cooked and variable indicators which show if an order is completed.

Order #	0
Food	Bagel
Number of items	2
Status	Not completed

Figure 1: A tabular representation of an Order object

The Order class has methods that can access these values, and manipulate other variables that change the status of the order if it is completed or not. These variables are updated when a Cook object takes on the task of cooking a meal on the Order sheet or completes a meal on this Order sheet.

10. Restaurant Class

The Restaurant class in this simulation represents a functioning restaurant. It has methods that control data flow within the simulation within itself and other classes. Thus, these methods can be described as prompts that initiate tasks to be completed.

To instantiate a class object requires a Food variable which indicates the meal to be cooked by restaurant, the number of staff (i.e number of Cashier and Cook objects in the restaurant).

There are methods that move data around (i.e pushing Customer objects into restaurant and out) which will be explained in subsequent paragraphs.

11. SimulationController Class

This is the class that controls the simulation of the whole program. The main method of this class accepts the filename for the log file, the data file and the parameters for the simulation. For example the number of staff for the simulation.

How The Simulation Runs

The simulation program runs inside the SimulationController class inside a while loop that terminates when ticks is equal to 720 ticks, equivalent to 12 hours of real world time.

The rate at which customers get into the restaurant is determined by a Random class objects which adds 0 to 1 customer every tick and 3 to 7 customers during rush hour per tick. The time at which rush hour ends and begins is also controlled by a Random class object and has no limit on the time it lasts for.

As earlier stated, the Restaurant class has methods that prompts data flow in the system and interacts with methods of other classes to get the specific job, expected by the simulation of the restaurant operation..

First, a real operating restaurant opens its doors to customers using the Restaurant class method “receiveCustomers()”. Rate of customers entering a restaurant is determined by a Random class object which adds from zero customers to a single customer per tick. This is referred to as normal time operation.

Customers are created, given a unique ID value and then added to a waitingToOrder Priority Queue according to their ID. Lower ID means earliest Customer object instantiated.

Secondly, the method of the Restaurant class acceptOrders() is called to accept orders from Customers waiting in the WaitingToOrder Priority Queue based on number of cashiers available at the restaurant. Lower cashiers means lower number of orders made per unit time. The Customer objects who have made their orders are moved to waitingForOrder Priority Queue. Orders made by Customers are placed in an OrderList Priority Queue.

Then, the method executeOrders() is called to execute Order objects in the OrderList Priority Queue. First, the head of the OrderList is peeked, and then, the LinkedList of Cook objects is parsed to find the available Cook objects and a new task is given depending on availability. As meals on the Order object are executed, the quantity left to cook decreases and when that value reaches zero, total Order is completed and this, Order object is moved to checkoutList Priority Queue.

After, the method receiveOrders() is called to check if any of the cooks are done completing the task given to them by the executeOrders() command. A method of the Cook class isDone() indicates if a Cook object is done with order or not. Then, the Customer object for whose object has been completed is moved from waitingForOrder Priority List to waitingToCheckout Priority List.

Finally, the method checkout() which polls Customers out of the waitingToCheckout Priority Queue collecting data such as satisfaction, wait time and number of order made when interacting with Cashier objects.

Data Flow

For Customers:

Enter restaurant → Wait to make order → Make order → Wait to receive order → Receive order → Checkout.

For Orders:
Order made → Execute order → Wait for order to be completed → Receive made order → Order given to customer.

C. Method

To compare the profitability of the three restaurants, it would be preferable to compare the three restaurants at their highest efficiency possible. Profitability depend on costs and order volumes and number of returning customers so as to ensure continuous cash flow for the business. Using less staff would result in low cost but would hurt customer satisfaction and hurt cash flow into restaurant. Using more staff would increase customer satisfaction but too much would hurt the profitability of the restaurant.

Thus, a Staff Utilization Index (SUI) is created to find the lowest number of staff for which the highest customer satisfaction is obtained.

This is first computed to find the highest SUI for each restaurant.

Then for each restaurant at the highest SUI, the computation for profitability, order volumes customer wait times at busy times and normal operations are computed. These graphs are then compared to find the most suitable restaurant for Lafayette College.

