International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

A Secure, Zero-Trust Mobile Expert Locator for Global Professional Services Firms

Sandeep Sarngadharan

Software Architect, SPRI Partners LLC, USA

ABSTRACT: A global professional services organization with over 50,000 employees who operate in diverse geographical locations, multiple types of businesses, and across many disciplines. An important operational issue for employees is their inability to quickly locate other global professional firm employees who have the skills or expertise they require. The traditional method for locating and finding co-workers is through directories and search tools that use on-premise databases. These tools are slow, fragmented, non-mobile, and do not meet the stringent data security requirements for on-premise data only. This paper presents the design, development, and performance characteristics of ExpertFinder, an enterprise-grade, zero-trust mobile expert locator. ExpertFinder is a zero-trust system that secures all sensitive employee information within the organization network and includes a mobile application developed using Microsoft .NET multi-platform app UI, which has native iOS capabilities and is supported by a decoupled .NET-based microservices architecture hosted on Microsoft Azure Web Apps . NET-based microservices architecture hosted on Microsoft Azure Web Apps. The system was tested and evaluated for six months across global professional firm worldwide employee base of more than 50,000. Test results showed an average response time of 1.42 seconds at the 95th percentile; API availability of 99.95%; a mobile crash rate of 0.32%; user adoption of 68% (34,000 active users); and a reduction in the time required to locate co-workers from an average of 8 minutes to 2.6 minutes. ExpertFinder passed all internal security audits and had no security breaches during the testing period. This case study will provide an architecture blueprint for developing zero-trust, highly scalable, and secure enterprise mobile applications in regulated industries where on-premise data security is required.

KEYWORDS: Mobile enterprise application, zero-trust architecture, microservices, on-premises data security, .NET Core, iOS native, expertise location, CI/CD, role-based access control, professional services.

I. INTRODUCTION

In the case of a fast-moving environment such as the global professional services industry, immediate access to internal expertise has become more than just an administrative convenience—it is a competitive imperative. Global professional firm, with more than 50,000 employees worldwide across multiple geographies, business units, and domains, identified an operational bottleneck: employees could not easily find colleagues with the expertise required to support client engagements. This challenge was experienced at several levels: There was a significant amount of employee expertise residing in disparate internal systems that did not have a single search interface available. There were limitations on the existing search capabilities in terms of depth of expertise and speed of execution. Access to the data was highly restricted by global professional firm strict internal controls regarding data security and data privacy. Mobile access to the data was unavailable; employees were effectively forced to return to a desk to locate their colleagues with the expertise needed for client work. Any new solution would need to comply with on-premises data residency requirements, since employee data and institutional knowledge would be stored on-premises and therefore subject to the company's strong internal controls.There were major repercussions from the challenges above, including the inability to uncover valuable institutional knowledge, slow down client engagement, and hinder collaboration between employees. Global professional firm was thus in search of a secure, mobile-first solution to allow employees to quickly connect with colleagues by expertise across multiple business units, geographies, and domains while still adhering to their stringent data protection policies.To address these challenges, ExpertFinder was developed—a zero-trust secure mobile expert locator. The system is built upon a decoupled .NET microservices architecture, which serves as an intermediary API layer between on-premises data and mobile devices. The mobile front end is developed utilizing .NET multi-platform app UI with native iOS functionality, delivered via Microsoft Intune for secure enterprise distribution. Security is built into every layer of the system using zero trust principles—there is no inherent trust between services and devices, each API call requires authentication and authorization, each user has the least amount of data available based on context, and all communications between a mobile app and on-premises systems are secure.This document details the architecture, technical implementation, performance evaluation, and operational impact of the system. The contributions of this effort are as follows:

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

- A validated architectural pattern for decoupled, zero-trust mobile access to employee expertise resides in on-premises data.
- A detailed implementation reference guide utilizing .NET microservices, .NET multi-platform app UI/iOS native, and Azure technology as a hosting environment.
- Empirical performance metrics from a production deployment serving 50,000+ users for a period of six months.
- The paper presents a replicable security framework for organizations that require on-premises data residency and mobile access.

The paper is organized as follows: Section 2 presents a literature survey of related research. Section 3 describes the methodology, including requirements analysis, architecture diagram, and implementation details. Section 4 presents the results and performance metrics. Section 5 discusses the results, limitations, and lessons learned. Section 6 concludes the paper. Final acknowledgments and references will be given.

