Optimizing Inventory Management Systems for Enhanced Operational Efficiency

Rekha J E
Post Graduate Student
tocontactrekha@gmail.com
Dept of MCA, DSCE
Bangalore

Dr . Vibha M B
Assistant Professor
vibhamcavtu@dayanandasagar.edu
Dept of MCA, DSCE
Bangalore

***Abstract*--Many corporate enterprises have faced difficulties in managing their inventory, leading to a need for understanding and educating management on effective inventory usage. The primary objective of this study is to analyse how inventory management systems affect the operational performance of industrial companies. The findings reveal a strong correlation between inadequate inventory management systems and poor organizational performance. Based on the research titled "Optimizing Inventory Management Systems for Enhanced Operational Efficiency," it is evident that subpar inventory control can detrimentally affect an organization's productivity and financial success. It is crucial for businesses to train their employees in inventory control and management. Overall, the study's conclusions highlight the numerous benefits that businesses can gain from implementing an effective inventory control and management system, such as cost reductions, increased profitability, waste reduction, transparency, accountability, and streamlined stock storage and retrieval.**

I. INTRODUCTION

Inventory management is a critical aspect of any business, determining the fine balance between maintaining sufficient stock levels to meet customer demand while avoiding excessive carrying costs. Traditional inventory management models, such as the Economic Order Quantity (EOQ) model, have been the go-to approach for decades. However, with the advent of cutting-edge technologies, businesses now have the opportunity to revolutionize their inventory management systems, enhancing efficiency and accuracy. In this context, developing a ReactJS application to optimize inventory management becomes an exciting prospect. By developing a ReactJS application for inventory management, businesses can harness the power of a cutting-edge, user-friendly, and highly interactive front-end technology. ReactJS allows for the creation of responsive and intuitive user interfaces, enabling inventory managers to visualize and control inventory data more effectively. The ReactJS application can integrate with backend systems and APIs to retrieve and process real-time inventory data, ensuring up-to-date insights for decision-making. In this comparative analysis, we will look at the benefits and drawbacks of the current EOQ model and contrast it with the inventory management features provided by the ReactJS application. examining how the ReactJS app may perform better than the conventional EOQ model in terms of precision, efficiency.

II . LITERATURE REVIEW

Inventory management is a significant concern for organizations, and it involves various steps taken by managers that incur costs, especially when dealing with perishable stock (Taygi, 2014)[6]. Successful businesses have attributed their achievements to improved inventory management systems, while those that solely focused on web development faced challenges (Tam, 2003)[7]. Retailers consider managing inventory crucial for achieving higher turnover and implementing just-in-time delivery practices (Lankford, 2004; Schniederjans et al., 2013)[8]. A case study conducted by Lal (1981)[9] examined inventory management at Modi Steels Limited and introduced a model that incorporated price variables, previously neglected by the company, to improve working capital management. The study recommended comprehensive policies considering internal and external factors to enhance efficiency. Furthermore, Madishetti and Kibona (2013)[10] found that well-designed and executed inventory management positively influenced the profitability of small and medium-sized enterprises (SMEs). Their research, based on a sample of 26 Tanzanian SMEs, demonstrated a significant negative linear relationship between the inventory conversion period and profitability.

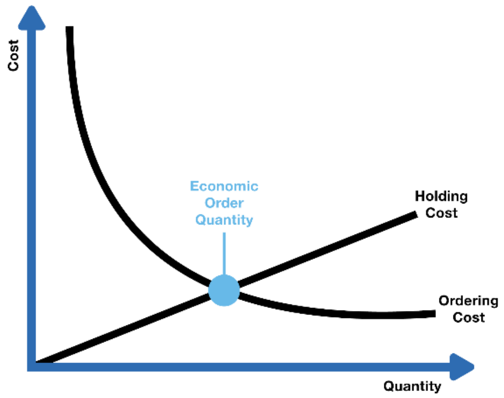
In the fast-changing market environment with fluctuating customer demands, efficient logistical processes and inventory management are essential for achieving competitiveness (Kovács & Kot, 2016)[11]. As a global operations strategy, inventory management, in conjunction with supply chain management (SCM), aims to provide flexibility and responsiveness. Inventory optimization involves ensuring the availability of the right product, in the right place, at the right time, in the right quantity, and of the right quality to meet demand and supply (Armenzoni et al., 2015) [12]. To achieve this, classifying inventory and utilizing demand forecasting models for accurate predictions play crucial roles in managing inventory efficiently.