D. Data and Analysis

1. Finding the Highest Staff Utilization Index

The meal served at restaurant was Bagel. Without consideration with profit generated, the hypothesis of the SUI giving the highest efficiency of the restaurant possible.

SUI data generated was collected 5 times and the average of these values were taken. An average of 5 was selected to ensure the probability of setting up the simulation with the same conditions would give a value around the average.

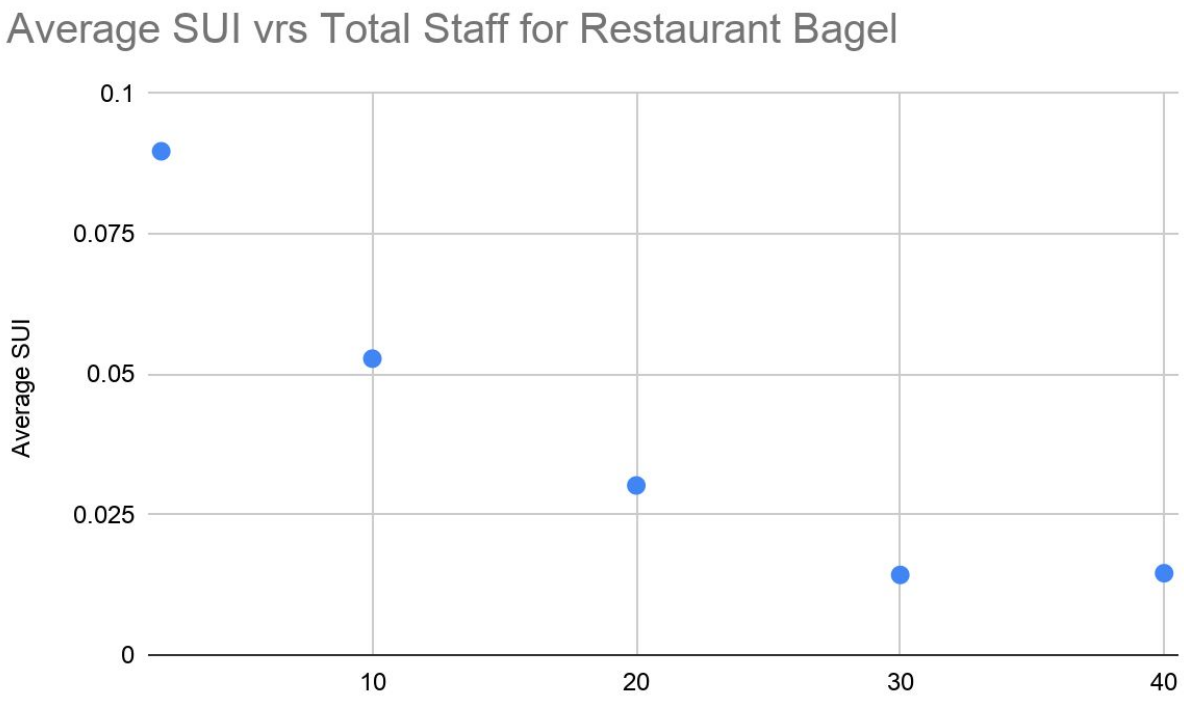

Figure 2: Scatter plot of Average SUI against total staff for Restaurant Bagel

The data here shows the inverse of the expected. Higher SUIs were obtained for lower staff numbers instead of higher ones, and this would not be the optimum since the restaurant would be getting lower order volumes compared to a higher number of staff.

The second reasoning is that the queue moves so slow in the restaurant that disgruntled customers are not checked out before the simulation ends thus, it does not give a clear picture of what all the patrons who use the restaurant might feel.

This result would be non-specific to Restaurant Bagel. Cost stays the same per staff and the queue would move slow for all restaurant types due to low staff numbers. Much more happier customers are checked out early and disgruntled customers would not be checked out before simulation ends. This model is not fit as a way to compare efficiency and thus, the model is disregarded.

The new model would be based on these assumptions.

1. The average number of cashiers for each restaurant would be assumed to be 5 for our simulation for all restaurants.
2. The average number of chefs for each restaurant would be assumed to be 10 for our simulation for all restaurants.

