**CT – Optimizing a Social Media Content Recommendation System with Hash Tables**

Dwight Abrahams

CSU Global

CSC506-1: Principles of Software Development

Professor Pubali Banerjee

March 16, 2025

**Optimizing a Social Media Content Recommendation System with Hash Tables**

## **Introduction**

The rapid growth of social media platforms necessitates efficient and scalable data structures for content recommendation. This project explores how a hash table can be utilized to store user preferences and deliver real-time recommendations. By relying on an efficient hashing mechanism, the system achieves constant average-time lookups, which is crucial for platforms handling vast amounts of data with low latency requirements (Cormen et al., 2009).

## **Content**

The primary challenge in developing a real-time recommendation system is managing collisions in the hash table when numerous users share similar preference profiles. Our implementation uses chaining to handle collisions, ensuring each hash index can store multiple key-value pairs. Despite the theoretical average time complexity of O(1) for insertion and retrieval, real-life factors such as uneven data distribution, resizing overhead, and hash function quality may degrade performance to O(*n*) in worst-case scenarios. External factors like network latency, distributed computing overhead, and the size of user traffic spikes can further influence the efficiency lower bound when delivering content recommendations (Knuth, 1998).

A key implementation in this program is how collisions are showcased. Rather than printing the entire hash table, the user can pick a particular user ID and check which bucket it occupies. If the bucket contains more than one (key, value) pair, it indicates a collision. Reducing the hash table size (e.g., size of 2 or 3) while simultaneously simulating several users almost guarantees collisions, illustrating chaining in action (Knuth, 1998).

The solution also includes an interactive menu:

1. **Generating Recommendations** – Users can query the hash table for recommended content preferences; if no preferences exist, a suitable message is returned.
2. **Checking Bucket Info** – By requesting the bucket for a specific user, the program identifies the corresponding hash index and prints all entries in that bucket, revealing collisions if more than one user shares it.

## **Skills Acquired and Obstacles**

During development, mastery over hashing techniques and collision resolution was gained. Ensuring the hash function evenly distributes user keys significantly reduced collisions and improved overall throughput. Additionally, integrating the hash table into a recommendation pipeline required understanding user modeling, content-based filtering, and data streaming. A key obstacle was balancing memory usage with performance—larger tables reduce collision probability but consume more memory. This design improves user engagement by allowing them to see how a particular key shares a bucket with others, clarifying the benefits of chaining.

## **Conclusion**

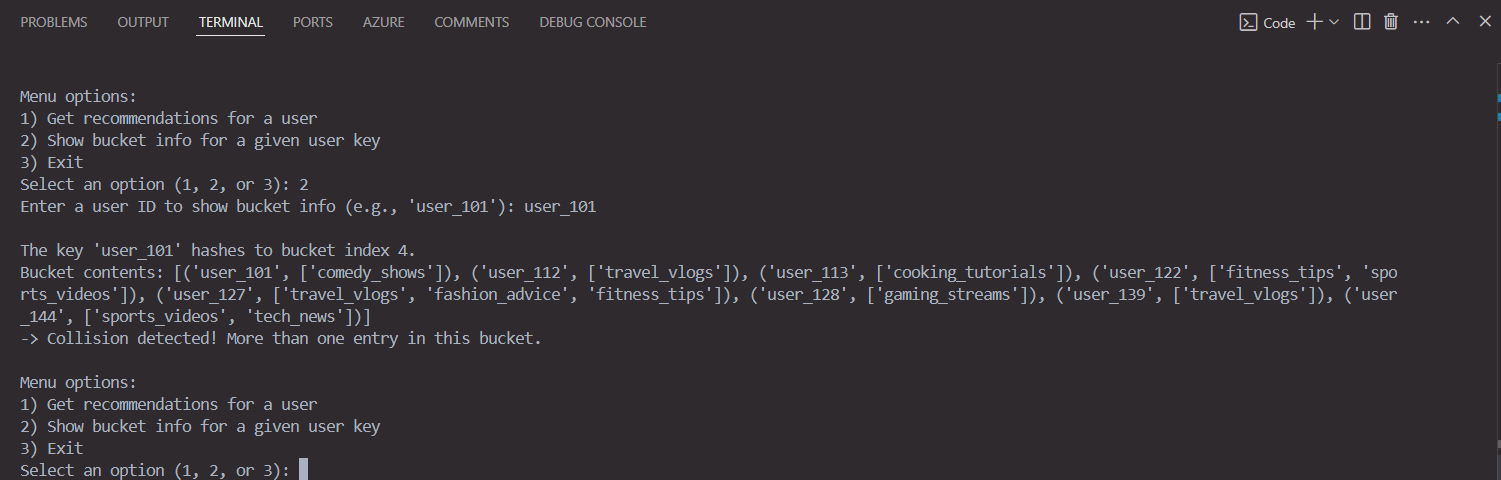
Hash tables, when properly implemented, offer substantial advantages for social media recommendation systems. Despite real-world constraints such as uneven data distribution and hardware limits, hashing continues to be a cornerstone of scalable, high-speed data retrieval. By adopting robust hashing strategies, platforms can enhance user satisfaction through timely and accurate content recommendations. By letting end-users confirm collisions directly, the program provides deeper insight into how data structures scale under load and fosters a better grasp of hashing strategies.

# **References**

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). The MIT Press.

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching* (2nd ed.). Addison-Wesley.

# Appendix A



*Figure 1. Program execution showcasing successful data retrieval, bucket contents, and collision if it occurs.*