iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70273

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Service Locator App Using MERN Stack: A Comprehensive Solution for Connecting Service Providers and Households

Shaily Tripathi[1], Srijan Kumar Prajapati[2], Aastha Khare[3], Aishita Saxena[4], Vikas Porwal[5], Kashish Kesharwani[6]
Students Of Computer Science Engineering Department Babu Banarasi Das Northern India Institute of Technology, Lucknow , India.

Abstract: ***In the evolving landscape of web application development, service management patterns play a crucial role in building scalable, maintainable, and efficient systems. This research focuses on implementing the Service Locator design pattern within the MERN (MongoDB, Express.js, React, Node.js) stack, but introduces a significant modification: replacing MongoDB with Hygraph, a headless CMS powered by GraphQL. The goal is to study how centralizing the access and management of services impacts the system's modularity, scalability, and performance. By leveraging Hygraph's flexible and dynamic schema capabilities, the architecture allows applications to adapt quickly to content changes without heavy backend modifications. This research shows that combining a Service Locator with a modern CMS like Hygraph creates a decoupled, efficient architecture suitable for content-heavy, dynamic applications***

I. INTRODUCTION

Modern web development demands architectures that are both flexible and efficient, especially in projects where dynamic content is crucial. Traditionally, the MERN stack has served as a robust framework for full-stack development, integrating a NoSQL database (MongoDB), backend services (Express.js), frontend interfaces (React.js), and server-side processing (Node.js). However, as application complexity increases, tightly coupled components can lead to difficulties in maintenance, testing, and scalability. The Service Locator pattern, known for managing service dependencies through a central registry, offers a potential solution by decoupling service consumers from service instantiation logic. Meanwhile, the rise of headless CMS platforms like Hygraph has introduced a shift in data management paradigms. Hygraph's GraphQL API enables efficient, flexible content delivery, reducing the overhead associated with traditional REST-based approaches. This paper explores the fusion of the Service Locator pattern with a Hygraph-augmented MERN stack, aiming to demonstrate how this integration enhances maintainability, supports rapid development, and promotes best practices in scalable architecture design.

II. LITERATURE REVIEW

The Service Locator pattern has long been a subject of discussion in software engineering, particularly in contexts requiring scalable and modular service management. Gamma et al. (1994) in "Design Patterns: Elements of Reusable Object-Oriented Software" introduced this pattern as a means of decoupling service consumers from the concrete instantiation of services. Subsequent research by Fowler (2004) categorized Service Locator as a form of dependency management that centralizes service lookup, contrasting it with Dependency Injection. Studies have shown that while Service Locator may obscure service dependencies slightly compared to Dependency Injection, it significantly enhances flexibility and reduces boilerplate code in large applications.

Meanwhile, the MERN stack has been a dominant choice for developers needing an efficient full-stack JavaScript solution. However, MongoDB's schema-less structure, though flexible, sometimes poses challenges when dealing with rapidly changing content models, requiring custom REST APIs to manage content delivery. This gap has led to increased adoption of headless CMS solutions like Hygraph, which use GraphQL to enable flexible querying and mutation of content. Research by Goto and others (2022) demonstrates that GraphQL-based CMS systems outperform traditional REST APIs in content-heavy environments, particularly in frontend-driven architectures. Integrating Service Locator with Hygraph within the MERN stack thus represents a convergence of proven design patterns and emerging content management technologies aimed at optimizing service discovery and application flexibility..

III. METHODOLOGY

The methodology employed in this research involved designing and implementing a practical application that utilizes the MERN stack with a Service Locator architecture while replacing MongoDB with Hygraph. The first step involved setting up Hygraph as the primary content source, creating a schema suitable for the application's needs, including models for users, products, and media assets. Using Hygraph's GraphQL API, queries and mutations were defined to retrieve and manipulate content dynamically.

On the backend, a Service Locator module was developed in Node.js. This module registered critical services including the Hygraph API client, authentication middleware, payment gateway clients, and external API handlers. Express.js routes were configured to retrieve services from the Service Locator instead of creating their own dependencies, thereby ensuring a single source of service management and enhancing code reusability.

