MUTUAL FUND RECOMMENDATION SYSTEM

M.Tech. Data Structure project report submitted to

By

Elroy Merwyn Monis
240984005

Department of Data Science and Computer Applications,
Manipal Institute of Technology, MAHE, Manipal
2024

ABSTRACT

This project focuses on developing a Mutual Fund Recommendation System aimed at helping users select suitable mutual funds based on their financial preferences and risk tolerance. The primary objective is to provide personalized fund recommendations by leveraging financial indicators such as expense ratio, standard deviation, Sharpe ratio, and fund performance. The system is built using Java Spring Boot for the backend and includes advanced features like KMP string matching for searching funds by name. Dijkstra's algorithm is employed to optimize fund selection based on composite scores and user-selected filters such as fund type, AMC, and risk category.

The front end integrates interactive filters, real-time search, and dynamic recommendations, making it user-friendly. Key findings show that this system efficiently narrows down mutual fund options, providing users with optimal recommendations based on their criteria. The implementation of KMP string matching for fund name search and the composite scoring method for fund ranking were crucial in enhancing the system's accuracy and usability.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In today's financial world, selecting the right mutual fund is a crucial decision for investors, as it directly impacts their financial growth and security. With hundreds of funds available, each having different attributes such as risk level, returns, fund size, and expense ratio, the decision-making process can become overwhelming. This project addresses the problem by developing a Mutual Fund Recommendation System that simplifies fund selection through data-driven insights and advanced algorithms.

The project leverages critical data structures and algorithms to optimize fund recommendation processes. Data structures such as graphs are used to model relationships between various funds, enabling the implementation of Dijkstra’s algorithm to find the most optimal fund based on user-defined criteria. Additionally, KMP (Knuth-Morris-Pratt) string matching is utilized to efficiently search for funds by their scheme names, improving the speed and accuracy of searches, especially in a large dataset.

The significance of this project lies in the application of these algorithms to solve a real-world problem. Dijkstra’s algorithm is instrumental in identifying the best mutual funds based on a composite score derived from multiple financial factors, while KMP string matching enables fast and reliable fund searching. These algorithms, coupled with an interactive and responsive frontend, make the recommendation system both powerful and user-friendly, offering investors a more streamlined and informed decision-making process.

2. LITERATURE REVIEW

1. **Das, A., Datar, M., Garg, A., & Rajaram, S. (2013)** The paper introduces scalable online collaborative filtering models, such as those used by Google News, focusing on real-time personalization. It highlights the challenges of large-scale data filtering but does not address financial-specific needs like risk adjustment in recommendations.
2. **Dijkstra,E.W.(1959)**
   Dijkstra's foundational work on graph-based algorithms introduces an efficient method for finding the shortest path in weighted graphs. This work is instrumental in optimization tasks like mutual fund recommendations, offering efficient performance for real-time decision making.
3. **Knuth, D. E., Morris, J. H., & Pratt, V. R. (1977)** The KMP algorithm offers a highly efficient approach to string matching by avoiding redundant comparisons, making it suitable for search operations like finding mutual funds by name. Its linear time complexity is well-suited for financial applications where speed is crucial.

3.PROBLEM STATEMENT

The project addresses the challenge of providing efficient and personalized mutual fund recommendations to investors, who face difficulties in managing and analyzing large volumes of financial data. Without effective data structures and algorithms, managing this data manually can lead to inefficiencies, inaccuracies, and poor decision-making. Investors often struggle to evaluate mutual funds based on metrics like returns, risk levels, fund size, and expenses. The absence of automated systems that apply robust algorithms, such as Dijkstra's for optimal recommendation and KMP for search, exacerbates this issue. Solving this problem is essential to enhance the accuracy and efficiency of mutual fund selection, enabling better financial decision-making for users.