These numbers are assumed due to the nature of the restaurant built. Would-be Competitors such as Wawa and Lower Farinon tend to have cashier staff lower than these values.

The number of chefs assumed is chosen based on the number of staffing arrangements in Lower Farinon which is about 10 chefs during peak hours.

2. Data Analysis

Data collected would be collected by a number of 6 to study trends in data produced by simulation.

A. Order Volumes

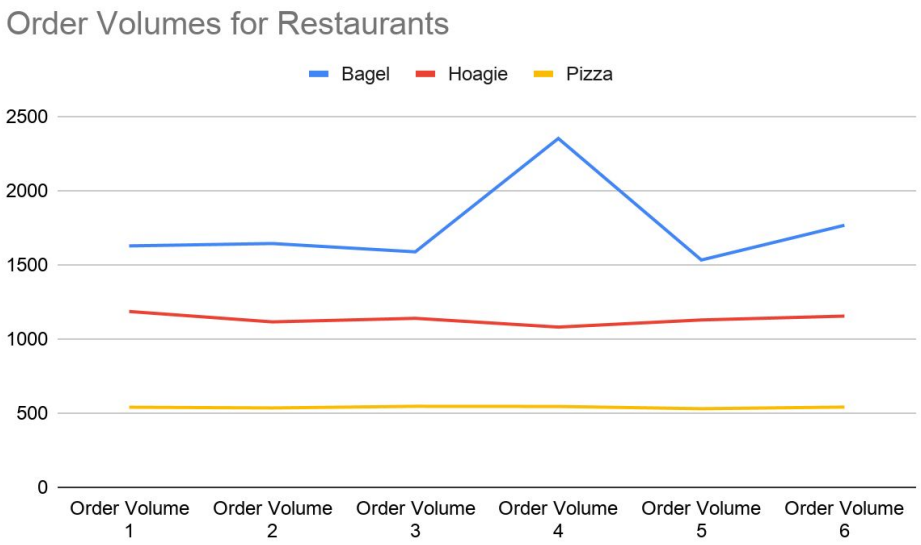

Figure 3: Linear plot of order volumes against for Restaurant Bagel, Hoagie and Pizza

Order volumes for bagels are more than that of hoagies and pizzas. This is a result of tendency to be ordered in large volumes compared to hoagies and pizzas. This is an item which can sell fast, especially during rush hour.

B. Revenues and Profits

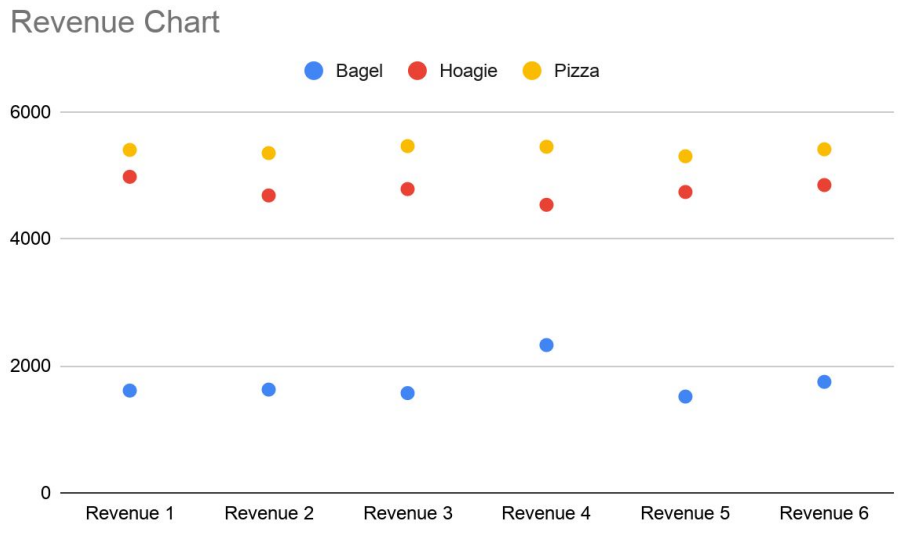

Figure 4: Scatter plot of revenue in dollars obtained from bagels, hoagies and pizzas

From the graph, pizza has an average revenue which is clearly higher for both hoagies and bagels. Pizza is right below it in revenues. Bagels are the least with revenues averaging around 1900 US dollars. Not so much money from the large order numbers.

