International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06427
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Local Service Provider

1. Mrs. Apeksha Nayak 2. Yash Verma 3. Vijay Kumar 4. Iti Bansal

1- Assistant Professor , Department of Computer Science and Engineering, Shri Ram Group Of Colleges, Dr. A.P.J. Abdul Kalam Technical University
2- Student , Department of Computer Science and Engineering, Shri Ram Group Of Colleges, Dr. A.P.J. Abdul Kalam Technical University
3- Student , Department of Computer Science and Engineering, Shri Ram Group Of Colleges, Dr. A.P.J. Abdul Kalam Technical University
4- Student , Department of Computer Science and Engineering, Shri Ram Group Of Colleges, Dr. A.P.J. Abdul Kalam Technical University

Abstract:

The unorganized local home services sector, which includes plumbing, electrical repairs, and deep cleaning, faces issues with unclear pricing, unverified professionals, and poor communication between customers and providers. To solve these problems, we present **Local Service Provider**. This is a cloud-based local service marketplace built on the Salesforce ecosystem. The unorganized local home services sector, which includes plumbing, electrical repairs, and deep cleaning, faces issues with unclear pricing, unverified professionals, and poor communication between customers and providers. To solve these problems, we present **Local Service Provider**. This is a cloud-based local service marketplace built on the Salesforce ecosystem. Unlike traditional large applications, **Local Service Provider** separates a high-performance frontend, which includes fuzzy searching, from a strong Salesforce backend. The system uses Salesforce's Web-to-Lead function for automated customer onboarding and employs asynchronous Apex processing to dynamically create restricted user accounts. To improve communication without sacrificing privacy, we developed a custom real-time chat application using Salesforce Platform Events. This paper outlines the architectural choices, how we dealt with transaction limits during automated user provisioning, and the launch of a highly scalable, secure, and user-friendly service platform. The system uses Salesforce's Web-to-Lead function for automated customer onboarding and employs asynchronous Apex processing to dynamically create restricted user accounts. To improve communication without sacrificing privacy, we developed a custom real-time chat application using Salesforce Platform Events. This paper outlines the architectural choices, how we dealt with transaction limits during automated user provisioning, and the launch of a highly scalable, secure, and user-friendly service platform. The unorganized local home services sector, which includes plumbing, electrical repairs, and deep cleaning, faces issues with unclear pricing, unverified professionals, and poor communication between customers and providers. To solve these problems, we present UrbanServe. This is a cloud-based local service marketplace built on the Salesforce ecosystem. Unlike traditional large applications, UrbanServe separates a high-performance frontend, which includes fuzzy searching, from a strong Salesforce backend. The system uses Salesforce's Web-to-Lead function for automated customer onboarding and employs asynchronous Apex processing to dynamically create restricted user accounts. To improve communication without sacrificing privacy, we developed a custom real-time chat application using Salesforce Platform Events. This paper outlines the architectural choices, how we dealt with transaction limits during automated user provisioning, and the launch of a highly scalable, secure, and user-friendly service platform.

Keywords:

On-Demand Services, Salesforce Automation, Cloud Computing, Real-Time Chat, Apex Triggers, Platform Events, Lightning Web Components (LWC)

Introduction:

The gig economy has changed how people in cities use services. Still, the local home service industry is very fragmented. Customers usually depend on word-of-mouth or unverified directories. This leads to security issues, inconsistent pricing,

and low-quality service. On the service provider side, they often don't have a centralized Customer Relationship Management (CRM) tool. This makes it hard for them to manage their leads, schedules, and client communications.

Although there are some market aggregators, they often use rigid, hardcoded systems that are tough to scale or adjust to new business needs. To address this issue, **Local Service Provider** was created to combine the flexibility of a modern web interface with the reliability of Salesforce. By using Salesforce as an active automation engine rather than just a database, the platform reduces manual administrative tasks. The platform also features a custom-built, real-time messaging interface that keeps customer-provider interactions secure, documented, and instant.

