Hi everyone,

In this presentation, we will be talking about the importance of Data Structures and Algorithms in real-life applications. In today’s world, where performance and scalability are integral part of an application, understanding data structures and algorithms is crucial in creating software solutions that can provide maximum performance and memory efficiency. Although we now have hardware’s that has higher configurations such as higher RAM, or memory, the efficiency of applications greatly depends on the data structures used and the algorithms used to perform the queries.

Software may have the best functionality, however if it does not deliver performance, the end users will not prefer using the software. A crucial but often overlooked aspect of data structures and algorithms is their dependence on system hardware. Hardware constraints such as CPU architecture, memory hierarchy, and storage medium play a vital role in determining the efficiency of an algorithm. For example, if we visit e-commerce websites such as amazon, if the search takes a lot of time, most users will not prefer using the application. However, as amazon does have millions of records or inventories, it still delivers higher performance with extremely big data sets and with millions of people accessing the website or application at the same time. So, the success of an application does lie in the functionality or the problem the software solves as well as its responsiveness and performance.

Let’s dive into the abstract project. The main goal of this project is to understand the data structure and algorithms that can be used in creating enterprise grade applications. To understand the different data structures, we analyze the trade-offs between the data structures in Python such as Dictionary, Lists, Tuples and compare the time complexity of each data structure. By carefully analyzing their tradeoffs in performance, flexibility, and security, suitable data structures were selected in creating a dynamic inventory management system using Python as the coding language. In addition, to creating the inventory management system, the project also discusses the problems encountered while using these data structures, and how it solved these hindrances. Additionally, it also focuses on how the software can be scaled to handle large data sets and multiple users accessing it at the same time by using concepts such as multi-threading and cache. Finally, it outlines the steps that can be taken to take this project to the next level.

To begin, let us first try to understand what an Inventory Management System is, and the problem it solves. Inventory management is a critical component of any business that deals with physical goods. Whether it's a retail store, a manufacturing plant, or an e-commerce platform, keeping track of inventory efficiently ensures smooth operations, cost savings, and customer satisfaction. An Inventory Management System (IMS) helps businesses monitor stock levels, track orders, optimize supply chains, and prevent losses due to overstocking or stockouts. It is also important to provide essential data and graphs to businesses such as history of a product price, the price trends, the demand trends of products, the customers buying history and much more which the inventory management system can provide. A good inventory management system also keeps audit of the user activities, such that it can trace back user actions when needed.

To create an effective inventory management system, the project analyzes the tradeoffs between the data structures in Python. The project makes use of Dictionaries, Lists, Tuples and provides reasons why these were selected over other data structures. To achieve better performance, the time and space complexity of these data structures were examined.

We examined the time complexity in the general case for insertion, deletion, and lookups. As we can see, the time complexity of lists is O(n) on average and dictionaries is constant time of O(1), dictionary was chosen over lists, arrays, queues, heaps as it provides better performance. Arrays and lists are also suitable, however to search in array and lists, it may need to search through the entire array or list to find the element, however in dictionaries, it takes a constant time, as each value is represented by a key, and a key points to a specific memory location, boosting the search performance. In addition, the value that a dictionary store does not need to be of same type, which provides higher flexibility if in case in future, the fields needs to be changed, and we do not have to worry about data migration. Arrays however need the data types to be of same type, suppose an array of strings, an array of integers. But this limitation is not present in dictionaries. This makes dictionaries ideal to store the products and categories that the inventory management system keeps track of and performs CRUD (create, retrieve, update, delete) operations on. Dictionaries allow the values to be accessed using keys, and they are an ideal data structure to store, retrieve, and update data using unique keys. The dictionary in Python also has tons of built-in methods, such as get, contains, pop, and keys, which makes the execution of tasks easier.  
 For tracking the price history, we used Tuple as tuples are immutable which means their value cannot be altered once inserted. This is ideal for history, it is highly of security reasons that no user can go back and change the price. This helps maintain data integrity.

The program has three main classes: Inventory, Category, and Product. The inventory class sits at the top of the hierarchy, which consists of multiple categories and products.

The inventory class uses dictionaries to store categories and products. A dictionary is a collection where each value is associated with a unique key (Beats, 2023). Dictionaries can have only one key per value, so the use of dictionaries also helps prevent redundancies and allows uniqueness. As dictionaries are mutable, the records that have these data types can be modified after creation (Python Data Structure”, 2024).

The product class uses tuples to store the price history of the product. Tuples in Python are immutable collections, which means that once created, they cannot be modified (“The power of tuples,” 2024).

The Python dynamic inventory management program was optimized using various techniques to achieve faster execution times and optimal memory usage. It was designed to handle large datasets in cases where the application may be scaled.

**Demo – Walk through the code. Explain functionalities and run the test cases. Also talk about cache implementation.**

After demo, return back to the presentation, and talk about areas of improvement.

In conclusion, the data structures utilized in developing the inventory management system provide an optimal time complexity of O (1) for basic CRUD operations concerning inventory categories and products. LRU Caching has been implemented to enhance performance. While testing among twenty thousand categories and a hundred thousand products, the time 15 complexity to search the record using factors such as name, price range, and ID was optimal, below 0.3 seconds. The test cases also covers the accuracy of the functionalities implemented.

The Python inventory program currently uses only hardware resources for storage and cache. However, practical, real-life applications must be web-based so users can access them without running the program on their local machines. The fundamentals of the inventory program analyze many data structures and use the best data structures like dictionaries, tuples, and lists to achieve optimal time and space complexity. The next step to scale this application is to integrate it with database management systems like MySQL and MongoDB and have a user interface such that users can interact with the system. Angular, Vue.js, and React are some of the most commonly used frontend frameworks that can be used to develop the user interface (Mendes & Rodrigues, 2024).

In addition to the Python program, REST APIs will need to be implemented so that the frontend application can perform the required functions. Some basic examples of APIs are the POST endpoint, which creates categories and products; the PUT endpoint, which updates them; and the DELETE endpoint, which deletes a category and product.

Redis, Apache Ignite, and Hazelcast are some of the commonly used in-memory caching frameworks that can improve the performance of web-based applications (“Top 10 in-memory caching frameworks for web application development in 2023,” 2023). Multithreading can effectively improve performance if functions need to be performed in batches. As multithreading allows the execution of parallel operations, if the operations are independent of one another, it can enhance the performance by a significant factor (“Benefits of multithreading,” n.d.). 16

Finally, to maintain the code standards and check if all the functions are performing well, test cases must be written and updated whenever code changes occur. Regression testing, stress testing, and end-end testing must be done to ensure that the program is functioning well and delivers optimal performance.