III. IMPLIMENTATION OF INVENTORY MANAGEMNT

Here we are implementing inventory management through the eoq model and reactjs, where we are showing that the eoq model fails to show time efficiency, so for that, we are implementing the reactjs application to achieve time efficiency.

A. Existing EOQ model for Inventory management.

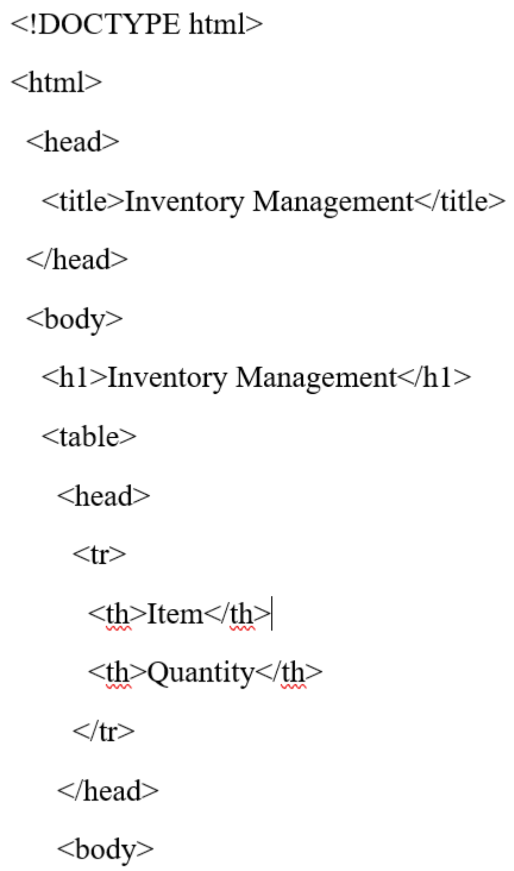
The Economic Order Quantity (EOQ) model is an inventory management strategy that determines the optimal order quantity to minimize costs by balancing holding costs and ordering costs. It assists businesses in achieving the ideal balance between carrying excessive inventory and placing orders too frequently.



`

Now, let's consider an example where the Economic Order Quantity (EOQ) model fails to achieve time efficiency due to its limitations. EOQ is a traditional inventory management model that calculates the optimal order quantity to minimize costs, but it doesn't directly address rendering efficiency.

Suppose we have an inventory management system built without using ReactJS or any frontend framework. The inventory list is displayed as a static table where all items are rendered at once.



```
<!DOCTYPE html>
<html>
  <head>
    <title>Inventory Management</title>
  </head>
  <body>
    <h1>Inventory Management</h1>
    <table>
      <head>
        <tr>
          <th>Item</th>
          <th>Quantity</th>
        </tr>
      </head>
      <body>
```

<!-- Loop through inventoryItems and render each row -->

```
          <tr>
            <td>{{ item.name }}</td>
            <td>{{ item.quantity }}</td>
           </tr>
         </body>
       </table>
     </body>
   </html>
```

Fig 1: Here is a small snippet of code for how the EoQ model is implemented in normal HTML code.

In this case, as the inventory grows in size, rendering all items at once can lead to slow initial loading times and poor performance. There's no optimization for virtualization or memorization, and the entire inventory list is rendered each time a page loads.

```
+------------------------------+
|                              |
|       Inventory Table        |
|                              |
+------------------------------+
|                              |
|
```

Fig 2: Illustrate the Inventory table for the above code.

```
Time (minutes)
^
|    +-----------------+
|    |                 |
|    |   EOQ Model  |
|    |                 |
|    +-----------------+
|    |                 |
|    |                 |
|    | Order           |
|    | Placement       |
|    |                 |
|    +--------------+
|    |                 |
|    |                 |
|    |                 |
|    |                 |
|------|--------------|------------------------->
          Initial        5 minutes
```

Fig 3: The graph that illustrates the timeline for an EOQ model taking 5 minutes to place an order

In the graph, the timeline starts at the initial state and progresses to the right. At time 0, the EOQ model initiates the order placement process. The order placement takes 5 minutes to complete. Once the 5-minute mark is reached, the order is considered placed, and subsequent processes, such as s, can be carried out.

