Non-Functional Requirements (NFR)

Introduction

This document details the top 3 Non-Functional Requirements (NFRs) prioritized during the development of our system. The focus was on ensuring the application meets the necessary standards for **Performance, Security, and Scalability**.

Top 3 NFRs

1. Performance

- **Description:** The system needs to handle a high volume of simultaneous requests without significantly degrading performance. Ensuring efficient processing of requests is critical for delivering a responsive user experience.
- **Reason for Prioritization:** The backend's ability to respond quickly to requests is crucial, especially for the work-intensive operations like custom sorting and auto-suggestions. Improving performance directly impacts user satisfaction.

2. Security

- **Description:** Protecting user data and ensuring only authenticated users can access restricted endpoints. Using Firebase Authentication and validating tokens is a primary mechanism for achieving this.
- **Reason for Prioritization:** Security is fundamental to prevent unauthorized access, protect sensitive user information, and avoid vulnerabilities such as NoSQL injection.

3. Scalability

- **Description:** Ensuring the system can scale efficiently as the user base and data volume grow. Considerations include implementing caching mechanisms, load balancing, and horizontal scaling.
- **Reason for Prioritization:** As the application’s popularity increases, the ability to scale without compromising performance or security is essential.

Trade-offs Made

Trade-off 1: Reduced Throughput for Enhanced Security

- **Description:** Implementing Firebase Authentication for every request introduces additional processing time, slightly reducing throughput.
- **Justification:** The benefit of having a secure system outweighs the minor performance impact, especially since security is a critical requirement for user data protection.

Trade-off 2: Prioritizing Performance over Code Maintainability

- **Description:** Certain performance optimizations were applied directly to critical backend functions (e.g., custom sorting and auto-suggestions) instead of creating generic solutions, reducing code reusability.
- **Justification:** This approach was acceptable since the goal was to improve user experience with efficient responses. Refactoring for maintainability can be scheduled for future sprints.

Conclusion

The prioritization of **Performance, Security, and Scalability** ensured the system met the necessary standards to provide a reliable, secure, and responsive user experience. While some trade-offs were made, they were justifiable given the overall improvement achieved.