Project Report

Chamanthi Pyneni -700755941

1. Project Title:

Group number: 8.

|  |
| --- |
| **Restaurant function Using Multi-Threading and Synchronization** |

2. Team members:

|  |  |  |
| --- | --- | --- |
| Name | 700# | Responsibility |
| Chamanthi Pyneni | 700755941 | Multi Threading |
| Pavan Kalyan Kambala | 700756964 | Design of classes and methods |
| Hema Sri Perumalla | 700758435 | Multi Threading |
| Yagna Yashwanth Nimmagadda | 700757921 | Design of classes and methods |

3. Project Summary (What you have done in this project or what is your contribution? What you have learned in this project):

Functioning of Restaurant involves many tasks which has to be performed in an order so as to satisfy the customer. These tasks includes managing arrival time of customers, assigning tables, cooks available to cook and machines availability for burgers fries and coke. There are two main thread classes in our project diner class and cook class.

In the Cook class, I've implemented several methods to manage the cooking process efficiently. Each Cook instance represents a thread responsible for processing orders in the restaurant. Here's a breakdown of the implemented functionalities:

The Constructor initializes the Cook instance with a unique identifier. Methods handles the processing of orders. It takes an order and the corresponding diner ID as parameters. It calculates the total time needed to prepare the order by simulating the cooking process for burgers, fries, and coke. It utilizes three Boolean flags to track the completion status of each item in the order. The method iterates until all items are cooked. Inside the loop, it determines which item to process next based on the machine with the least processing time. It updates the processing time accordingly for each item and marks it as completed once done.

Least Time Machine Method is a private method which finds the machine with the minimum starting time among the available machines. It takes an array of Boolean flags indicating the completion status of each machine and an array of start times for each machine. It iterates through the machines, excluding those already completed, and selects the one with the minimum starting time. There is a stop method which interrupts the Cook thread, stopping its execution. Run method Overrides the run method of the Thread class. Inside this method, the Cook thread continuously checks for new orders from the OrderList. If there are no orders, it waits for notification from the OrderList. Once an order is available, the Cook processes it by calling the processing method and updates the diner's order processing time accordingly.

This implementation allows efficient management of cooking tasks in a restaurant setting using multithreading and synchronization concepts. Each Cook thread handles orders independently, ensuring smooth operation and timely service to customers.

I have learned how multithreading works and how to implement it in a real time example. Through this project I have gained hands on experience in multithreading and synchronization concepts.

4. Conclusion (Is the project completed successfully as planed? What you have learned from this project?):

Yes, the project has been successfully completed as we have planned. Each of us have done their work perfectly which made us to finish the project in time and outcome as we expected. I have learned the importance of multithreading and synchronization concepts. Through this project I have gained hands on experience. I have learned how to implement them in a real time scenario.