B. Drawback 0f EOQ model

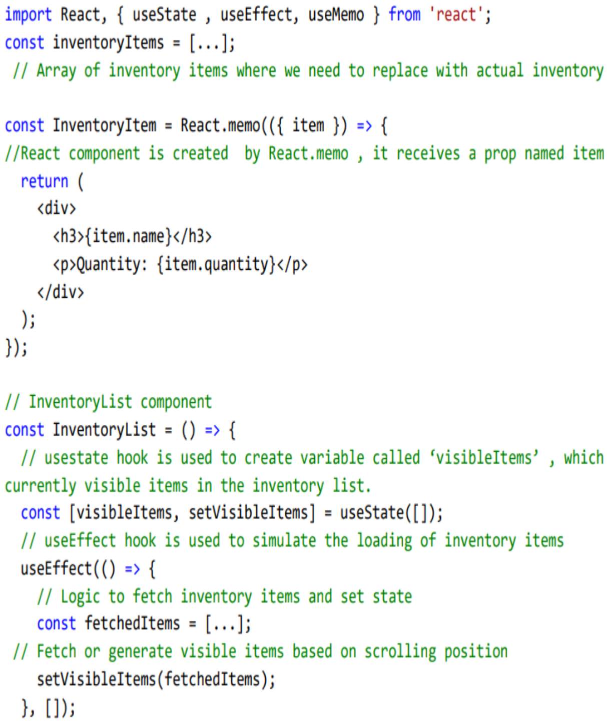
Here's a scenario where the EOQ model may face challenges in achieving time efficiency

Procedure for Manual Ordering: Manual estimates are frequently used in the EOQ model to determine the optimal request quantity. Once the order amount has been decided, it must be manually communicated to the procurement division or the supplier. This manual process can be time-consuming and prone to errors, especially when there are several items or subsequent request positions.Request Handling Time: In the EOQ model, the supplier or procurement division may need some time to process and complete the order once it has been placed. Numerous strategies, such as request confirmation, choosing and pressing, and shipment schemes, can be used in this cycle. The general response time to requests may change depending on how well the inventory network and acquisition system function.

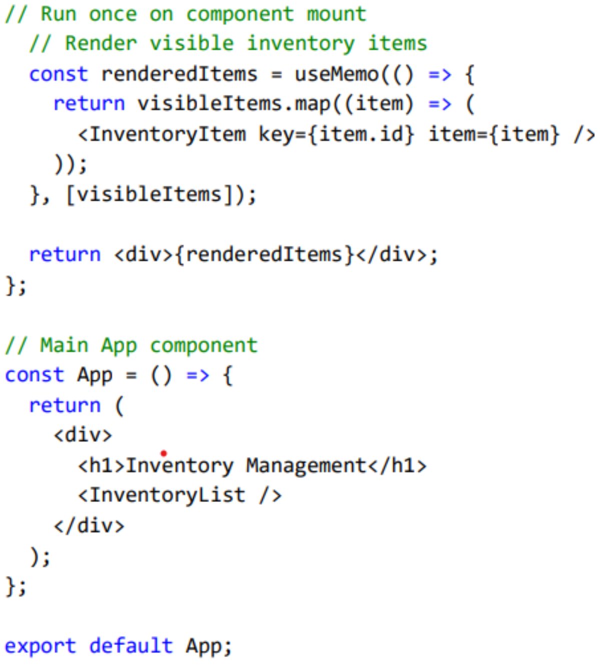
C. IMPLEMENTING REACTJS ON INVENTORY MANAGEMNT

ReactJS, which demonstrates the time efficiency achieved through optimization techniques. We'll use a simplified inventory list with memorized components and virtualized rendering.

