Question 3

# Summary of Task

A RESTful API developed using Node.js, HTML and SQLlite that handles concurrent banking transactions - Debit, Credit, Creating new account and checking balance. The task handles atomicity and concurrency control. Supports banking transactions and throws error messages when transactions are inconsistent.

# How to run?

-- 1) clone repository and cd into it

-- 2) delete banking.bd in the folder if it exists (I have included it in the repo for the sake of demonstration. A new .db file will be created locally).

-- 3) run: npm install (to download necessary dependencies)

-- 4) start server with: node Transactions.js

-- 5) open local host on browser: http://localhost:3000

# Task 2.2 Explanations

# 2.2 a)

Database schema:

- ID\_Transactions - Primary Key

- ID\_Account - Foreign Key (REFERENCES accounts(...))

- Type (debit/credit)

- Amount

- Timestamp

- Commit (bool if went through or not)

Each transaction will be linked to an accounts table with ID\_Account as primary key and other fields like Acc\_Balance etc.

# 2.2 b)

To ensure consistency:

-- Commit only when a transaction is completed.

-- Checkpointing with timestamps

-- Replicate database by storing multiple copies

# 2.2 c)

Optimizations for high performance:

-- Use big-data databases like Snowflake

-- Use meta-data management

-- Cache important data (if a user is doing multiple transactions)

-- Rate-limiting to prevent overload

-- Parallel processing and load-balancing with batch processing

-- Optimize queries with AI-API Calls

-- Secure interfaces to prevent data-leak

# Task 3 - Incorporated in Transactions.js