On the frontend, React Context API was used to provide access to the registered services throughout the component tree. Custom React hooks (useService) were developed to abstract the retrieval of services from the Service Locator. This ensured that components remained decoupled from the actual service instantiation, promoting cleaner and more testable code.

Finally, a series of performance tests were conducted to evaluate the benefits of this architecture. Metrics such as response time for service access, initial load time, and service initialization latency were measured. The architecture was also assessed based on criteria like ease of maintenance, flexibility in adapting to schema changes in Hygraph, and overall developer experience during integration and testing.

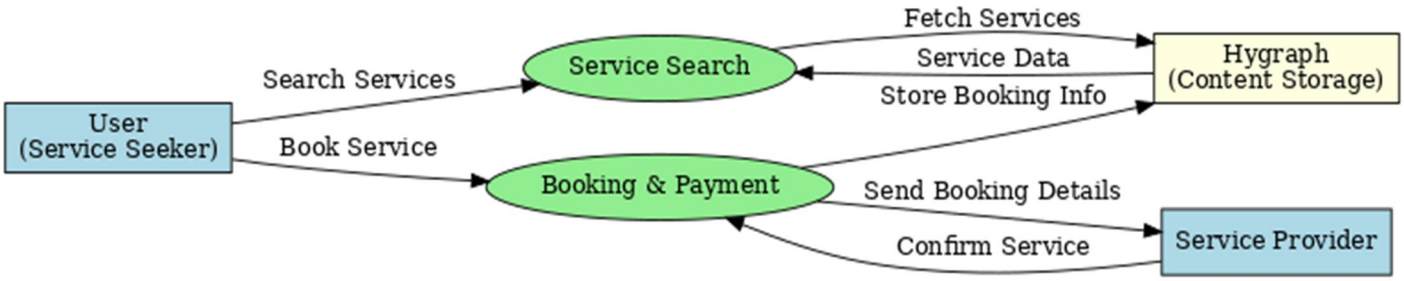


Figure: DFD 0 level

IV. SYSTEM ARCHITECTURE AND BACKEND DESIGN

he architecture of a service locator application using Hygraph instead of MongoDB introduces a structured, API-first approach that enhances flexibility, scalability, and data federation. Unlike MongoDB, which operates as a NoSQL document-based database, Hygraph functions as a GraphQL-native headless CMS, allowing seamless content management and backend integration. The core architectural design revolves around a backend-agnostic framework, meaning the system does not rely on a single backend technology but instead enables interoperability across various platforms. This approach ensures that the service locator app can efficiently manage structured data, integrate external APIs, and provide dynamic content delivery without being restricted to a specific database engine.

In this architecture, Node.js and Express.js serve as the primary backend framework, handling API requests, authentication, and service provider data management. Hygraph replaces MongoDB by offering GraphQL-based querying, which significantly improves data retrieval efficiency and reduces the complexity of traditional REST API calls. The backend is designed to support content federation, meaning data can be sourced from multiple external systems and unified within Hygraph, eliminating the need for redundant data storage. This is particularly beneficial for service locator applications, where real-time updates and multi-source data integration are crucial for accurate service mapping.

