# Mini E-Commerce Backend – Report

Student: Your Name – CMSID: Your CMSID

Date: 15-Aug-2025

## Section 1: System Overview

This project implements a simplified backend system for an online shopping application.  
It contains:  
1. Product Catalog – implemented using a custom hash table (separate chaining with linked lists) to store and manage products efficiently.  
2. Order Management System – implemented using a custom FIFO queue (linked list) to manage orders in the order they arrive.  
  
The main API is exposed via the ECommerceSystem class, which provides:  
- addProduct(...)  
- findProduct(...)  
- createOrder(...)  
- shipNextOrder()  
- restockProduct(...)  
- listPendingOrders()

## Section 2: Product Catalog Analysis

Why a Hash Table?  
A hash table provides average O(1) lookup, insertion, and update times, which is ideal for frequent product searches and updates in an e-commerce system. Separate chaining handles collisions efficiently and allows dynamic growth.  
  
Time Complexity:  
- findProduct(...) → Average O(1), Worst O(n) if all keys fall into one bucket.  
- restockProduct(...) → Average O(1), Worst O(n) for the same reason.

## Section 3: Order Management Analysis

3.1 Chosen Structure  
A FIFO Queue implemented using a singly linked list.  
  
3.2 Justification  
Orders must be shipped in the same sequence they were placed. FIFO ensures the first placed order is processed first.  
- enqueue → createOrder(...)  
- dequeue → shipNextOrder()  
  
Both operations run in O(1) time with a linked list.  
  
3.3 Comparison with Alternatives  
Alternative: Stack (LIFO)  
Using a stack would process the most recent order first, delaying older orders indefinitely. This is unfair for customers who ordered earlier.

## Section 4: Time Complexity Table

|  |  |  |
| --- | --- | --- |
| Function | Avg Time Complexity | Justification |
| addProduct | O(1) | Hash table insertion |
| findProduct | O(1) | Hash table lookup |
| createOrder | O(k) | k = number of items to check & reduce stock |
| shipNextOrder | O(1) | Queue dequeue |
| restockProduct | O(1) | Hash table lookup + update |

## Section 5: Reflection

Challenges:  
Implementing custom hash table and queue without using Java’s built-in collections required careful handling of collisions, memory management, and pointer manipulation in linked lists.  
  
What I Learned:  
I gained practical experience in designing systems with appropriate data structures for specific requirements. Implementing the data structures manually deepened my understanding of their working, complexities, and trade-offs.