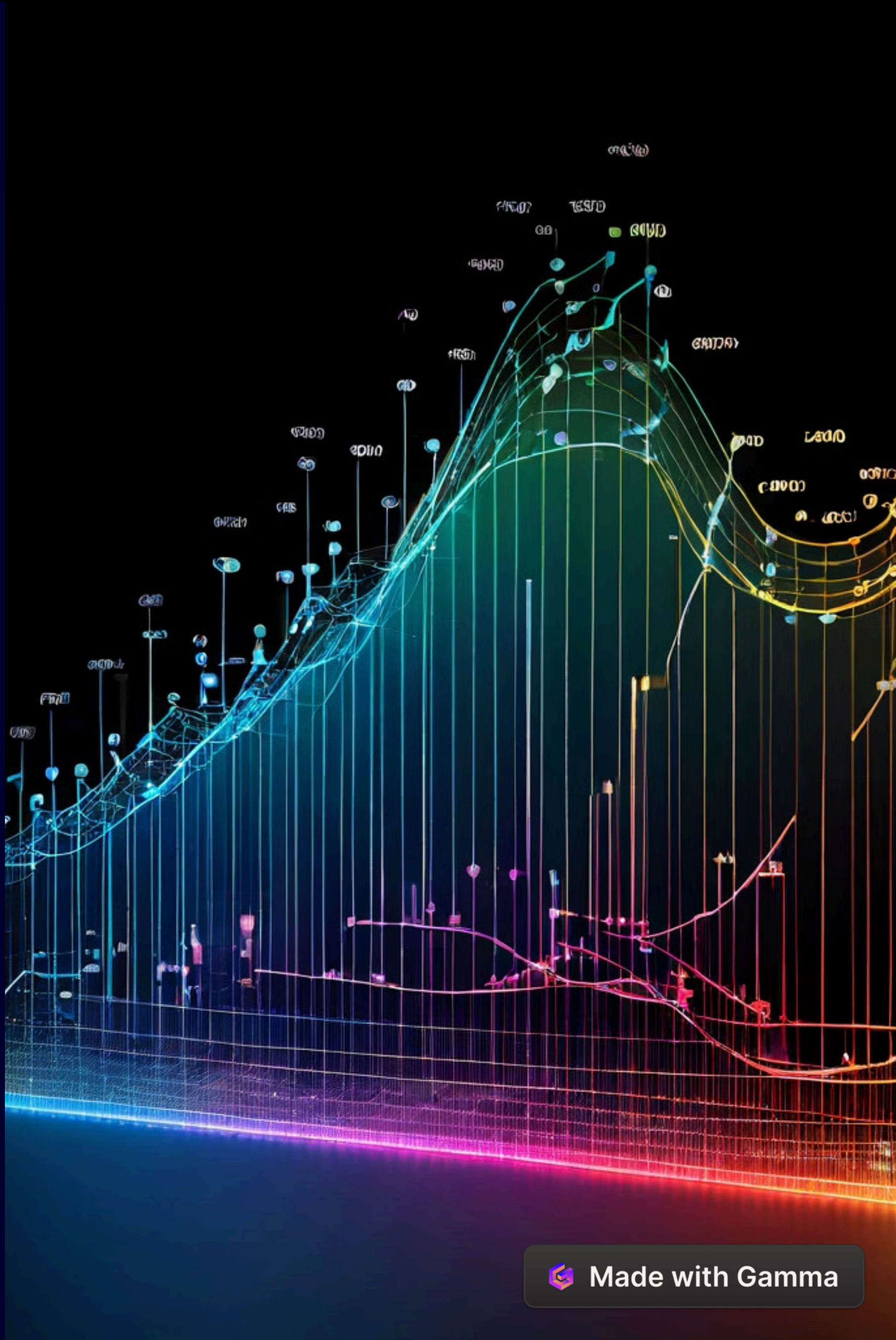
E-Commerce Inventory Management System!



Project Overview and Objectives

objective

This project is designed for a multi-user platform where **Admins**, **Sellers**, and **Buyers** interact with the system. The system is implemented in C++ using **unordered maps** for efficient storage and retrieval of data, along with **stacks** to manage seller communication messages.

Goal

The goal of this project is to create an **E-Commerce Inventory Management System** that efficiently manages stores, products, and user interactions using **unordered maps** and **stacks** in C++. It aims to showcase real-world applications of data structures in a user-friendly, multi-user platform.

Key Features and Functionalities

Admin Functions

- Add, view, and delete stores with detailed information.
- Send messages to sellers using a **stack** for LIFO-based message retrieval.

