International Journal of Advanced Development in Science and Technology
Volume : 3 Issue : 03 April 2021 www.ijadst.com e-ISSN: 2582-1059

MULTITENANT GENERIC SOFTWARE AS A SERVICE OMNI CHANNEL E-COMMERCE APPLICATION

[1] A.Christen, [2] R.Nikhil, [3] K.Nattarkannan, [4] R.K.Kapilavani,
[1,2] Students, [3,4] Faculty,
[1,2,3,4] Prince Shri Venkateshwara Padmavathy Engineering College, Ponmar,
[1] antonychristen1@gmail.com

ABSTRACT

Unlike traditional web applications, cloud-based applications can provide services for large number of tenants using the limited hardware and software by implementing multitenant architectures. Software as a service (SaaS), because the top model of cloud computing, has assisted the industry to develop different multi-tenant applications. In SaaS, the software portion lies outside the physical portion of the organization. Organizations buy the services of the software instead of buying the software itself. The software application is delivered and managed remotely by the SaaS providers. Multi-tenant could be a key characteristic for cost effective Software as a Service (SaaS) applications which drive down total cost of ownership for both service consumers and providers.

This paper describes the designing and building of a cost effective, ease of maintenance, generic, flexible, scalable, cost-effective, data isolated, customizable, Omni-Channel E-Commerce application with multitenant database and micro services which might greatly accelerate the localized businesses to migrate towards the trending technology.

Keywords – Software as a Service, Omni-Channel, Point of Sale, E-commerce, Multi-tenant.

1. INTRODUCTION

Starting a business with a website is an aspiration for many localized businesses. Almost every business use Basic Single PC Point of Sale Software for in-store billing which costs a maximum of around Rs.30K and a customized web application costs around Rs.2-4L and further with increase in requirements, which is a long shot for those who make a turn-over of Rs.30K to Rs.50K per month. Few cost-efficient website building sites help users to create website of their will just by drag and drop, but provide only static contents with their default templates and have limited features, hence for a customized solution, the cost will be very high. Therefore, for a customized dynamic application, it would cost a minimum of around Rs.2L for the software and high subscription cost for a cloud based application.

The proposed concept in simple words is, Generic SaaS POS Application that manages both in-store and online sales in a single integrated channel using the concepts of multi-tenancy and cloud (i.e.) a Single Server is used for managing all the tenants which will result in cost reduction for end users and a high profit for the SaaS business. As in the proposal, since the tenant allocation is subscription based, from the end users' perspective it is possible to provide entire solution for just Rs.10K or above depending on their subscription and with increase in tenants the development and deployment cost can be covered and huge profit can be achieved.

2. RELATED WORK

Multi-tenantE-commerce supported SaaS Provider needs to purchase just one domain. Each of the tenants gets sub domain of his/her choice. For example, Domain is appdomain.com and therefore the sub domains for tenant1, tenant2 and tenant3 are abc, xyz and 123 respectively. Each store (tenant) is related to store id (realmId) thatuniquely identifies eachstore. The service is provided based on the Pay-as-you-go revenue model. This model reduces IT cost (i.e.) the vendors (tenants) wouldn't be charged any fixed amount instead transaction basis pay per use.

In some existing approach a novel, cloud-native architecture for personalization, i.e., customizing mutely-tenant SaaS by micro services is explained. A customization for a selected tenant is running as a standalone micro service and dynamically registered to the merchandise service for this tenant. At runtime, the customization micro services are triggered by the merchandise service when the latter reaches a registered extension point. They interact with one another via REST APIs. The customization micro services are hosted by the same cloud vendor as the product service. With the correct technical choices, such micro service-based architecture includes a good balance among isolation, assimilation and economy of scale.

3. SYSTEM MODEL

Omni-channel point of sale is an in-trend best approach in which both in-store and online sales operations should be running off one integrated channel. That means, one platform to manage your stock levels, pricing, customers and other crucial data. It is hot business as it makes businesses accessible 24/7 to the customers. This approach mainly focus on providing seamless customer experience whether the client is shopping online from a mobile device, a laptop or in-store. This paper presents a generic Omni-channel application to target mainly the localized businesses, to pull them into the current trend and provide services at a very minimal cost, thereby achieving high business profits. At present, the applications available are either organization-specific (who have own data centres) or profession-specific (specially designed only for drug stores,etc.) This paper makes possible a flexible application with omni-channeling that can be used by any people in combination with the powers of multitenancy and micro services, thus making it a real time accomplishment .The subsequent set of modules includes:

- Creation of Database Schema
- Creation of Spring boot Micro service
- Development of Omni-Channel Point of Sale Application
- Development of Simple E-commerce Application
- Deployment of micro service and applications to Cloud Servers

A. CREATION OF DATABASE SCHEMA

Firstly a database MySQL server is created. Here, Shared Database Schema model database is created. In the database a "tenant" table is created which contains columns which has the 'tenant id'. This tenant id plays a crucial role in this architecture. That is, it helps to access only the rows and columns of the respective tenant, thereby achieving data isolation. Each tenant is given a separate tenant ID which works exclusively for accessing and verification of data while firing each and every query. It is a cost effective approach to save resources and highly profitable business database model. As we know each tenant will have 2 or more tables merging these tables into a single table will depend upon tenant requirement and may vary from tenant to tenant. If one tenant send any query to database with its associated tenant id, before processing the query, database will check tenant id provided already exists or not. If yes, then it will match the id and create a cache list of allowed rows and column (allowed column are those columns in which tenant has permission to read and write) and then it will match the column to access column list and proceed the task.