4.OBJECTIVES

- To develop a mutual fund recommendation system that uses composite scoring to rank and suggest optimal funds.
- To implement an efficient KMP-based search functionality for quick fund lookup based on scheme names.
- To create a user-friendly interface for filtering and exploring mutual funds based on various criteria.
- To ensure accurate and real-time financial data management using appropriate data structures.
- To build a scalable system capable of handling large datasets while maintaining fast performance.

5. METHODOLOGY

1. **Data Collection**: The project began with the collection of mutual fund data from Kaggle, a reputable source for datasets.
2. **Data Preprocessing**: The collected data was preprocessed using Google Colab, where it was cleaned, missing values were handled, and relevant columns were formatted correctly.
3. **Composite Weight Calculation**: Composite weights were computed by creating a formula that combined key factors such as the expense ratio, Sharpe ratio, and fund returns.
4. **Data Storage**: The preprocessed data was imported into an Oracle SQL Developer database, ensuring that the data was stored with appropriate relationships and indexing for efficient querying.
5. **Backend Development**: The backend system was developed using Java Spring Boot, which involved designing RESTful APIs to interact with the database and process user requests.
6. **Algorithm Implementation**: Algorithms such as the KMP string matching algorithm were implemented for efficient search, while Dijkstra's algorithm was used to determine the best fund recommendations based on composite scores.
7. **Frontend Development**: The frontend interface was created using HTML, CSS, and JavaScript to allow users to filter and view recommended mutual funds easily.
8. **Integration & Testing**: Finally, the backend and frontend systems were integrated, followed by extensive testing to ensure the application functioned correctly.

