# CSCI 5408
# Data Management and Warehousing

# Group-04
# Project: Tiny DB

**Group Members:**
Ramya Kommalapati (B00982851)
Parth Madhvani (B00981268)
Shifa Mirza(B00981843)

**GitLab:** https://git.cs.dal.ca/mirza/csci_5408_s24_4

## Data Structure Selection:

### 1. Linear Data Structures

#### Array List

- **Purpose**: To handle query and data processing in-memory efficiently for our TinyDB project, allowing quick access, addition, and deletion of records.
- **Justification**:
    - **Efficient Data Retrieval**: ArrayLists allow for quick access to specific records using their index. This is essential for queries that require scanning through a large dataset. For instance, finding a customer within many orders is a simple index lookup.
    - **Dynamic Resizing**: As our TinyDB grows, handling increasing data volumes is essential. ArrayLists provide dynamic resizing, expanding or contracting to accommodate the changing data size.
    - **Code Integration**: Since we are working in Java, ArrayLists integrate well with our existing codebase. Built-in methods for adding, accessing and removing elements simplify development.

#### LinkedList

- **Purpose**: To support operations where frequent insertions and deletions are required, ensuring efficient data modification for our TinyDB project.

- **Justification**:
    - **Efficient Modifications**: LinkedLists excel when frequent insertions and deletions are needed. If we need to constantly add or remove records from the middle of a list, a LinkedList becomes advantageous. Its insertion and deletion operations are efficient for such scenarios.
    - **Specialized Use Cases**: While ArrayLists are useful, LinkedLists become valuable in specific situations. Managing a list of active transactions or temporary data buffers is a good example where a LinkedList's strengths are beneficial.

## 2. Custom File Format

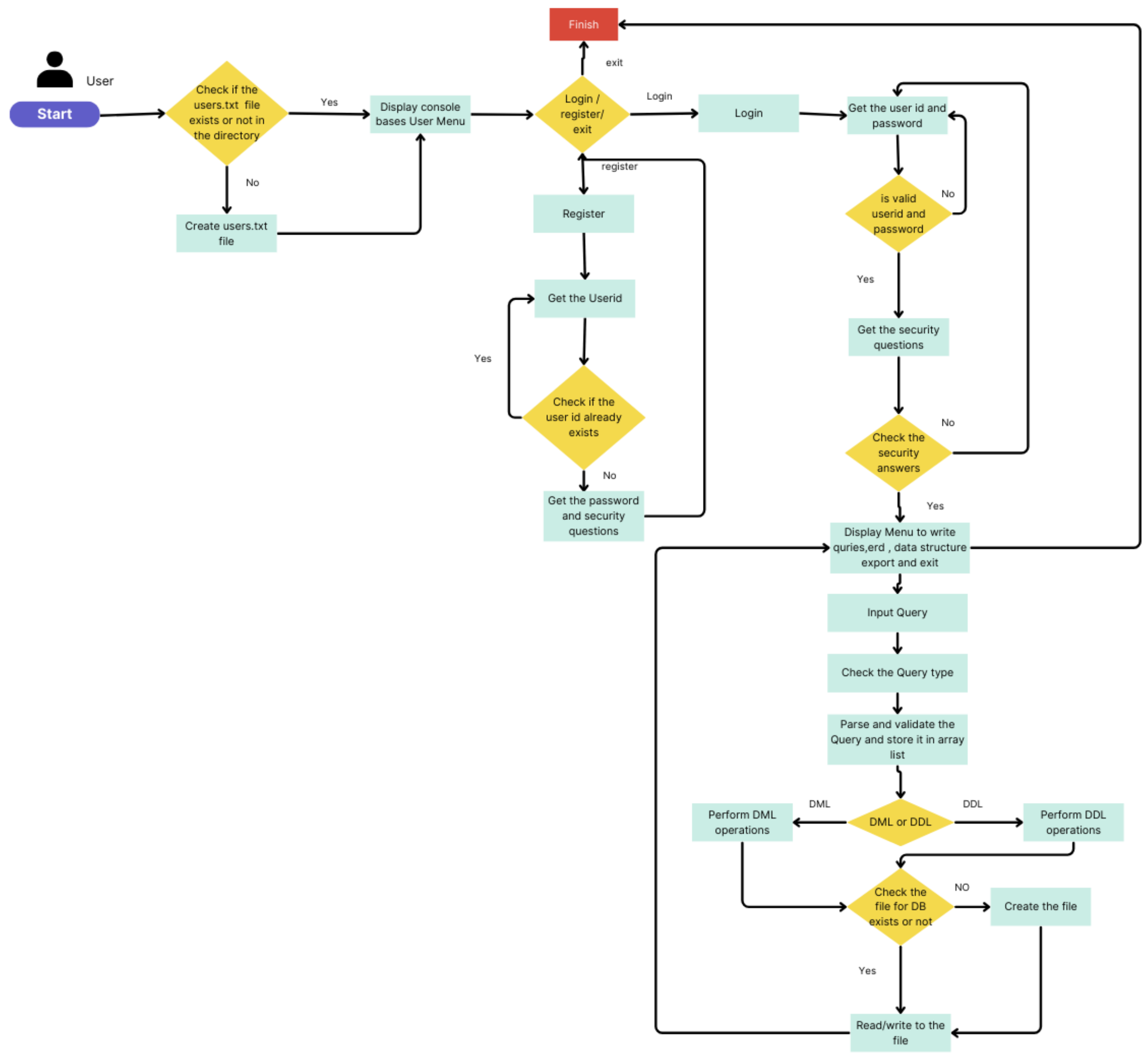
**Custom Format:** Pipe-separated Text File (.txt)

**Purpose**: We've opted for a custom file format to ensure data integrity and easy retrieval. It's a pipe-separated text file, using "###" as a unique delimiter to separate each field within a record.

**Justification**:

- **Clear Delineation of Fields**: Our custom format uses the "###" delimiter to clearly distinguish between data fields. This simplifies parsing the data and maintains its integrity. Parsing becomes straightforward, and data remains consistent and reliable.
- **Simplicity**: Our custom format prioritizes ease of use, both for reading and writing data programmatically. It fulfills TinyDB's persistence needs without introducing unnecessary complexity that might come with standard formats like JSON, XML, CSV, or serialized/binary formats.

## Workflow diagram:



Finish
User
Start
Check if the users.txt file exists or not in the directory
Yes
No
Create users.txt file
Display console bases User Menu
Login / register/ exit
exit
Login
register
Login
Get the user id and password
is valid userid and password
No
Yes
Get the security questions
Check the security answers
No
Yes
Register
Get the Userid
Yes
Check if the user id already exists
No
Get the password and security questions
Display Menu to write quries,erd , data structure export and exit
Input Query
Check the Query type
Parse and validate the Query and store it in array list
DML or DDL
DML
DDL
Perform DML operations
Perform DDL operations
Check the file for DB exists or not
NO
Create the file
Yes
Read/write to the file

*Fig 1: Workflow diagram of Tiny DB*