Network Management: Design, Implementation, and Evaluation

Abstract:

This report presents the design, implementation, and evaluation of Network Management, a web-based application for monitoring and managing computer networks. The application provides various functionalities like network monitoring, port scan, subnet calculation, and a traceroute utility. The report follows a logical organization, discussing the architectural choices, design patterns, and code reusability.

1. Introduction

   Network Management is a comprehensive tool designed to aid network administrators in monitoring and managing computer networks. It combines various features, such as network monitoring, port scan, subnet calculation, and a traceroute utility, to provide a user-friendly interface for managing network resources. This report outlines the design, architecture, and implementation of the application, as well as the factors that influenced our design choices.

2. Architectural Overview

   The application follows a client-server architecture, utilizing React.js for the front end and Python with Flask for the back end. This architecture allows for a clear separation of concerns between the user interface and the backend logic, enabling code reusability and modular design patterns.

   2.1 Frontend

      The front end is developed using React.js, a popular JavaScript library for building user interfaces. The Material-UI library is employed to create a responsive and visually appealing interface. React's component-based architecture facilitates code reusability and maintainability.

   2.2 Backend

      The backend is built using Python and the Flask web framework, providing a lightweight and flexible solution for handling API requests and managing network-related tasks. The choice of Python allows for easy integration with various networking libraries, such as psutil and ipaddress.

3. Design Patterns and Code Reusability

   The project utilizes several design patterns and practices to promote code reusability and maintainability:

   3.1 Component-Based Architecture (React)

      React's component-based architecture allows for the creation of reusable UI components, resulting in a more modular and maintainable codebase.

   3.2 RESTful API (Flask)

The backend follows the RESTful API design, providing a standardized communication interface between the frontend and backend. This approach promotes scalability and modularity.

3.3 Dependency Injection and Inversion of Control

The application can easily swap out components or services using dependency injection and inversion of control, promoting flexibility and testability.

4. Alternative Strategies and Choices

Several alternative strategies were considered during the design and implementation process:

4.1 Alternative Frontend Frameworks

Vue.js and Angular were also considered for frontend development. However, React was chosen due to its popularity, extensive community support, and component-based architecture.

4.2 Alternative Backend Frameworks

Django and FastAPI were considered alternatives to Flask. Django was deemed too heavy for the project's requirements, while FastAPI, although promising, lacked the maturity and community support compared to Flask.

5. Evaluation and Conclusion

The Network Management was successfully completed based on the original design document, providing a comprehensive and user-friendly tool for network administrators. The chosen architecture and design patterns allow for a modular, maintainable, and scalable application, while the use of React and Flask ensures a robust and efficient solution.

Future work on the project could include integrating additional network management features, performance optimizations, and enhanced documentation. Overall, the Network Management is a powerful tool to help network administrators monitor and manage their networks more effectively.