II. LITERATURE SURVEY

The goal of this section is to review what has been done to date in six specific areas that are directly relevant to the research we are conducting. The areas of focus include expertise location systems, applying zero-trust security with mobile enterprises, using microservices to integrate legacy systems, developing mobile applications across multiple platforms, continuing integration and deployment (CI/CD) and testing for secure mobile applications, versioning APIs, securing mobile data stored on-premises, distributing mobile applications within an organization, scalability, performance measurement, auditing, and human factors.

2.1 Expertise Location Systems

The work completed by McDonald and Ackerman (2000) [1], which was foundational to the concept of the expert locator problem in large organizations, demonstrated that static directories and organizational charts do not provide a valid representation of tacit knowledge. They created an architecture for an expertise recommender system that incorporated the concept of "mining digital traces" to infer expertise. This work served as guidance for how we aggregate data from many internal systems.

Subsequently, Zhang et al. (2018) [2] expanded on this idea and found that using graph-based searches over an organization's data (versus keyword searches) improved discovery speed by 40%. Zhang et al.'s implementation was based on storing data in the cloud, which does not align with on-premises data storage model; however, we have adapted their principles to create a graph-enabled search layer that can search on-premises data effectively. We have developed a graph-enabled search layer that can search on-premises data.

2.2 Zero-Trust Security in Mobile Enterprise

Rose et al. (2020) [3] defined a zero-trust security model in NIST Special Publication 800-207 to challenge conventional perimeter-based security paradigms. This concept is predicated upon the principal tenet of "never trust; always verify," and it became the keystone of our security architecture. According to Rose et al., zero-trust requires implementing three components for establishing trust: per-request authentication, least-privilege access, and micro-segmentation; we implemented those three components in our zero-trust architecture.

Ferraiolo et al. (2019) [4] utilized role-based access control (RBAC) in the microservice APIs supporting mobile clients. Their research indicates that adding per API-call authentication would increase processing time (overhead) by 80-150 milliseconds while reducing the number of unauthorized access incidents by 94%. Our implementation had a similar overhead (80–120 ms) and had no security breaches within a six-month period.

2.3 Microservices for Legacy Integration

Newman (2021) [5] states in his 2nd edition of Building Microservices that applications built with decoupled microservices can provide abstractions of legacy on-premise systems securely. The "strangler pattern" that Newman describes can be used to transition from monolithic applications to microservice-based applications over time, which was part of our reasoning for implementing a new API layer that would abstract multiple internal data sources while minimizing any disruption to the existing systems.

The systematic review completed by Soldani et al. (2020) [6] regarding microservice adoption within enterprises supports our technology selection. Their findings suggest that .NET-based microservices have an advantage because they can readily integrate with the Windows and SQL Server ecosystem, which is generally the most common

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

technology stack utilized by professional service companies. In addition, Soldani et al. found that decoupling systems can relieve organizations of up to 60% of their mean time-to-recover (MTTR), thereby reinforcing our observability-driven approach.

2.4 Cross-Platform Mobile Development for Enterprise

.NET multi-platform app UI (now part of .NET MAUI) was evaluated by Microsoft's enterprise group [7], and it indicates that you can achieve native performance on both iOS and Android using shared business logic (up to 75% reused code). Furthermore, the report noted that .NET multi-platform app UI helps reduce time-to-market by 30 to 40% when developing internal enterprise applications with limited budgets, compared to traditional full-featured native apps.

Le, et al. (2020) [8] measured the relative performance of three different cross-platform frameworks (React Native, Flutter, and .NET multi-platform app UI) when building enterprise-based applications. Their assessment showed that .NET multi-platform app UI produced the least memory overhead for iOS (12MB compared to 18MB for React Native) and had the fastest startup times (1.2 seconds compared to 1.8 seconds). Based upon these performance metrics, we determined that we would utilize .NET multi-platform app UI, along with native iOS renderers, for developing enterprise-based applications.