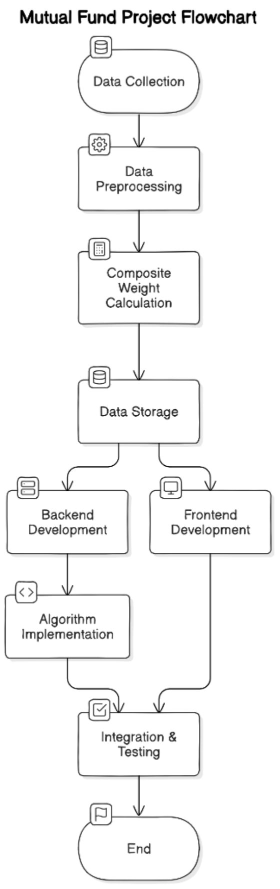


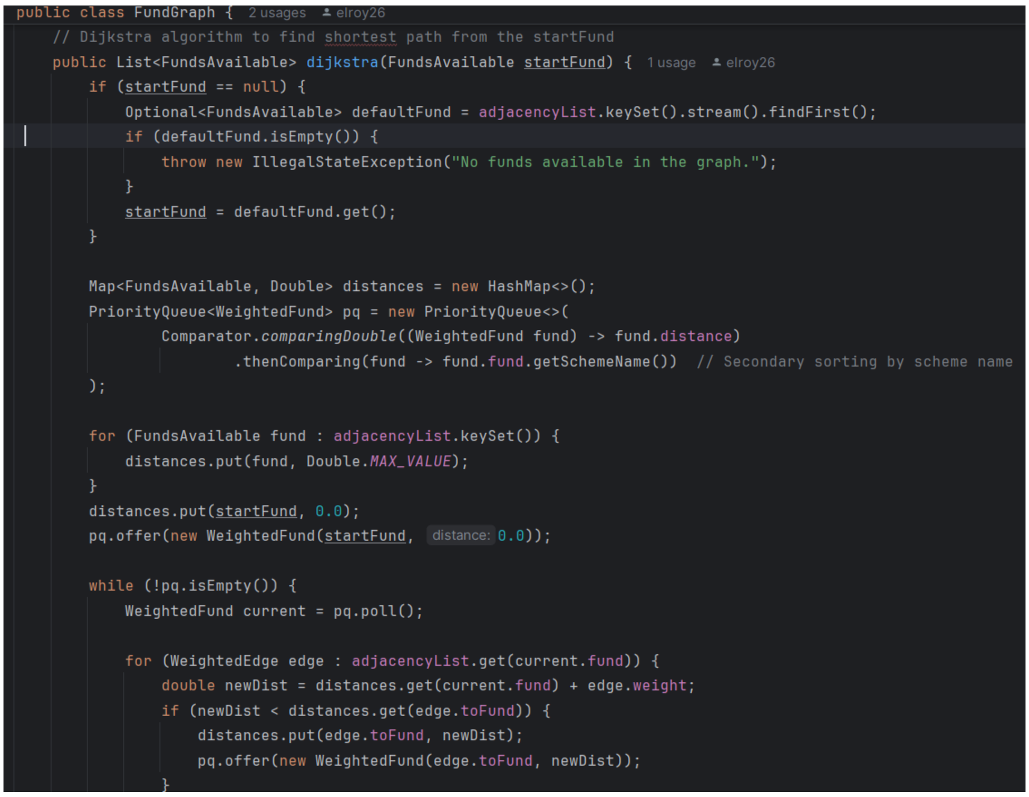
Fig. 1: Project Methodology

6. RESULTS AND DISCUSSIONS

The project successfully developed a mutual fund recommendation system using a variety of performance metrics and risk-adjusted measures. Key metrics, including the NAV (Net Asset Value), AUM (Assets Under Management), risk levels, and expense ratios of the funds, were displayed clearly on the user interface.

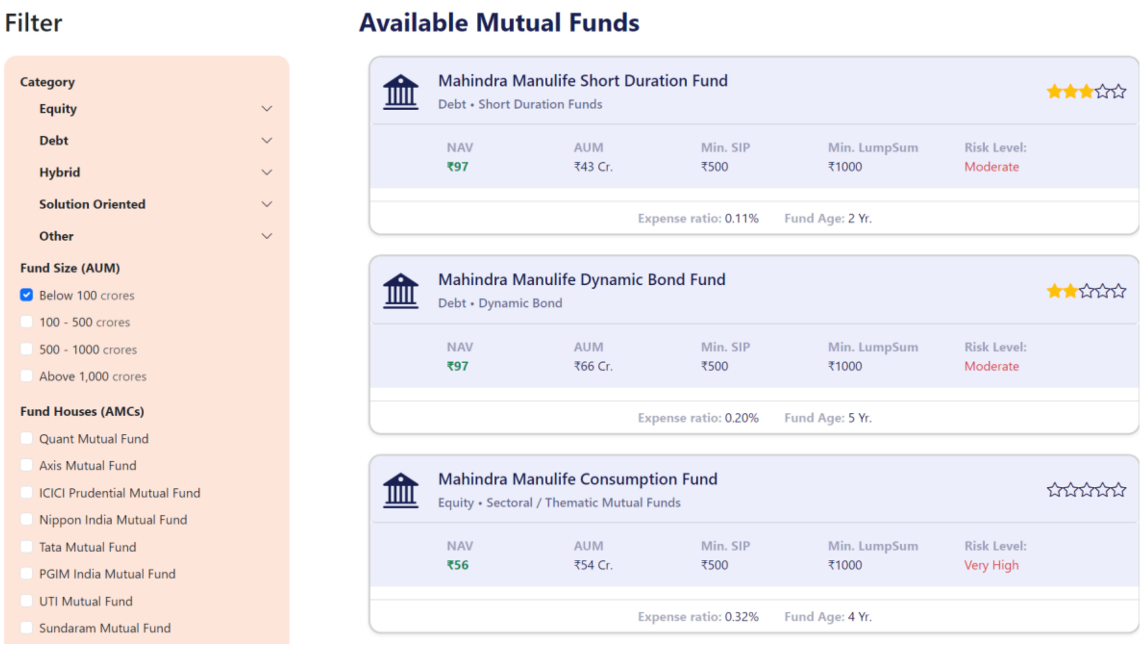
The recommendation system evaluated mutual funds based on composite scores calculated using parameters such as Sortino ratio, alpha, standard deviation, beta, and Sharpe ratio. This composite score allowed for an optimized fund ranking, helping users easily select the best performing funds according to their investment preferences.