Problem Statement:

The current local service booking system faces three main problems. First, there is a large gap between generating leads and onboarding users. When a customer asks for a service, it usually takes a lot of manual work to set up an account, verify information, and assign a professional. Second, search engines on these platforms are often not well-optimized. This leads to frustrating experiences for users searching for specific service categories. Third, communication after booking often relies on outside channels, such as WhatsApp or direct phone calls, which breaches user privacy and leaves the platform without a record in case of disputes. We need a single system that manages lead conversion, secure user setup, and real-time communication all in one transaction process.

Objectives:

The main objectives of the **Local Service Provider** platform are:

- **To build a responsive and easy-to-use single page** interface where users can quickly search for services. The search feature is designed to help customers find the required service smoothly from more than 40 available service categories.
- **To make the customer registration process simpler by using Salesforce** Web-to-Lead forms, so that service requests submitted by users can automatically be converted into customer records without the need for manual data entry.
- **To maintain system security by implementing a role-based access** mechanism, ensuring that new customers can only access the features and information that are relevant to them.
- **To provide a built-in communication system by developing a real-time chat** feature using Apex and Platform Events. This allows customers and their assigned service providers to communicate instantly and securely, and also enables file sharing when required.

Methodology:

The development of the **Local Service Provider** platform focuses on transforming the traditional service booking process into a more organized and automated cloud-based system. Earlier, the workflow was mostly manual and scattered across different tools, which made it slow and difficult to manage. By introducing a structured platform built on modern web technologies and Salesforce, the system aims to improve efficiency, reduce manual work, and create a smoother experience for both customers and service providers.

A. Existing System Vulnerabilities

In many traditional setups, customer service requests are handled through basic methods such as shared email inboxes or Excel spreadsheets. When a customer sends a request, an administrator usually has to manually check the message, record the details, contact the customer for confirmation, and then share the service provider’s contact information.

This process involves several manual steps, which often leads to delays in response time. In addition, maintaining data across multiple files or emails can create duplication and confusion. As the number of service requests increases, managing these tasks becomes even more difficult and inefficient.

B. Proposed System Architecture

The **Local Service Provider** platform is designed around three main components that work together to automate and simplify the service booking process.

1. Frontend Search Engine and User Interface

The user interface of the platform is developed using **HTML5, CSS3, and JavaScript**. The goal was to create a clean and responsive design that allows users to easily navigate through different services.
A dynamic **Mega Menu** and a predictive search bar are implemented to help users quickly find the service they need. To improve performance and avoid unnecessary server requests, a **300ms debounce function** is used in the search feature. This means the system waits briefly while the user is typing before processing the search query. Along with this, a **smart substring matching technique** is applied so that even partial search terms can return relevant results. For example, when a user types a keyword like "clean", the system can immediately suggest services such as *Deep Cleaning* or *Sofa Cleaning* without repeatedly querying the database.

2. Automated Lead Conversion Engine (Apex)

When a customer submits a service request through the website, the information is sent to Salesforce using a **Web-to-Lead HTTP POST request**. This allows the system to capture customer inquiries directly within the Salesforce environment.

One of the technical challenges during development was handling the **MIXED_DML_OPERATION** issue. This error occurs when operations on standard objects (such as Lead or Contact) and setup objects (such as User) are attempted in the same transaction.

To handle this situation, an **asynchronous Apex service class** was implemented using the **@future annotation**. A trigger first captures the incoming lead and sends its ID to this background process. The service then converts the lead into a customer record, generates a unique username, creates a user account with a restricted **Chatter App Only** profile, and finally sends an automated email containing login credentials. This automated process removes the need for manual account creation and ensures faster onboarding.