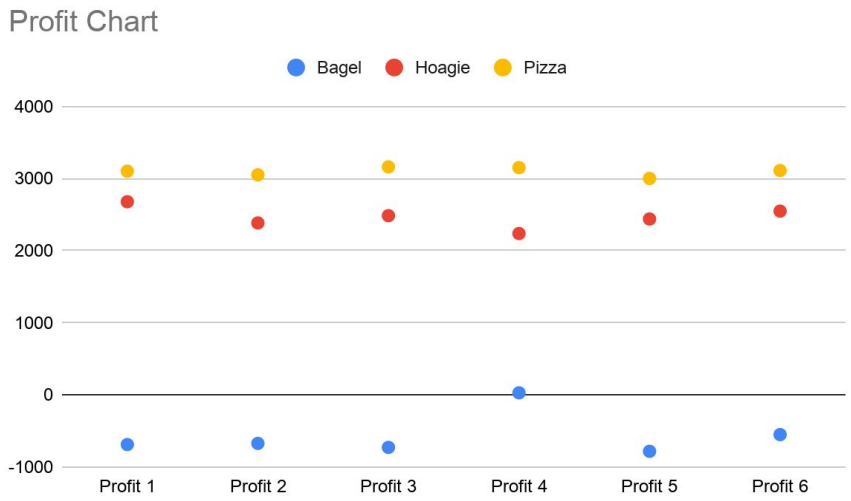

Figure 5: Scatter plot of profit in dollars obtained from bagels, hoagies and pizzas

Restaurants from simulation are running at a loss when bagels are served for a 12-hour runtime. Hoagies and Pizzas are always running at a profit during these simulations with the highest profits going to pizzas.

C. Customer Satisfaction

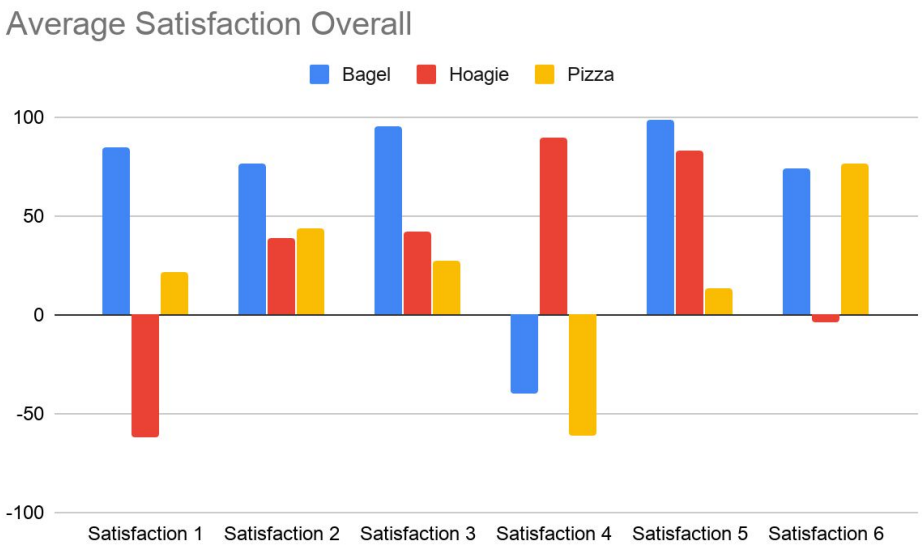

Figure 5: A bar chart that shows average Customer Satisfaction on a scale of -100 to 100. 100 means highly recommend and would visit again. -100 means does not recommend and would never visit again.

Overall customer satisfaction of bagels trumps the rest of the meals. It has consistently higher satisfaction. Hoagies and pizzas are difficult to separate in this case.