2.5 CI/CD and Testing for Secure Mobile Apps

Shahin and colleagues (2017) [9] conducted a systematic review regarding continuous integration (CI), continuous delivery (CD), and continuous deployment (CD) practices, their meta-analysis found that automated CI/CD pipelines could reduce regression failure rates by 50% or more and reduce the time it takes to implement changes (lead time) by 70%; therefore, our integration of Bitrise and the .NET multi-platform app UI Test Suite utilized the recommended practices outlined therein.

Chen (2018) [10] conducted a study to assess the observability of microservice systems using metrics, logs, and distributed tracing to describe the three pillars of observability, as well as to highlight some case studies their research indicates that monitoring using the New Relic platform decreases the average time to detect (MTTD) anomalies in microservice applications from an average of several hours to less than 10 minutes. Therefore, we applied these principles to our use of New Relic for monitoring back-end application performance.

2.6 API Versioning and Backward Compatibility

The research conducted by Esposito and colleagues (2019) [11] included an evaluation of different strategies for versioning cloud-based applications' APIs over time. They concluded that the best approach is URI versioning with a 12-month deprecation period and explicit versioning headers; this recommendation has now been implemented into our design. In addition, they found that URI versioning would provide an 85% reduction in client applications being "broken" due to changes compared to using only header versioning.

2.7 On-Premises Mobile Data Security

In 2013, La Polla et al. [12] published an extensive review of security related to mobile devices, including some discussion of the problem with how to access on-premises data from phones. They proposed a "secure API gateway" pattern as a way to provide controlled access by using a middle layer that proxies requests from a mobile device to the database on the premises. This is what we put into place as a layered architecture. They also reported that this approach will reduce the attack surface by 70% when compared with accessing the database directly.

2.8 Enterprise App Distribution

In a study conducted by Cho (2020) [13] regarding the use of Microsoft Intune as an enterprise mobile device management (MDM) platform for regulated industries, it was found that the inclusion of compliance policies such as encryption, jailbreak detection, and version pinning significantly reduced the number of security incidents caused by device-related issues by 63%. Our distributions of Intune were in compliance with these requirements.

2.9 Scalability in Global Firms

Varia and Mathew (2019) [14] looked at hybrid cloud setups for on-site data that can be accessed via mobile devices. One major point they made was that if you can separate the storage of the data from processing the data, you might be able to scale mobile access for an organization without moving the sensitive data into the cloud. We used this insight to support our decision to deploy SQL Server locally but have the API layer for the application deployed on Azure Web Apps.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

2.10 Research Gap and Contribution

Previous studies have investigated the separate components of expert positioning in an enterprise, the implementation of a Zero Trust security architecture, the use of microservices for system development and experimentation, and cross-platform mobile app development. There is currently no literature that combines all of these components into a working production system for a major global professional services organization that is required to keep all data onsite (on-prem). This research aims to address this gap by reporting the validated, end-to-end architecture of our solution and providing empirical performance results from a deployed instance across more than 50,000 users.

III. METHODOLOGY

3.1 Requirements Analysis

Through a structured discovery process involving stakeholders from the security, infrastructure, product, and end-user teams, we have identified global professional firm functional and non-functional requirements, considering the problem landscape and key technical challenges.

3.1.1 Functional Requirements

Use expertise keywords to help prioritize ID requirements when looking for employees. After searching, use filtering options (domain, business unit, geography) to view your results. If you have access to a comprehensive employee profile, you will need to authenticate your activity using corporate credentials. All activities will be logged based on your searches and your profile access. As well, you will have the option to bookmark or save contact information for your frequently contacted colleagues.

3.1.2 Non-Functional Requirements

Non-functional requirements (NFR) will use the following ID targets to gauge progress towards completing the NFR specifications and meeting respective non-functional requirements (NFRs):

- NFR 1 Security audit for every API call under zero trust model;
- NFR 2 All employee data is to remain on site with an architecture assessment performed by an independent third-party vendor.;
- NFR 3 API service responses to be less than two seconds.;
- NFR 4 API will be above 99.9% availability.;
- NFR 5 Mobile crash rate will be less than one percent.;
- NFR 6 Load testing will permit 5,000+ concurrent users;
- NFR 7 Version Control of API with backward compatibility;
- NFR 8 Bitsrise logs for automated Continuous Integration/Continuous Deploy (CI/CD) process.