3. Real-Time Communication Module

To enable smooth communication between customers and service providers, a custom chat system was developed instead of using the standard Salesforce communication tools. The system uses an **Apex controller class (ChatController)** to manage message creation and storage within the database.
For real-time message updates, **Salesforce Platform Events** are used through the **EventBus mechanism**. Whenever a new message is sent, a custom event (**ChatEvent__e**) is published. The frontend interface subscribes to this event, allowing the message to appear instantly on the user's screen without needing to refresh the page.
The communication module also supports **file sharing**, which is handled using **ContentDocumentLink**, allowing users to attach and exchange files within the chat. Additionally, message control features such as **"Delete for Everyone"** and **"Delete for Me"** are included to give users better control over their conversations.

Results & Discussion:

The implementation of the Local Service Provider platform showed clear improvements compared to traditional service booking methods. The optimized search feature made browsing faster and smoother for users. By reducing unnecessary processing in the browser, the platform was able to provide a more responsive and seamless experience while users searched for different services.
On the backend, the automated user creation process simplified the overall workflow.

Customer requests submitted through the platform were automatically processed and converted into user accounts without requiring manual intervention. Separating these operations using asynchronous processing helped prevent system conflicts and ensured that the platform continued to run reliably even when handling multiple requests.
The platform also introduced a custom real-time chat system that allows customers and service providers to communicate directly within the application. Messages and attachments can be shared quickly and securely, which helps both parties discuss service details without relying on external messaging apps. Since all conversations remain within the system, they can be easily tracked when needed.
Security and data privacy were also given significant importance in the system design.

To ensure controlled access, a **Role-Based Access Control (RBAC)** approach was implemented using Salesforce profiles. When a new customer account is created, the user is automatically assigned limited permissions. This means customers can only access features that are relevant to them, such as the chat interface and their own service or booking details. They are not able to view or interact with data belonging to other users. This structured access control helps maintain data separation within the platform and ensures that information remains secure.

Conclusion:

The Local Service Provider platform shows how enterprise **CRM technologies** can be used to solve real problems in the local service and gig-economy sector. By combining a simple, user-friendly frontend with the powerful capabilities of Salesforce Apex, the system reduces a large amount of manual administrative work that was previously required.

The automated process—from submitting a **Web-to-Lead** request to creating a user account and enabling real-time communication—makes the entire service workflow faster

and more organized. In the future, the platform can be further improved by adding AI-based sentiment analysis in the chat system to identify dissatisfied customers early, along with the development of a mobile application using the **Salesforce Mobile SDK** to make the platform more accessible.

References:

- Salesforce Developers, "Future Methods and Asynchronous Processing in Apex," Salesforce Official Documentation, 2024. [Online]. Available: developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_invoking_future_methods.htm
- Salesforce Developers, "Platform Events Developer Guide: Event-Driven Architecture," Salesforce Official Documentation, 2024. [Online]. Available: developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm
- M. Fowler, "Patterns of Enterprise Application Architecture: Decoupling Frontend and Backend," Addison-Wesley Professional, 2021. [Online]. Available: martinfowler.com/eaaCatalog/
- Mozilla Developer Network (MDN), "Optimizing Web Performance: Implementations of Debouncing and Throttling," Web Technology Documentation, 2023. [Online]. Available: developer.mozilla.org/en-US/docs/Web/API/setTimeout
- Salesforce Help, "Guidelines for Setting Up Web-to-Lead," Salesforce Administration Guide, 2024. [Online]. Available: help.salesforce.com/s/articleView?id=sf.setting_up_web-to-lead.htm&type=5
- Salesforce Developers, "Lightning Web Components (LWC) Reference," Salesforce Component Library, 2024. [Online]. Available: developer.salesforce.com/docs/component-library/overview/components
- S. Kumar and R. Sharma, "Impact of Cloud CRM on the Gig Economy and Local Service Aggregators," International Journal of Cloud Computing, vol. 12, no. 3, pp. 45-58, 2023. [Online]. Available: www.inderscience.com/jhome.php?jcode=ijcc