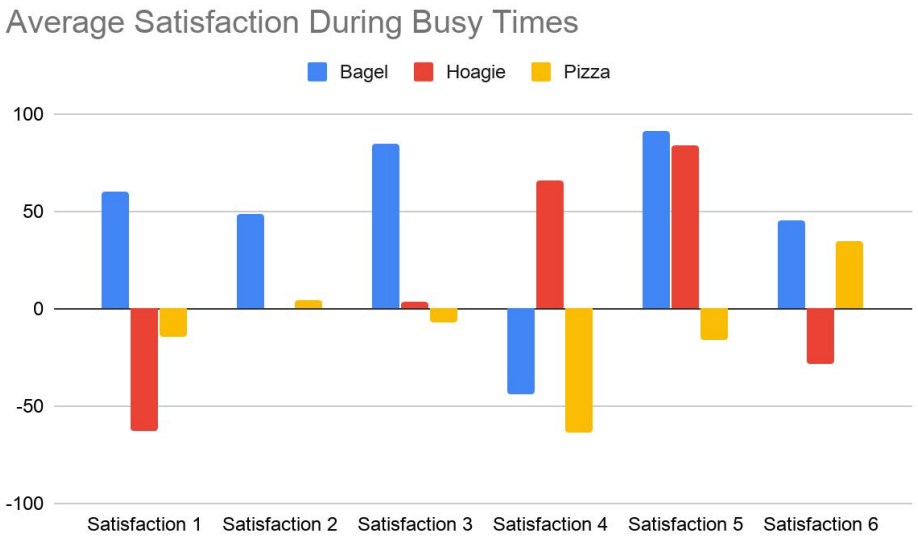

Figure 6: A bar chart that shows average Customer Satisfaction during busy operations on a scale of -100 to 100. 100 means highly recommend and would visit again. -100 means does not recommend and would never visit again.

Here again Bagels beat Hoagies and Pizzas in this round. Dissatisfaction with pizza is much more clearer here too when compared to bagels.

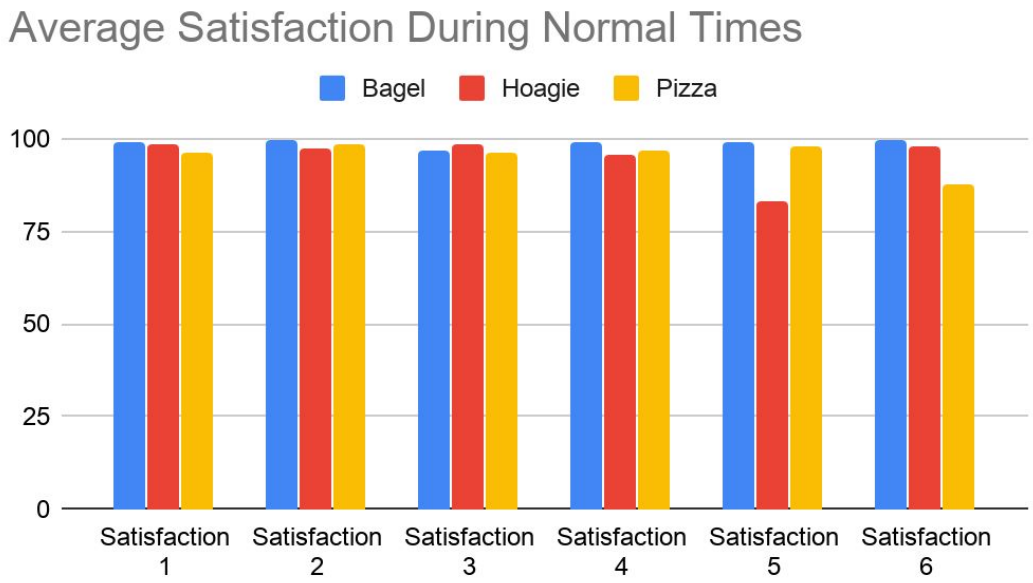

Figure 7: A bar chart that shows average Customer Satisfaction during normal operations on a scale of -100 to 100. 100 means highly recommend and would visit again. -100 means does not recommend and would never visit again.

Here the customer satisfaction values are neck and neck with each other, but bagels are the best with customer satisfaction.