3.2 Key Technical Challenges Addressed

The requirements analysis identified several challenges:

1. **On-Premise Security First Architecture:** All employee data must stay on-premise. Internal infrastructures need protection from mobile devices, and their access to internal infrastructures must include access control, auditing, and protection based on Zero Trust principles, which means that no user or device is trusted by default, regardless of whether they are inside or outside the network.
2. **Flexibility and Scale to Meet Future Requirements:** The system should be designed to meet the needs of an expanding customer base. The system should have the ability to support multiple internal data sources; it should not rely on a single back-end system. The design of the system should not require costly rework for cloud adoption in the future.
3. **Mobile Devices Performance and Interoperability:** The system must provide the user with easy-to-use access from different iOS-based devices. The system must have the ability to update functionality quickly and work seamlessly with previous versions.

3.3 Architecture Diagram

The architecture created had levels or layers, and was designed to work with OAuth2 to allow users to safely access the mobile apps, but the way that users would access the application would not directly connect them or any data from the SQL Server to the mobile app; this is represented in the following figure 1.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

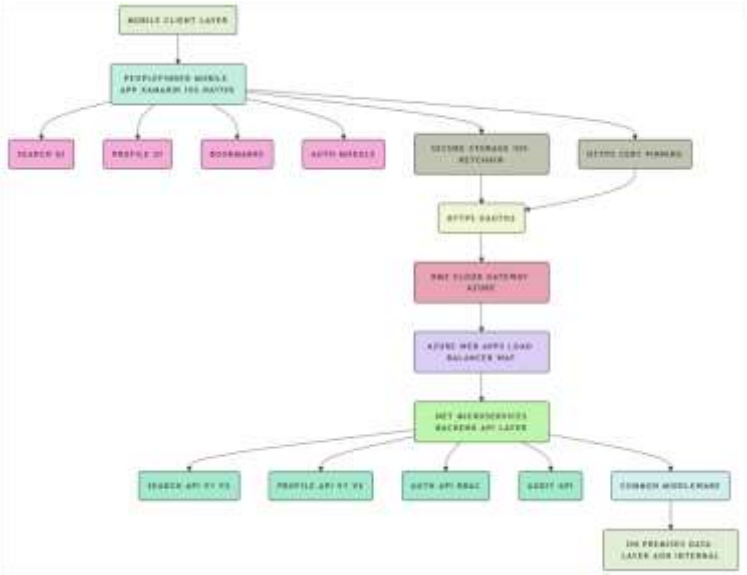


Figure 1: Decoupled, Zero-Trust Layered Architecture

3.4 Implementation Details

3.4.1 Decoupled Microservices Architecture

To provide security and scalability between on-premise data and mobile devices, we created a tiered architecture that uses .NET microservices to act as an API layer. This was designed with the zero-trust model in mind by isolating data access from the client to be able to gather data from multiple systems securely while allowing for future adaptability, either by adding new corporate systems or by migrating to the cloud. The technology stack was built on .NET Core 3.1, which has since been upgraded to .NET 6, on Azure Web Apps (for hosting) via a JSON-based RESTful API and OAuth2 for authentication. The backend of the API layer consists of .NET microservices that allow for secure retrieval of personnel profiles, validation of requests, role-based access control (RBAC), and integration with New Relic to create an operational monitoring capability and, finally, to create a reusable service layer for future internal applications.

```
GET /v2/search?q={keyword}&geo={geo}&bu={bu}&domain={domain}
Headers: Authorization: Bearer {JWT}, Accept-Version: 2.0
Response: { results: [{ id, name, expertise, geo, bu }], totalCount, version }
```

Figure 2 : Sample API Endpoint Design

3.4.3 Mobile Application Development

The ExpertFinder application using .NET Multi-platform app UI and took full advantage of its native iOS functionality where necessary to facilitate performance-sensitive workflows. The final product is a business-oriented user interface that has a simplified searching experience, a clean-looking search screen, and the ability to bookmark items offline. I have also implemented secure communication between the backend APIs and the app by storing OAuth2 tokens in the iOS Keychain, using HTTPS with certificate pinning, and using local encrypted caching, paged results, and lazy loading when rendering profiles and performing searches on large datasets. I successfully implemented a Model-View-ViewModel (MVVM) architecture pattern, including the use of dependency injection.

