|  |  |
| --- | --- |
| **Item** | **Answer** |
| **Section** | G1 |
| **Name(s)** | *Adeline Chin Wen Jie (Adeline)*  *Gokarn Mallika Nitin (Mallika)* |
| **Brief description of project** | *Our project is a restaurant simulator. The owner of Gluttons Bay Restaurant is interested in looking at the difference between his company and the restaurant Subway, in terms of service.*  *Subway has a very linear process and attends to one customer fully at a time and then moves onto the next customer. Gluttons Bay on the other hand is a Vegan 5 course restaurant that sonsists of soup, salad, appetiser, Main Course and Dessert. Gluttons Bay provides seating as well. In order to compare Subway and Gluttons Bay, a few assumptions have been made, they are as follows:*   1. *The food preparation overall, takes the same amount of time for chefs of both Gluttons Bay and Subway. The division of time on each part of preparation is as shown below:*      1. *The time taken by each customer on average overall, to decide on their food and thereafter let the waiter place the order is the same between both Gluttons Bay and Subway. The division of time on each part of ordering is as shown below:*          1. *The time taken by each waiter on average overall, to serve customers their food is the same between both Gluttons Bay and Subway. The division of time on each part of ordering is as shown below:*      1. *The time taken by each customer on average overall, to eat the food served is the same between both Gluttons Bay and Subway. The division of time on each part of ordering is as shown below:* |
| **Justification for multi-threading** | *Each waiter can serve a customer only one course at a time. Once the course is completed, he or she must wait for the chef to complete cooking the next course and thereafter the waiter to serve this course. However, there are a minimum of 5 and a maximum of 10 customers. Also, each customer is tended to by one and only one waiter. This leads the need for concurrency.*  *A single threaded program would have only one waiter working at a time which would lead to hungry angry and impatient customers who might leave their meal midway.* |
| **Transactional integrity** | *Were there any potential race conditions; if so, how did you resolve them? Or are there still unresolved race conditions that you are aware of?* |
| **Performance** | *Is there anything noteworthy you did that improved performance? (include stats if any)* |
| **Evidence of exploration** | *Is there anything noteworthy that you did that required additional research (of topics not covered in class)?* |
| **Innovation** | *Why is your project innovative?* |
| **Adherence to coding conventions & good practices** | *Is there anything noteworthy that you want to mention? E.g. usage of a well-known multi-threading “best-practice”, or OO design pattern etc.* |
| **References/Acknowledgement** | *We used third party code from github user heidtJJ (Jared Heidt) as our starting point and built on it*  *The code to his project is in the folder titled reference.*  *We have also used the StopWatch* |