D. Time Spent at Restaurant

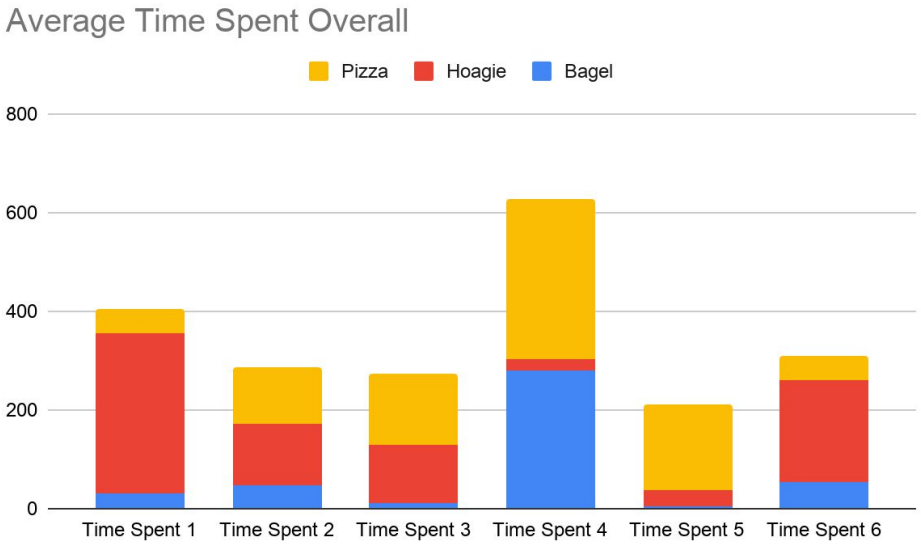

Figure 8:Stacked bar chart of time spent overall obtained from restaurants in seconds

Customers who ordered pizza consistently spent a longer time when obtaining their meals. Hoagies were the next. Bagels overall spent a shorter time in getting their meals.

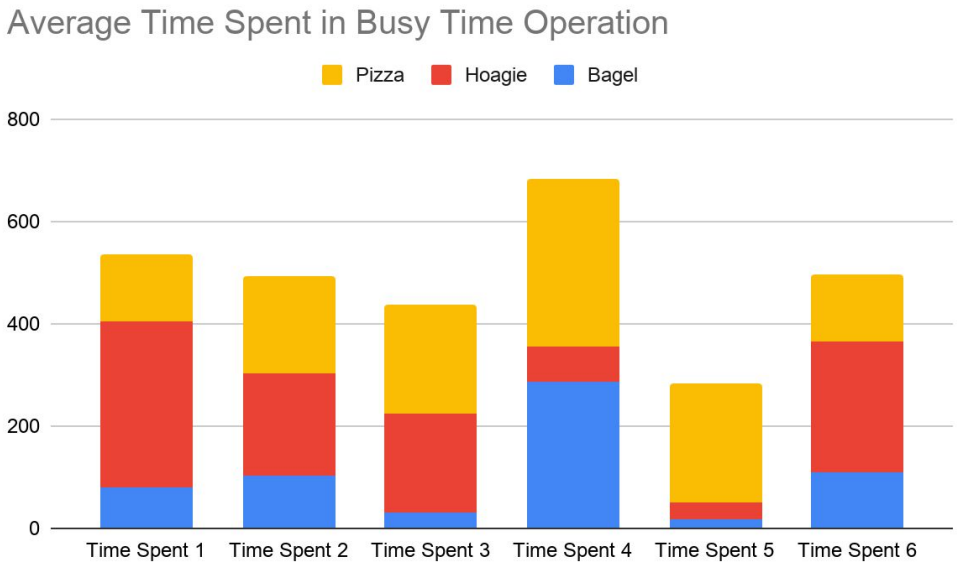

Figure 9:Stacked bar chart of time spent in busy time obtained from restaurants in seconds

The same trend is noticed here. Customers who ordered pizza consistently spent a longer time when obtaining their meals. Hoagies were second in line.. Bagel customers overall spent a shorter time in getting their meals.