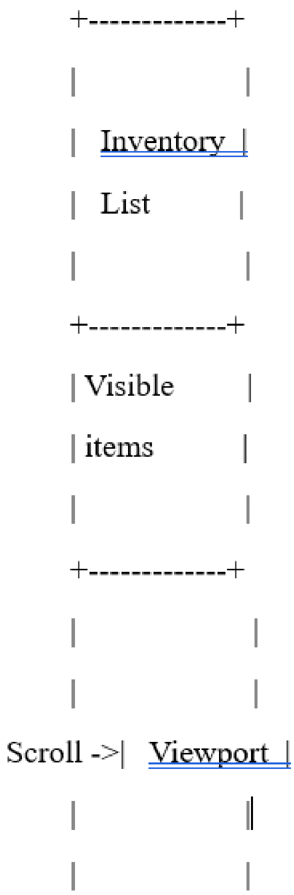
1. Below is a pseudocode example showcasing the implementation of a ReactJS application for inventory management:



```
import React, { useState , useEffect, useMemo } from 'react';
const inventoryItems = [...];
 // Array of inventory items where we need to replace with actual inventory

const InventoryItem = React.memo(({ item }) => {
//React component is created  by React.memo , it receives a prop named item
  return (
    <div>
      <h3>{item.name}</h3>
      <p>Quantity: {item.quantity}</p>
    </div>
  );
});

// InventoryList component
const InventoryList = () => {
  // usestate hook is used to create variable called ‘visibleItems’ , which
currently visible items in the inventory list.
  const [visibleItems, setVisibleItems] = useState([]);
  // useEffect hook is used to simulate the loading of inventory items
  useEffect(() => {
    // Logic to fetch inventory items and set state
    const fetchedItems = [...];
 // Fetch or generate visible items based on scrolling position
    setVisibleItems(fetchedItems);
  }, []);
```



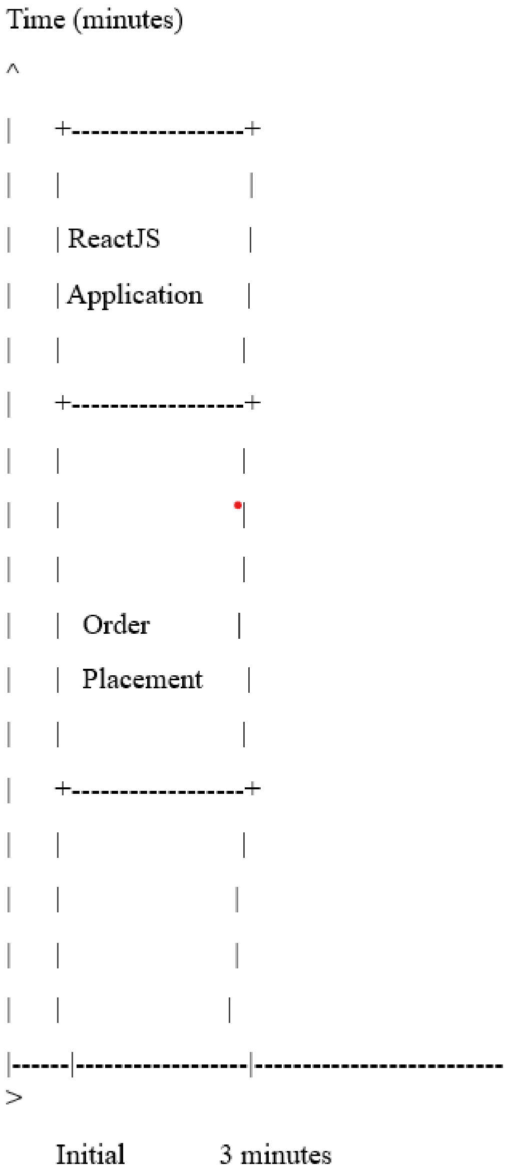
```
// Run once on component mount
  // Render visible inventory items
  const renderedItems = useMemo(() => {
    return visibleItems.map((item) => (
      <InventoryItem key={item.id} item={item} />
    ));
  }, [visibleItems]);

  return <div>{renderedItems}</div>;
};

// Main App component
const App = () => {
  return (
    <div>
      <h1>Inventory Management</h1>
      <InventoryList />
    </div>
  );
};

export default App;
```



```
        +-------------+
        |             |
        |  Inventory  |
        |  List       |
        |             |
        +-------------+
        | Visible     |
        | items       |
        |             |
        +-------------+
        |             |
        |             |
Scroll ->|  Viewport  |
        |             |
        |             |
```

Fig.4: Graph illustrates the above Reactjs code.