1. Dijkstra Algorithm



```java
public class FundGraph {  2 usages  ≗ elroy26
    // Dijkstra algorithm to find shortest path from the startFund
    public List<FundsAvailable> dijkstra(FundsAvailable startFund) {  1 usage  ≗ elroy26
        if (startFund == null) {
            Optional<FundsAvailable> defaultFund = adjacencyList.keySet().stream().findFirst();
            if (defaultFund.isEmpty()) {
                throw new IllegalStateException("No funds available in the graph.");
            }
            startFund = defaultFund.get();
        }

        Map<FundsAvailable, Double> distances = new HashMap<>();
        PriorityQueue<WeightedFund> pq = new PriorityQueue<>(
                Comparator.comparingDouble((WeightedFund fund) -> fund.distance)
                        .thenComparing(fund -> fund.fund.getSchemeName())  // Secondary sorting by scheme name
        );

        for (FundsAvailable fund : adjacencyList.keySet()) {
            distances.put(fund, Double.MAX_VALUE);
        }
        distances.put(startFund, 0.0);
        pq.offer(new WeightedFund(startFund,  distance: 0.0));

        while (!pq.isEmpty()) {
            WeightedFund current = pq.poll();

            for (WeightedEdge edge : adjacencyList.get(current.fund)) {
                double newDist = distances.get(current.fund) + edge.weight;
                if (newDist < distances.get(edge.toFund)) {
                    distances.put(edge.toFund, newDist);
                    pq.offer(new WeightedFund(edge.toFund, newDist));
                }
```

Fig. 2: Dijkstra Algorithm code



Filter

Category
- Equity
- Debt
- Hybrid
- Solution Oriented
- Other

Fund Size (AUM)
- [✓] Below 100 crores
- [ ] 100 - 500 crores
- [ ] 500 - 1000 crores
- [ ] Above 1,000 crores

Fund Houses (AMCs)
- [ ] Quant Mutual Fund
- [ ] Axis Mutual Fund
- [ ] ICICI Prudential Mutual Fund
- [ ] Nippon India Mutual Fund
- [ ] Tata Mutual Fund
- [ ] PGIM India Mutual Fund
- [ ] UTI Mutual Fund
- [ ] Sundaram Mutual Fund

Available Mutual Funds

Mahindra Manulife Short Duration Fund ★★★☆☆
Debt • Short Duration Funds

NAV	AUM	Min. SIP	Min. LumpSum	Risk Level:
₹97	₹43 Cr.	₹500	₹1000	Moderate

Expense ratio: 0.11% Fund Age: 2 Yr.

Mahindra Manulife Dynamic Bond Fund ★★☆☆☆
Debt • Dynamic Bond

NAV	AUM	Min. SIP	Min. LumpSum	Risk Level:
₹97	₹66 Cr.	₹500	₹1000	Moderate

Expense ratio: 0.20% Fund Age: 5 Yr.