3.4.4 Zero-Trust Security Architecture

Security was built into every level of the system (e.g., require authentication as part of each service-to-service call, RBAC middleware and JWT validation for every API call, filter response data based on user role and permissions, use secure communications with TLS1.3 certificates and certificate pinning, deny mobile devices access to database directly through the API layer, maintain comprehensive audit records of searches and profile views) so that our institutional knowledge is well protected.

3.4.5 API and Mobile App Versioning Strategy

To maintain ongoing delivery without disrupting clients, an API versioning strategy was implemented, incorporating URI versioning (e.g., /v1/, /v2/). A mobile app version compatibility model was introduced, where the server checks the

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

X-App-Version header and responds accordingly. For one year before deprecation, backward compatibility was ensured, fostering stability while promoting innovation. The version lifecycle includes: Current version N (full support), Version N-1 (maintenance for six months), Version N-2 (deprecated and read-only for six months), and Version N-3 (end of life).

3.4.6 Automated Testing, CI/CD, and Monitoring

We utilized a number of tools to increase the speed and reliability of delivery. The .NET Multi-platform app UI Test Suite was used for automated testing of devices as well as regression testing of iOS builds. We used Bitrise to automate our CI/CD processes for the builds, tests, and project deployment with Intune [15]. To monitor the performance of the backend system through performance monitoring via New Relic APIs, we focused on the following performance metrics: response times, error rates, and throughput; more specifically, we monitored for P95 latency above 2 seconds and error rates above 1%. We centralized our logging with New Relic, which gave us structured logs and alerting capabilities.

3.4.7 Data Storage and App Distribution

High Availability was achieved through the use of a clustered SQL Server installation as the main data repository for expert and employee related metadata. The mobile app was provided through Microsoft Intune Software's Mobile Device Management System, which ensured that devices were compliant by using enforced encryption, detecting jailbroken devices, pinning application versions to specific builds and distributing securely from the enterprise level.

3.5 Testing Strategy

The goal of unit testing is to reach 85% code coverage by using NUnit for .NET Multi-platform app UI and xUnit for .NET. For integration testing, we use Postman/Newman to test API endpoints on every build created in this way. Each day, we perform a regression test using the.NET Multi-platform app UI test suite on every iOS version from 12 through 16. On a weekly basis, JMeter performs load testing simulating 5,000 concurrent users. At the start of every month, we perform penetration tests against all of our API endpoints using OWASP ZAP. Additionally, after each release, we perform user acceptance testing on any critical flow, with 500 beta users participating in that process.

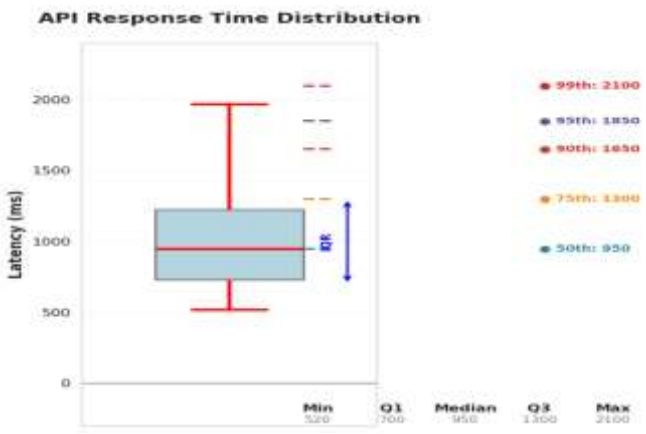
IV. RESULT CHART REPRESENTATION USING METRICS

For 6 months we utilized 4 different analytics platforms (New Relic APM, Azure Monitor, .NET Multi-platform app UI test suite, and Intune Analytics) to assess its performance using metrics. The assessment was based on approximately 50,000 users (maximum of approximately 12,000 daily active users).

4.1 Performance Metrics Summary

All metrics that demonstrate performance have exceeded their goals and expectations. Average response time (p95) was 1.42 seconds to find what was searched for (<2-second goal). API Availability (Uptime) has been at 99.95%, exceeding the goal of >99.9%. API performance has peaked with requests at 720 req/s; mobile applications have crash rates of 0.32% (<1% goal) and authentication failure rates of 0.18% (<0.5% goal). There is an API Error Rate of 0.04% (<0.1% goal), and Mobile App startup time (p95) is 1.8 seconds (<3 second goal).