Here's example of the graph that illustrates the timeline for a ReactJS application taking 3 minutes to place an order:



```
Time (minutes)
^
|    +------------------+
|    |                  |
|    | ReactJS          |
|    | Application      |
|    |                  |
|    +------------------+
|    |                  |
|    |                  |
|    |                  |
|    |   Order          |
|    |   Placement      |
|    |                  |
|    +------------------+
|    |                  |
|    |                  |
|    |                  |
|    |                  |
|------|------------------|--------------------------
>
      Initial          3 minutes
```

Fig 5: A graph that illustrates the timeline for a ReactJS application model taking 3 minutes to place an order.

In the graph, the timeline starts at the initial state and progresses to the right. At time 0, the ReactJS application begins the order placement process. It takes 3 minutes for the ReactJS application to complete the order placement. After the 3-minute mark, the order is considered placed, and further processes, such as communication with external systems or backend processing, may begin.

IV . Conclusion

The speed at which the ReactJS application refreshes the inventory is determined by the implementation speed and the complexity of the operations involved. Unlike manual procedures, ReactJS is specifically designed to optimize rendering and swiftly update the user interface as changes occur. By utilizing techniques like virtualized rendering and memorization, ReactJS enhances performance and reduces unnecessary re-renders.When there are changes in the inventory, ReactJS updates the relevant components and renders the new data accordingly. The update process in the ReactJS application generally happens more quickly compared to manual procedures. The actual time required for inventory updates in the ReactJS application can vary depending on factors such as the inventory quantity, UI complexity, and system efficiency. It can range from a few milliseconds to a few seconds. overall, the ReactJS application excels at updating the inventory promptly due to its efficient rendering, virtualization, and memorization techniques. This results in faster inventory updates compared to the EOQ model.

V. FUTURE SCOPE

In the future, there is potential to extend the inventory management application to mobile platforms by developing a native or hybrid mobile app using React Native. This would allow for the management of inventory on mobile devices, granting freedom and flexibility to field workers and warehouse employees. To streamline inventory management processes, it is advisable to integrate barcode or QR code scanning functionality into the ReactJS application. This integration would simplify item identification, tracking, and updates, resulting in rapid and accurate inventory management.

VI.REFERENCES

1. Salawati, R., Tinggi, M., & Kadri, M. (2012). Inventory management: A study on manufacturing companies in Kuching Samarahan Division Sarawak. Procedia Economics and Finance, 2, 133-139.

2. Nyabwanga, R. N., & Ojera, P. B. (2012). Effects of inventory management on organizational performance: A case study of Kenya Power and Lighting Company Ltd. International Journal of Social Sciences and Entrepreneurship, 1(7), 342-360.

3. Dimitrios, M. (2008). The importance of effective inventory management. International Journal of Economics, Commerce and Management, 6(2), 1-10.

4. Ghosh, S., & Kumar, V. (2003). A model for inventory control of a system with one supplier and multiple retailers. European Journal of Operational Research, 149(3), 536-548.

5. Barik, S. (2016). Performance evaluation of inventory management systems: A review. International Journal of Emerging Research in Management &Technology, 5(6), 14-19.

6. Taygi, V. K. (2014). Optimization of inventory model with deteriorating items. International Journal of Research in Engineering and Technology, 3(1), 495-502.

7. Tam, K. Y. (2003). Web-based business-to-business trading: An innovative approach to building supply chain partnerships. Journal of Organizational Computing and Electronic Commerce, 13(1), 13-35.

8. Schniederjans, M. J., Schniederjans, A. M., & Schniederjans, D. G. (2013). eBusiness, eCommerce, and eGovernment. In Business Logistics Management (pp. 529-551). IGI Global..

9. Lal, M. (1981). Inventory control: A case study of Modi Steels Limited. Vikalpa: The Journal for Decision Makers, 6(1), 39-44.

10. Madishetti, S., & Kibona, D. (2013). The impact of inventory management on profitability: A case of small and medium-sized enterprises in Tanzania. Journal of Economics and Sustainable Development, 4(1), 78-91.

11. Kovács, G., & Kot, S. (2016). Inventory management in the age of global economy. Acta Technologica Dubnicae, 6(1), 23-36.

12. Armenzoni, G., Borghesi, A., Bottani, E., & Mora, C. (2015). Inventory optimization: A combined approach for demand forecasting and service level estimation. International Journal of Production Economics, 160, 136-144,

13. www.shodhganga.inflibnet.ac.in