Seller Functions

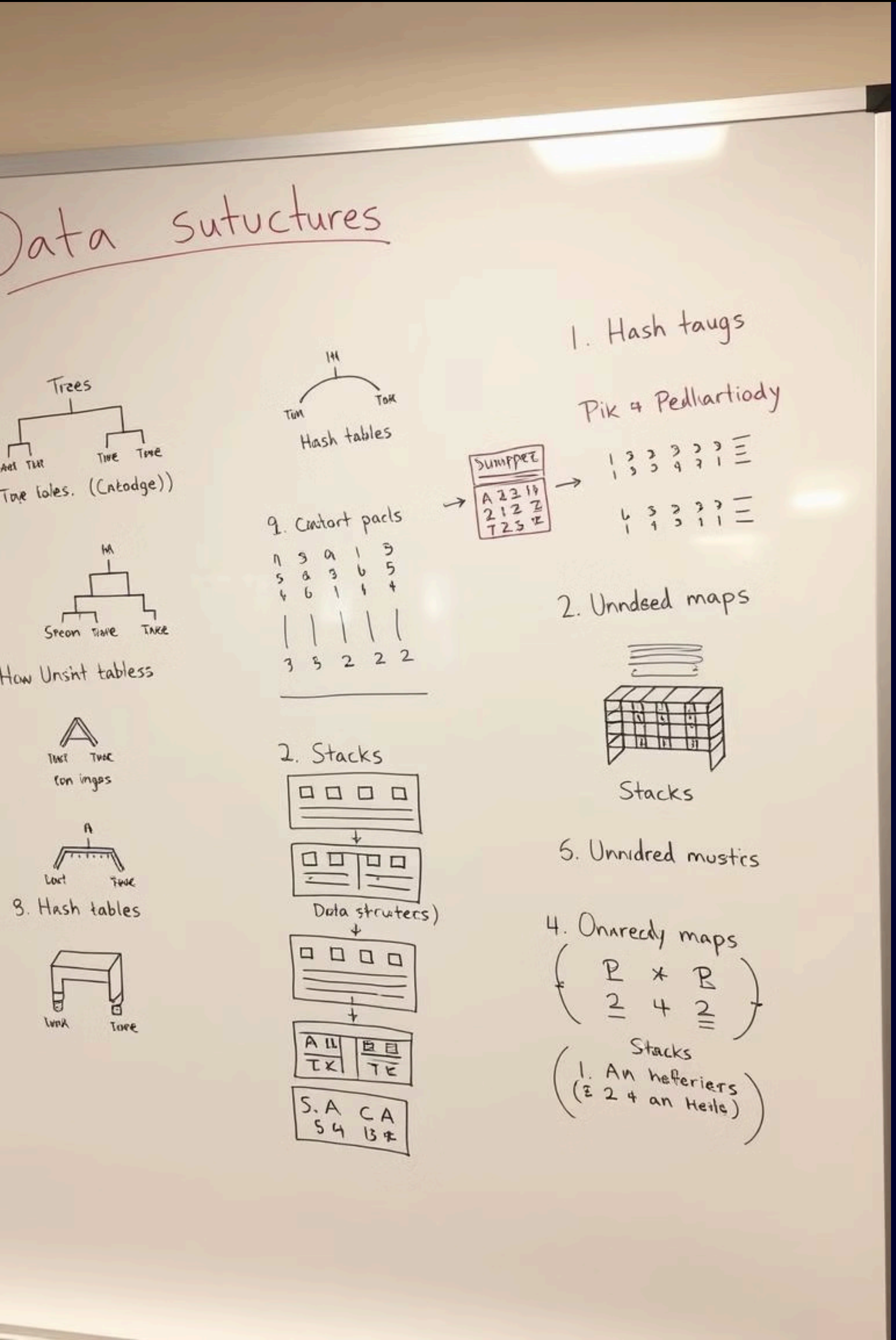
- Register and log in to stores securely.
- Manage inventory by adding, updating, or deleting products.
- View sales performance and total earnings.
- Access messages from admins stored in a **stack**..

Buyer Functions

- Register, log in, and manage account details.
- Browse and purchase products from available stores.
- Rate stores based on shopping experience.
- Check account balance and transaction history.

Core Features

- Unordered Maps: Fast and efficient storage and retrieval of buyer and store data using CNIC as a key.
- Stacks: Effective handling of admin-to-seller communication for prioritized message delivery.
- Used 3 Files Structure HeaderFile.h