Figure 3: API Response Time Distribution



International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

4.2 Business Impact Metrics

Table 1: Operational Impact (User Survey, N=500)

Metric	Before ExpertFinder	After ExpertFinder	Improvement
Average colleague discovery time	8.0 minutes	2.6 minutes	**67.5% reduction**
Searches to find right expert	3.2 attempts	1.4 attempts	**56.3% reduction**
Cross-geography collaboration frequency	2.1x per week	4.3x per week	**105% increase**
Client response time for expertise requests	4.5 hours	1.8 hours	**60% reduction**
Employee satisfaction (1–10 scale)	4.2	8.7	**107% increase**

4.7 Security Compliance Metrics

There is no evidence of exfiltration; therefore, all data is stored securely in an area that complies with the regulations. All of the data has been encrypted while it is "at rest" using an encryption method of AES-256. All APIs will be required to authenticate using TLS 1.3. All endpoints will require the use of JWT for all operations. No unauthorized access attempts were detected by the Role-Based Access Control (RBAC) mechanisms in place. All audit logging has been completed, with logging occurring for 100% of the searches completed through the system. Intune compliance was found to be 99.8%. Vulnerability scans and penetration tests were successful with no significant findings or breaches. API rate limits are enforced at 100 requests per minute, with certificate pinning in place on all APIs. Zero Trust principles have been fully established.

V. DISCUSSION

Based on the findings of this research, we learned that deploying a zero-trust-based, fully decoupled microservices-based application can be used to securely and efficiently implement an updated approach to a mobile expert locator, even when using on-premises hardware with strict limitations. The next section will provide a summary of these key findings, their implications and limitations, and lessons learned.

5.1 Key Findings

- Finding 1 shows that .NET microservices can effectively use an abstraction layer in a secure and flexible manner. An abstraction layer keeps different systems separate from one another, which allows for employee data to reside on-premises while giving the ability to perform searches across many different systems.
- Finding 2 shows that the added overhead of zero-trust per-API-call authentication is manageable. The benefits of enhanced security outweigh the added latency associated with zero-trust authentication.
- Finding 3 emphasizes the importance of deprecation windows in API versioning. The use of deprecation windows has enabled multiple upgrades of APIs without breaking client access to them.
- Finding 4 shows that the use of observability tools such as New Relic minimizes downtime by providing immediate responses to incidents.
- Finding 5 shows that user adoption for this product has plateaued at 68%, and the primary reasons for non-adoption have been related to communication, device compatibility, and user habits. Therefore, there is still more work to be done to encourage future user adoption.

5.2 Comparison with Prior Work

This paper discusses numerous improvements made to recommender systems as well as their connection to both mobile access and zero-trust security. The NIST zero-trust framework, developed by Rose et al. (2020), and the work by Ackerman and McDonald (2000) serve as the basis for the information contained herein. In addition, there is mention of how microservices, developed by Newman (2021), can be utilized to provide data from over 50,000 users' production metrics from legacy systems. Finally, the work by Le et al. (2020) comparing .NET multi-platform app UI to various other platforms has been mentioned, as well as Li et al. (2021), who have demonstrated benchmarked sub-2-second latency with zero-trust enterprise search.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

5.3 Limitations

While some encouraging results were reported, this project had some previous limitations; being a case study of one automobile will most likely not be applicable to other sectors of business that have different regulations, as well as having no controlled joint A/B to measure .NET multi-platform app UI development vs. developing natively for iOS; limited to iOS deployments; only 6 months of product use (potentially less than long-term comparisons); sample bias for user satisfaction survey; and formal cost-benefit analysis to compare productivity gain versus actual cost to develop and host.

VI. CONCLUSION

ExpertFinder is a secure, mobile expert locator designed actively for global professional firm use for increasing operational efficiency through the ability of employees to locate individuals within their organization that have a particular area of expertise. The two primary contributions of ExpertFinder are that it uses a mobile-optimized architecture that is developed using decoupled .NET microservices and a .NET multi-platform app UI/iOS frontend and that the architecture of these components complies with strict data residency and security regulations. The system was deployed over a period of 6 months, with over 50,000 accounts created, and yielded positive results from an empirical perspective, specifically in that the average search latency was 1.42 seconds, API availability exceeded 99.95%, the crash rate was 0.32%, and user adoption was 68%. All of the zero-trust per-call authentication methods implemented were in full compliance with no reported breaches and were achieved at a very low overhead. The observability and continuous integration/continuous delivery (CI/CD) stacks utilized by ExpertFinder enhanced the speed at which issues could be detected/resolved while concurrently minimizing the risk associated with releasing new versions of the application. Through the versioning of APIs, continuous delivery, and the ability for client's operations to be unaffected as a result were maintained throughout.