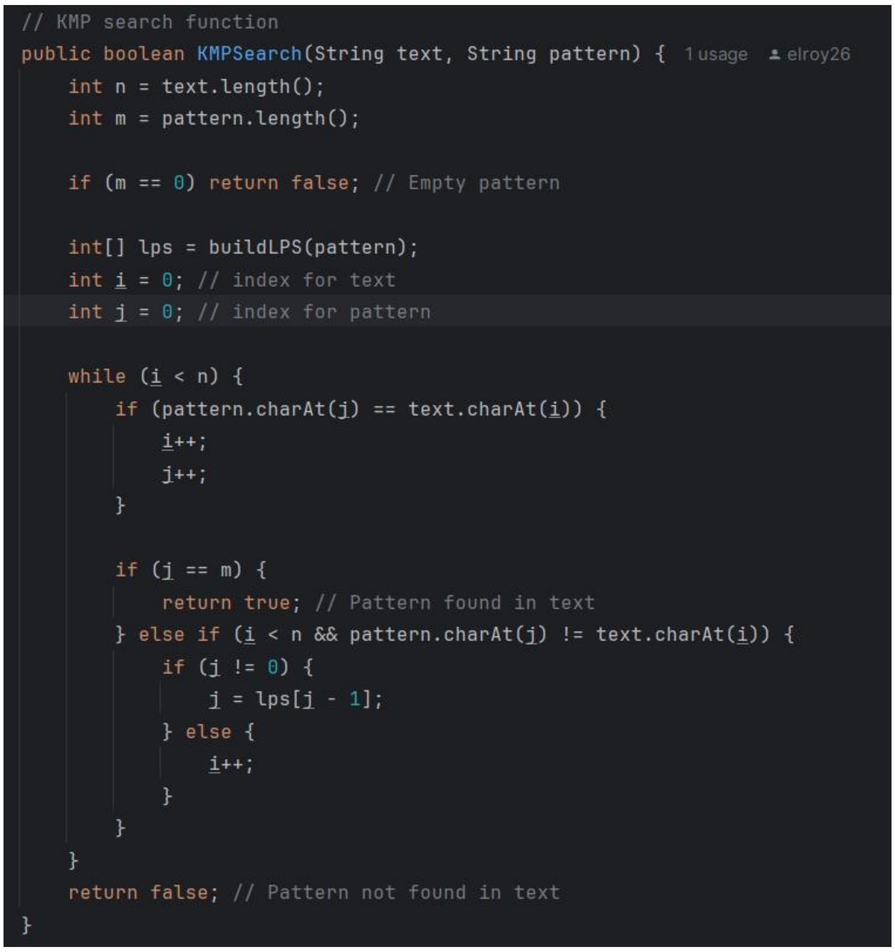
Mahindra Manulife Consumption Fund ☆☆☆☆☆
Equity • Sectoral / Thematic Mutual Funds

NAV	AUM	Min. SIP	Min. LumpSum	Risk Level:
₹56	₹54 Cr.	₹500	₹1000	Very High

Expense ratio: 0.32% Fund Age: 4 Yr.

