**COIT13229 Applied Distributed Systems**

**Term 1, 2024**

**Assessment 3**

Lecturer: Ather Saeed

Tutor: Ather Saeed

Prepared by:

Justin Hastings 12147349

Liam Elsum

Due Date: 07/06/2024

Table of Contents

[Assumptions 3](#_Toc168604890)

[System Architecture 3](#_Toc168604891)

[Test Instructions 3](#_Toc168604892)

[Synchronisation 3](#_Toc168604893)

[Conclusion 4](#_Toc168604894)

[References 5](#_Toc168604895)

# Assumptions

While investigating the requirements of the task, it was apparent that some assumptions would be required to clarify some ambiguous aspects. These assumptions included:

* There are 5 initial postcodes and 5 days of the week, so each postcode will have its own delivery day until an admin or staff user creates new delivery schedules.
* Each order line will be saved with GST included in its total.
* There is an infinite supply of products, hence we don’t need to handle out of stock occurrences.
* Since payment processing is not required, a simple button click will simulate the payment process.
* Customers will have the option to select a preferred delivery time between 8am and 5pm.
* Users must register as a customer in order to place orders.
* Admin and staff users cannot place orders, they must create their own customer accounts in order to place orders.

# System Architecture

# Test Instructions

# Synchronisation

Synchronisation in this context refers to ensuring the integrity of the data and that it is consistently maintained across the application. It focuses on locking out specific functions, usually database calls, whilst they are in use by a client to prevent conflicts by other clients trying to access or manipulate the same data in a separate transaction. Most distributed systems utilise multiple threads and processes to access data in concurrent transactions. Synchronisation is a necessity for these concurrent transactions to prevent corrupted and inconsistent outcomes (Lenovo, n.d.). If synchronisation is not utilised, or is incorrectly implemented, the application could generate drastically different outcomes for functions. For example, if staff user 1 is editing product 1 to change the price, but at the same time in a separate transaction, staff user 2 removes product 1. This could lead to a couple different outcomes, such as staff user 1’s changes being lost when staff user 2’s delete operation is committed. Another potential outcome is that staff user 1’s changes are now trying to reference a record that has already been deleted and no longer exists because staff user 2’s delete transaction was committed before the update, hence landing the database in an inconsistent state. This scenario is faced similarly by the majority of mutating functions in this application, including but not limited to editing users, delivery schedules, orders etc. Therefore, synchronisation, and the correct implementation of it, is imperative to the success of this application.

# Conclusion

# References

Lenovo, n.d., *Whys is Synchronization Important in Distributed Systems & Databases*, viewed 06 June 2024, <https://www.lenovo.com/au/en/glossary/syn/?orgRef=https%253A%252F%252Fwww.google.com%252F>