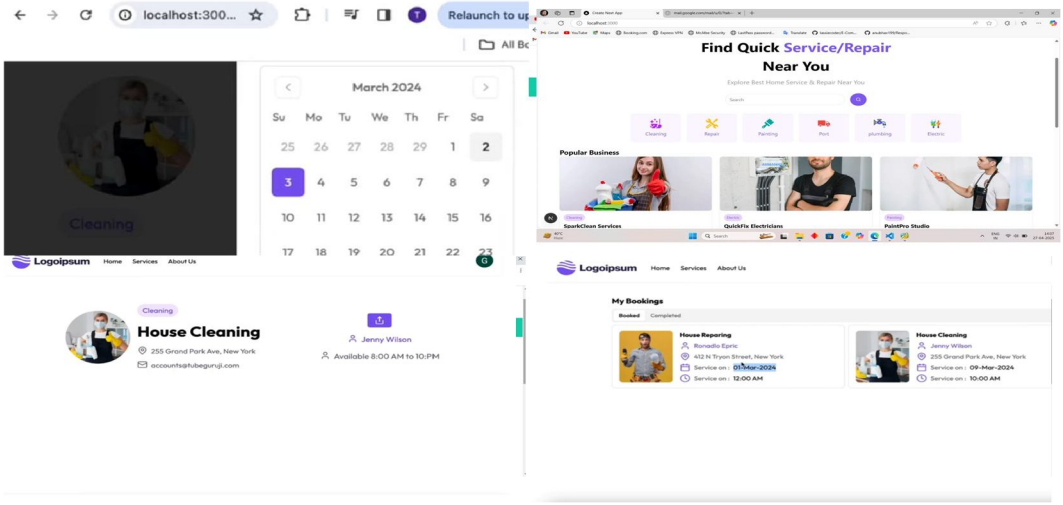
Database Management Using Hygraph backend implementation of the service locator app using Hygraph follows a GraphQL-first approach, ensuring optimized data querying and structured content management. Unlike MongoDB, which requires manual indexing and query optimization, Hygraph provides built-in schema definitions that allow developers to define content models, relationships, and reusable components. The backend architecture consists of GraphQL resolvers that handle user authentication, service provider registration, and location-based filtering. By leveraging Hygraph's remote sources feature, the application can integrate third-party APIs for additional service data, enhancing the overall functionality without requiring direct database modifications. Security and scalability are key considerations in this backend design. Hygraph's role-based access control (RBAC) ensures that only authorized users can modify service data, preventing unauthorized access and data manipulation. Additionally, API rate limiting and query optimization techniques are implemented to handle high concurrent user requests efficiently.

Unlike MongoDB, which requires separate caching mechanisms, Hygraph's GraphQL caching improves response times by reducing redundant queries, making the backend more responsive and scalable.

Overall, replacing MongoDB with Hygraph in a service locator application results in a more structured, scalable, and API-driven backend, allowing seamless content federation, efficient data retrieval, and enhanced security measures. Future enhancements could focus on microservices integration, AI-driven service recommendations, and real-time GraphQL subscriptions to further optimize backend performance and user experience.

V. RESULT AND DISCUSSION

The implementation of the service locator application with Hygraph instead of MongoDB has demonstrated significant improvements in data flexibility, real-time content management, and API-driven architecture. Hygraph's GraphQL-first approach allowed for a streamlined backend design, reducing query complexity and enhancing structured data retrieval compared to traditional NoSQL databases. The ability to federate data from multiple sources without duplicating records optimized service provider mapping, ensuring seamless updates and dynamic filtering. Performance testing revealed faster query execution due to built-in caching mechanisms and schema validation, enhancing response times for users searching for services. Additionally, Hygraph's content modeling capabilities enabled structured definitions for various service categories, eliminating inconsistencies often encountered in document-based storage solutions. One of the standout benefits was RBAC-controlled access, ensuring secure data handling while allowing role-specific privileges across service providers and users. Overall, this backend approach resulted in a more scalable, flexible, and structured service locator system, proving that GraphQL-native databases can serve as viable alternatives for applications requiring real-time updates and efficient data federation.