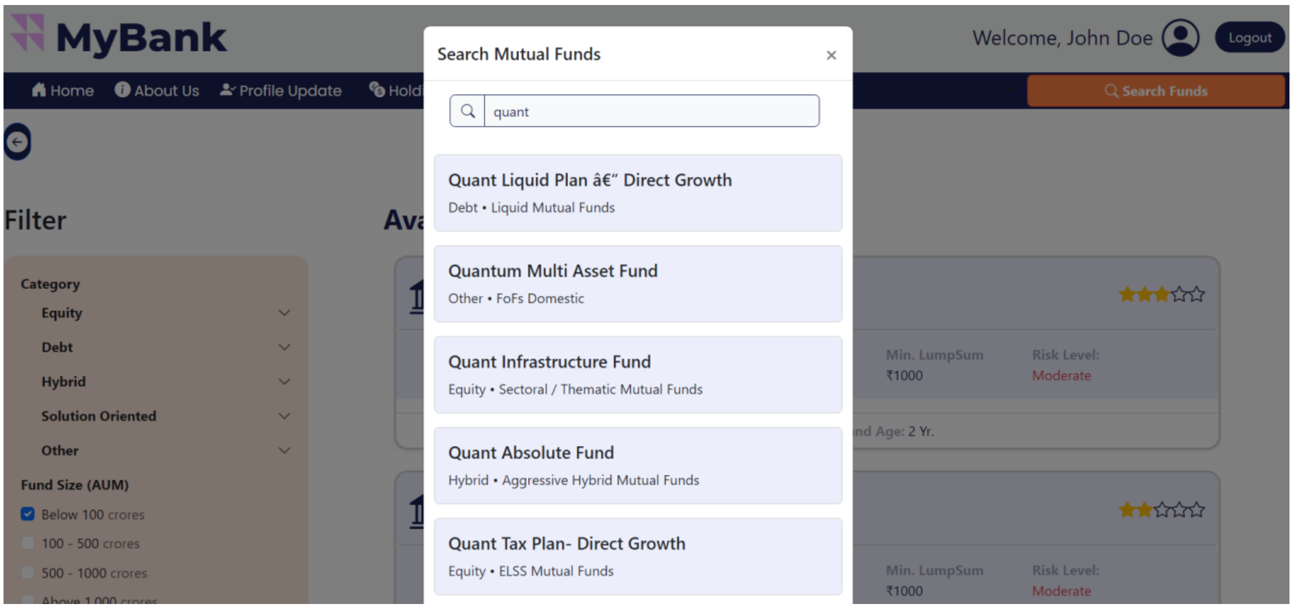
Fig. 3: Screenshot of Algorithm Output

2. Knuth Morris Pratt(KMP.) Algorithm



```
// KMP search function
public boolean KMPSearch(String text, String pattern) {  1 usage  elroy26
    int n = text.length();
    int m = pattern.length();

    if (m == 0) return false; // Empty pattern

    int[] lps = buildLPS(pattern);
    int i = 0; // index for text
    int j = 0; // index for pattern

    while (i < n) {
        if (pattern.charAt(j) == text.charAt(i)) {
            i++;
            j++;
        }

        if (j == m) {
            return true; // Pattern found in text
        } else if (i < n && pattern.charAt(j) != text.charAt(i)) {
            if (j != 0) {
                j = lps[j - 1];
            } else {
                i++;
            }
        }
    }
    return false; // Pattern not found in text
}
```

Fig. 4: KMP Algorithm Code



Search Mutual Funds

quant

Quant Liquid Plan â€“ Direct Growth
Debt • Liquid Mutual Funds

Quantum Multi Asset Fund
Other • FoFs Domestic

Quant Infrastructure Fund
Equity • Sectoral / Thematic Mutual Funds

Quant Absolute Fund
Hybrid • Aggressive Hybrid Mutual Funds

Quant Tax Plan- Direct Growth
Equity • ELSS Mutual Funds

Fig. 5: Output of KMP Algorithm

7. CONCLUSION

The project successfully developed a mutual fund recommendation system by leveraging data structures and algorithms to evaluate and rank funds based on risk-adjusted returns. All the key objectives were met, including implementing a composite scoring system, using Dijkstra’s algorithm for optimal fund recommendations, and providing a user-friendly interface. The use of Java Spring Boot for the backend and Oracle for data storage ensured a robust and scalable solution.

However, some limitations were encountered, such as the challenge of handling large-scale data efficiently, which impacted system performance slightly under heavy load. Future improvements could involve optimization techniques or integrating machine learning for more dynamic recommendations. Despite these limitations, the system proved effective in enhancing mutual fund selection and providing valuable insights to users.

8. FUTURE SCOPE

The future scope of the project with respect to data structures and algorithms includes optimizing the recommendation system for efficiency and scalability. More advanced graph algorithms, such as A* or Bellman-Ford, could be integrated for finding the shortest path or optimal mutual fund based on user preferences. Additionally, the current graph-based approach can be enhanced using dynamic graph algorithms to handle real-time updates in mutual fund data without recalculating from scratch. Data structures like balanced trees or hash maps could be used to improve search, insert, and update operations, reducing time complexity.

Furthermore, implementing parallel algorithms for tasks like data preprocessing or graph traversal could significantly enhance performance, especially with large datasets. Lastly, using more sophisticated priority queues (e.g., Fibonacci heaps) could further optimize Dijkstra's algorithm in recommending the best mutual funds, making the system more responsive and efficient.

9. REFERENCES

1. Das, A., Datar, M., Garg, A., & Rajaram, S. (2013). Google news personalization: Scalable online collaborative filtering. Proceedings of the 16th International Conference on World Wide Web, 271-280.
2. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269-271.
3. Knuth, D. E., Morris, J. H., & Pratt, V. R. (1977). Fast pattern matching in strings. SIAM Journal on Computing, 6(2), 323-350.