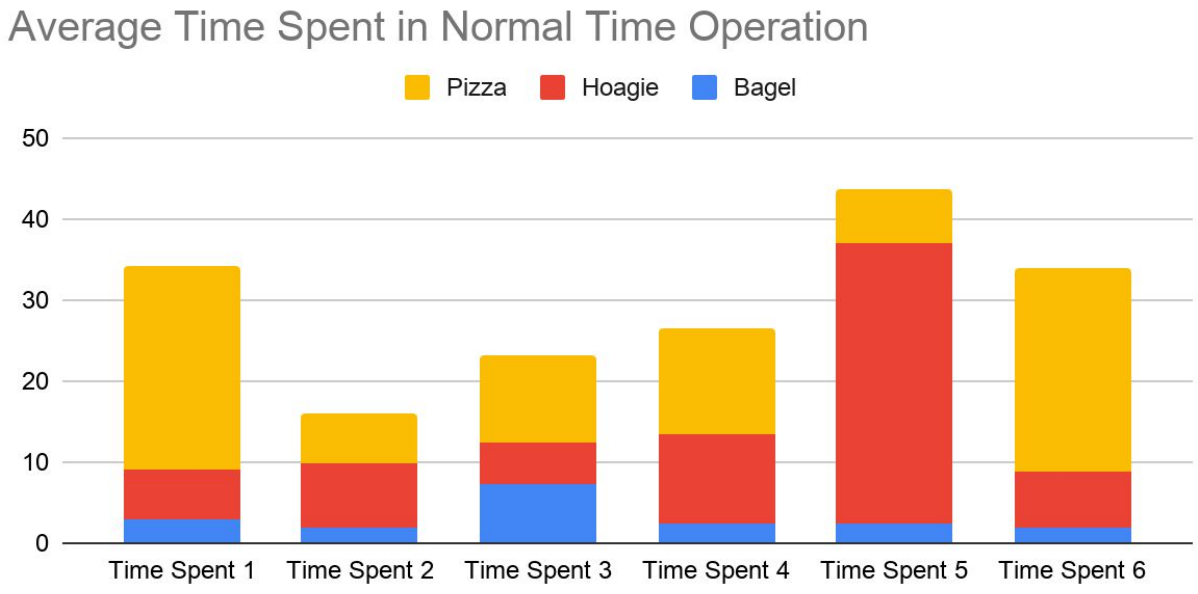

Figure 9:Stacked bar chart of time spent in normal time obtained from restaurants in seconds

The same trend is noticed here too. Customers who ordered pizza consistently spent a longer time when obtaining their meals. Hoagies were second in line.. Bagel customers overall spent a shorter time in getting their meals.

E. Conclusion

From our analysis we noticed that bagels were the fastest to sell and most customers were happy with their bagel meals. The unfortunate thing is given the assumptions, bagels when used as the main meal of the restaurant would make it run at losses and that is not a good way to run a business.

Pizzas generated the most revenue but had the poorest wait time for meals during normal operations and rush hours. Customers are not very happy with their orders compared to other meals.

Hoagies is midway between pizzas and hoagies. It generates a lot of revenue and order volumes which are very close to those of pizzas and has better wait time when compared to pizzas. Its satisfaction with customers are not very distinct from that of pizzas, but its faster completion time gives it an edge over pizza.

Final conclusion is the school should go with a restaurant that serves hoagies most of its time of opening. Customer satisfaction is very good during normal operations and handles better during rush hours compared to pizza. Generates revenues and volumes which are closer to pizzas also.

Another important suggestion is to have once a week during the restaurant's operation 'Bagel Lunch'. Bagels move very fast and attract customers due to its high customer satisfaction. Serving pizzas during late weekend nights would be a good idea due to low potential for rush hours during nights and its high profit margins.

F. References

(2018, October 6). Retrieved from https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html.

(2018, October 6). Retrieved from https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html.

(2019, March 6). Retrieved from https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html.

Munoz, S., & Bandler, H. (n.d.). Wawa Prices. Retrieved from https://www.fastfoodmenuprices.com/wawa-prices/.

Bologna, M., Beth, Ferrell, R., Gill, Haney, W., Johnson, K., … Alexandria Pastrami. (n.d.). Panera Bread Prices. Retrieved from https://www.fastfoodmenuprices.com/panera-bread-prices/.

Chef Salaries in the United States. (n.d.). Retrieved from https://www.indeed.com/salaries/Chef-Salaries.