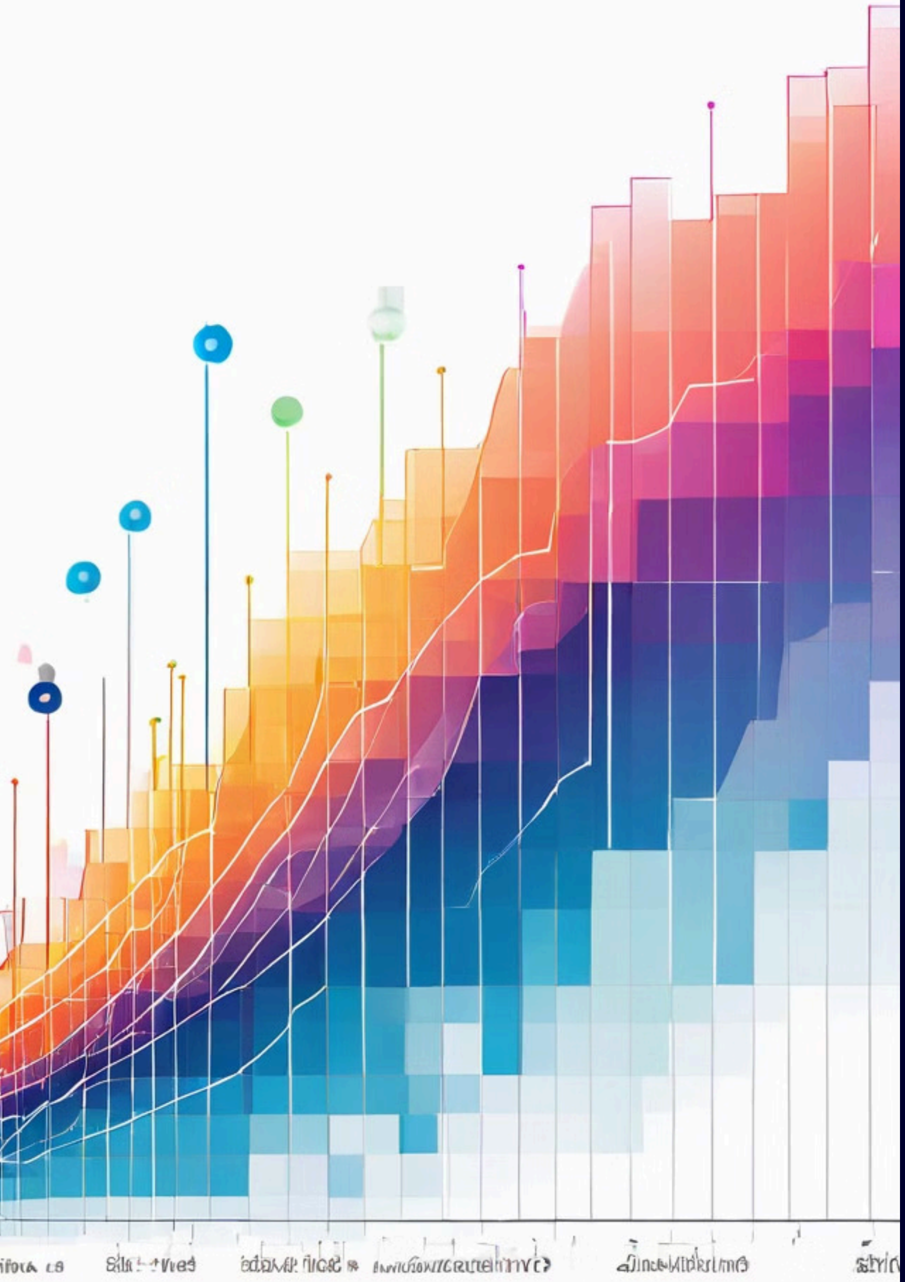
Data Structures Utilized

1 **Hash Tables / unordered_map / map**

Efficiently searching and retrieving product data based on product IDs.

2 **Stack**

Admin Sending Messages to different stores efficiently most important appear first.



Time Complexity Analysis

1. **Adding, Removing Stores Products**

   Hash tables offer efficient deletion with O(1) average time complexity.

2. **Updating Deleting Stores Products**

   Hash tables offer efficient updating and deletion with O(1) average time complexity.

3. **Searching Products Stores**

   Hash tables offer efficient searching with O(1) average time complexity.

4. **Push and Display Messages using Stack**

   Stack has time complexity of O(1) for insertion deletion.

System Architecture and Integration

Admin Module

Handles store registration, deletion, and viewing.

Manages communication with sellers via a **stack** for storing and retrieving messages in LIFO order.

Seller Module

Provides features for store registration, inventory management, and sales tracking.

Interacts with the stack to read admin messages and responds accordingly.

Buyer Module

Manages buyer registration, login, and account features.

Enables product browsing, purchasing, and store rating.

Data Management

Unordered Maps: Centralized storage for buyers and stores, ensuring fast data retrieval using CNIC as a unique key.

Integration

The modules are interconnected via shared data structures, ensuring smooth interaction between users.

3 File Structure Approach

I have used a 3 files structure approach to divide my code in separate files for easy reading and understanding.

Any Questions?

Thank you for your time. I'm happy to answer any questions you might have.