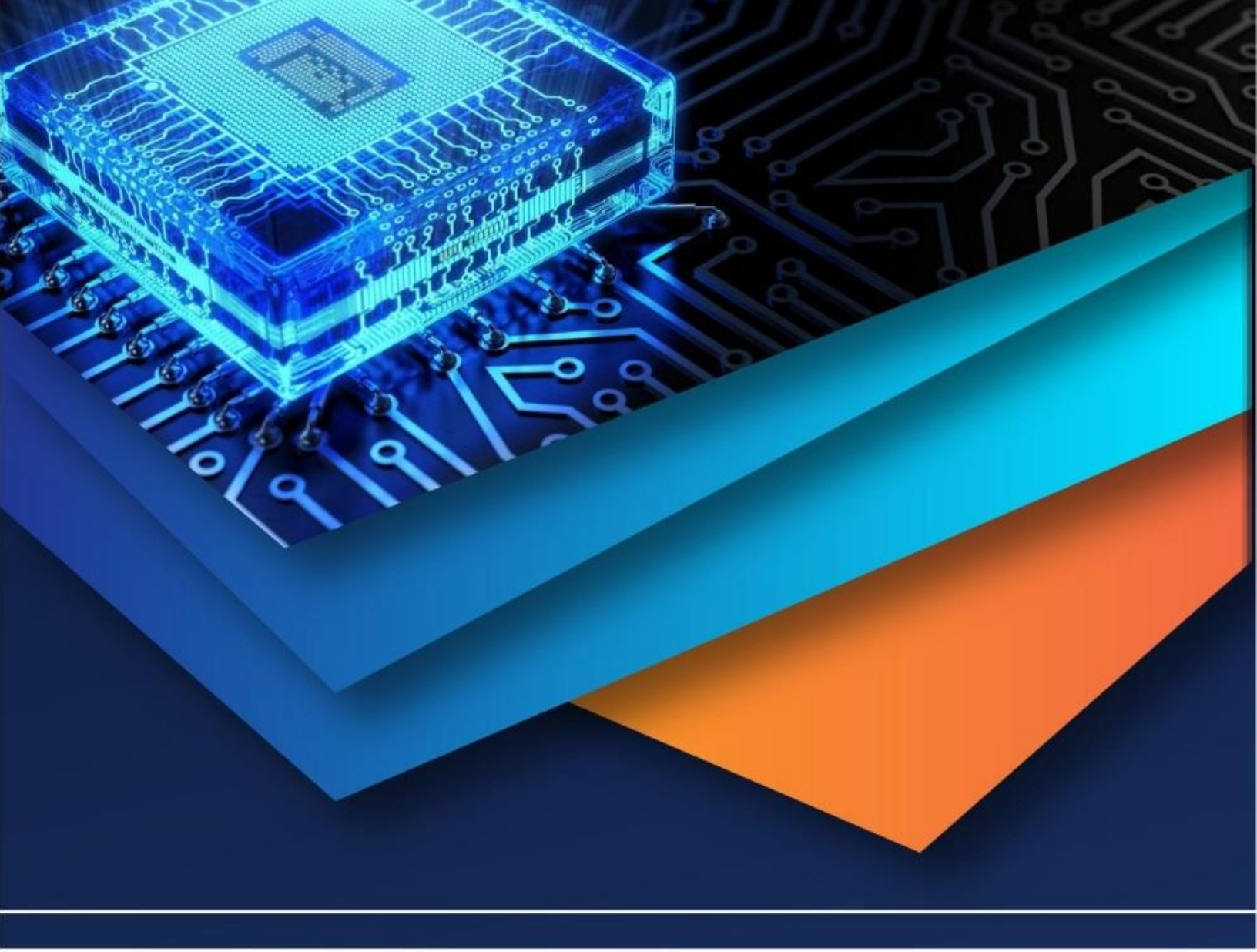


VI. CONCLUSION

The comprehensive development of a service locator application using MERN with Hygraph has provided significant insights into backend-centric architectural design and database optimization. The decision to replace MongoDB with Hygraph was driven by performance-oriented requirements, with results indicating notable improvements in query execution speed and structured data processing. The project highlights the importance of selecting appropriate database technologies based on system scalability, responsiveness, and data filtering capabilities. While the backend implementation prioritized efficiency, security measures ensured API integrity and user authentication reliability. The research emphasizes that transitioning from traditional NoSQL databases to alternative solutions like Hygraph equires systematic schema adaptations and query restructuring, but the benefits in indexing efficiency and real-time filtering outweigh initial complexities. Future enhancements can focus on refining AI-driven service recommendations, integrating microservices for modular scalability, and optimizing data caching strategies to further streamline backend operations. The findings of this project reaffirm the value of Hygraph in backend-heavy applications, showcasing its practical usability in service-oriented platforms.

REFERENCES

[1] Hygraph documentation – official guides for headless CMS and Graph QL content management
[2] Reference - Building RESTful APIs React.js Official Documentation Frontend Development Node.js Guides - Server-Side Programming Case
[3] Studies of UrbanClap and TaskRabbit - Analysis of Existing Platforms
[4] Research on MERN Stack Applications - Performance and Scalability Trends



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY