Date: ____________

DBS - Assignment - 4

Q1)

a) Properties violated are atomicity & consistency. Atomicity is broken because the transaction was only partially completed (ie. wallet deducted, driver accepted but confirmation not shown).

Consistency is violated because database is left in an inconsistent state (ie. the customer's balance was reduced without a corresponding ride ~~confirmation~~ confirmation).

b) Violation occured because transaction was not treated as an atomic unit & the system crashed before finalizing all steps. In database terms, there was no proper commit or rollback mechanisms in place.

The writes to the driver acceptance & wallet balance were persisted, but the final commit which includes customer confirmation did not occur due to the crash, leaving system in an inconsistent state.

Date:

c) When customer books ride, a single database transaction is initiated. The system performs these steps within the transaction:
1) Record the driver's acceptance.
2) Deduct payment from customer's wallet.
3) Prepare ride confirmation

If all steps succeed, the system issues a COMMIT, making all changes permanent & showing confirmation to the customer. If a server crash occurs before commit, upon recovery, the system uses transaction logs to ROLLBACK, undoing the wallet deduction and driver acceptance, ensuring no partial updates persist.

d) Frequent violations could lead to:
- Loss of customer trust due to payment errors or unconfirmed rides.
- Financial losses from refunds, disputes, or compensation payouts.
- Legal and regulatory issues related to transaction integrity.
- Reputational damage through negative reviews & reduced app usage.
- Operational inefficiencies and increased support costs.

Date:

Q1 2)

a) non-serial schedule is

T1: ~~r1(P1.quality)~~ r1(P1.quantity),

T2: r2(P1.quantity),

T1: W1(P1.quantity = sold_units),

T2: W2(P1.quality + returned_units)

b)

$$\begin{tikzcd}
T1 \arrow[r, "\text{w1(P1.quantity) before w2(P1.quantity)}", "\text{(write - write)}"'] & T2 \arrow[l, bend left, "\text{r2(P1.quantity) before w1(P1.quantity) (read-write)}"]
\end{tikzcd}$$

c) No, because the precedance graph contains a cycle (T1 → T2 → T1), indicating no equivalent serial order exists.

d) Corrected schedule is

T1: r1(P1.quantity),

T1: w1(P1.quantity - sold_units),

T2: r2(P1.quantity),

T2: W2(P1.quantity + returned_units)

Date: ______

e) The company could face:
- Inventory inaccuracies leading to overstocking or stockouts.
- Lost sales from items incorrectly shown as ~~unaval~~ unavailable.
- excess holding costs from overstocking.
- Financial discrepancies in accounting and reporting.
- Customer dissatisfaction due to order ~~cancellation~~ ~~or delays.~~ cancellations or delays.

Q3) $\boxed{S1}$ $\boxed{S2}$

1)

$$\begin{tikzcd} T4 \arrow[r] & T1 \arrow[dl] \\ T2 \arrow[r] & T3 \arrow[ul] \end{tikzcd}$$

$$\begin{tikzcd} T1 & T2 \arrow[l] \\ T3 \arrow[ur] & T4 \arrow[l] \end{tikzcd}$$

2) - S1 is not conflict serializable (cycle)
- S2 is conflict serializable (acyclic)

3) - S1 is not serializable.
- For S2, the serial schedule is:
T4 → T3 → T2 → T1

Date:

(Q4) 1) The problem is a lost update. Both transactions read the same initial ~~Total_Amount~~ Total_Amount (2000), computed independantly and overwrite each other's resulblt. T1's update of 1700 is lost when T2 writes 2800.

2) Final amount after both transactions = 2800

~~3) → $ T1 ⇒ T2 : 2000-300 →~~

~~T1 ⇒~~

3)
⇒ T1 → T2 : 2000 - 300 = 1700 → 1700 + 800 = 2500
⇒ T2 → T1 : 2000 + 800 = 2800 → 2800 - 300 = 2500

Both serial orders yield 2500