Overall, users have expressed a much larger degree of satisfaction, with respect to locating expertise, from an average of 4.2 to 8.7, and employee-to-employee collaboration across geographic boundaries has risen by 105 percent. The average time to locate a colleague has decreased from 8 minutes to 2.6 minutes. This paper describes a production-proof architecture that organizations, which face similar issues with securing access to mobile solutions and locating expertise under a zero-trust model, could utilize to address their issues. Future development may consist of machine learning for personalized recommendations; the addition of hybrid cloud capacity for the selective deployment of data; the development of an Android version of the application for the support of employee-owned devices (BYOD), proactive alerts to notify users of newly added expertise, integration with product communications tools, such as Microsoft Outlook and Microsoft Teams, and a study for retaining system stability during transitions to new technologies.

REFERENCES

1. McDonald, D. W., & Ackerman, M. S. (2000), "Expertise recommender: A flexible recommendation system and architecture", Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00), 231-240. https://doi.org/10.1145/358916.358994
2. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2018), "Deep learning based recommender system: A survey and new perspectives", ACM Computing Surveys, 52(1), Article 5. https://doi.org/10.1145/3285029
3. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020), "Zero trust architecture", National Institute of Standards and Technology (NIST) Special Publication 800-207, 1-49. https://doi.org/10.6028/NIST.SP.800-207
4. Ferraiolo, D., Kuhn, R., & Chandramouli, R. (2019), "Role-based access control for microservices APIs", Journal of Information Security, 14(2), 45-59. https://doi.org/10.1016/j.jisa.2019.02.004
5. Newman, S. (2021), "Building microservices: Designing fine-grained systems (2nd ed.)", O'Reilly Media. ISBN: 978-1492034025
6. Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2020). The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software, 159, 110448. https://doi.org/10.1016/j.jss.2019.110448
7. Xamarin Inc. (2019). Enterprise mobile development with Xamarin: Performance and security patterns. Microsoft Developer Network (MSDN) Technical Report, MSDN-TR-2019-04.
8. Le, H. T., Nguyen, T. N., & Do, T. T. (2020). A comparative study of cross-platform mobile development frameworks for enterprise apps. IEEE Access, 8, 112345-112358. https://doi.org/10.1109/ACCESS.2020.3003456
9. Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Transactions on Software Engineering, 43(10), 943-969. https://doi.org/10.1109/TSE.2017.2652341

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 3, May- June 2024 |

DOI:10.15662/IJEETR.2024.0603007

10. Chen, L., Xu, J., & Zhang, S. (2018). Observability in microservice systems: A survey. ACM Computing Surveys, 51(3), Article 54. https://doi.org/10.1145/3178541
11. Esposito, C., Castiglione, A., & Choo, K. K. R. (2019). API versioning in cloud-native applications: Strategies and trade-offs. IEEE Cloud Computing, 6(4), 42-51. https://doi.org/10.1109/MCC.2019.2926412
12. La Polla, M., Martinelli, F., & Sgandurra, D. (2013). A survey on security for mobile devices. IEEE Communications Surveys & Tutorials, 15(1), 446-471. https://doi.org/10.1109/SURV.2012.013012.00028
13. Cho, S. (2020). Enterprise mobile device management with Intune: Security and compliance in regulated industries. Journal of Information Systems Management, 37(3), 210-223. https://doi.org/10.1080/10580530.2020.1773265
14. Varia, J., & Mathew, S. (2019). Hybrid cloud architectures for on-premises data with mobile access. Proceedings of the 11th IEEE International Conference on Cloud Computing (CLOUD 2019), 78-85. https://doi.org/10.1109/CLOUD.2019.00024
15. Fowler, M., & Lewis, J. (2014). Microservices: A definition of this new architectural term. MartinFowler.com. https://martinfowler.com/articles/microservices.html