B. CREATION OF SPRING BOOT MICROSERVICE

The multitenant micro service logic is developed in such a way that it handles both the tenant and their client requests simultaneously from 'n' number of tenants and provide the desired data for each and every request. At the same time, it is provide access to data only to authorized tenants and customers without data collapse. Hence filtration security methods application level and encryption techniques play significant roles. Initially, both tenant and client are routed to their respective domains by automatic selection using sub domain method. This is the most accessible type of multi-tenancy to manage, but it is an advanced feature. Here, when the user wants to log into the application, they simply open the page in their browser and authenticate. Users do not see the entire process of authentication going on in the background. In the background, the entered credentials are mapped with corresponding sub domains and are checked for matching, only then the authenticated is completed. After secured login, a Java Web Token is created for each individual logins. Now for each http requests to the micro service, two level verification takes place: token and tenant id verification

Filter-based pattern filtration, the application level filter to restrict the data extracted from database in each request of tenants. For share table/schema isolation pattern, the filter is based on the tenant ID column in every table to access records associated with the appropriate tenant, e.g. modifying a SQL statement with where clause 'tenantID=XXX'.

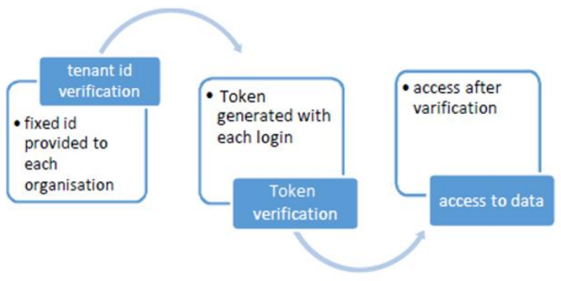

tenant id verification
• fixed id provided to each organisation
• Token generated with each login
Token verification
• access after varification
access to data

Fig 1 : Two Step Verification

For securing confidential data at the application level from and to both the interfaces, RSA (Rivest–Shamir–Adleman) encryption and decryption algorithm is used. For securing confidential data at the database level from and to the micro service, bcrypt encryption and decryption algorithm is used. Permission-based/Role-based pattern at the application level in each employees of the tenant is assigned an access account which only has privileges to access few features of the tenant resources.

In short, database is searched to get current tenant identity at the runtime which helps to get tenant's sharing model and security model configurations, also the runtime service will help routing and enforce the security mechanism for tenant's data access request and their corresponding client requests.

C. DEVELOPMENT OF OMNI-CHANNEL POINT OF SALE APPLICATION

This module is the user interface provided to the tenants. This interface consists of modules several modules such as dashboard, employee management, customer management, sales management, product management, point of sale for billing, etc. The dashboard pictures the sales took place for a period of time. The employee management provides the opportunity to grant permissions to them based on their roles and provides few other additional functionalities.

The customer management manages the online registered customers. In product management, one can add/remove/update products according to brands, categories or product itself. Administrator is responsible for giving details such as product image, description, price, etc. that will directly reflect store front (client interface). Similarly order management provides detailed report of both in-store and online sales. The above modules are individual services provided to the tenants depending on the subscription. The auto-scheduled and manual database back up option is integrated with the interface for to avoid loss of data at any situation. Figure 2 explains in clear the integrated working of tenant and their customer interfaces.

The proposed system is based on multi-tenant SaaS model which makes it possible to have multiple online stores from the single code base and database. In product catalog manager administrator can add/remove products according to brands, categories or product itself. Administrator is responsible for giving details such as product image, description, price, etc. that will directly reflect store front. Similarly order management will be done by store administrator.

D. CREATION OF SIMPLE E-COMMERCE APPLICATION

This module is the e-commerce application operated in a single channel along with the tenant interface. This e-commerce is provided as service to multiple vendors (tenants) as Tenant 1 to Tenant 'N'.With the help of this system, vendors can have their own online store opened instantly. The system provides eachvendor (tenant) with separate store front and store backend (POS Frontend) and their data is stored in thedatabase schema created. For launching an online store vendor first needs to register her/himself as store administrator, then administrator of the store will beprovided with different managements such as customermanagement, product catalog management, order management, coupons management, media management and many more features in the front end of point of sale application itself for both in-store and online sales.

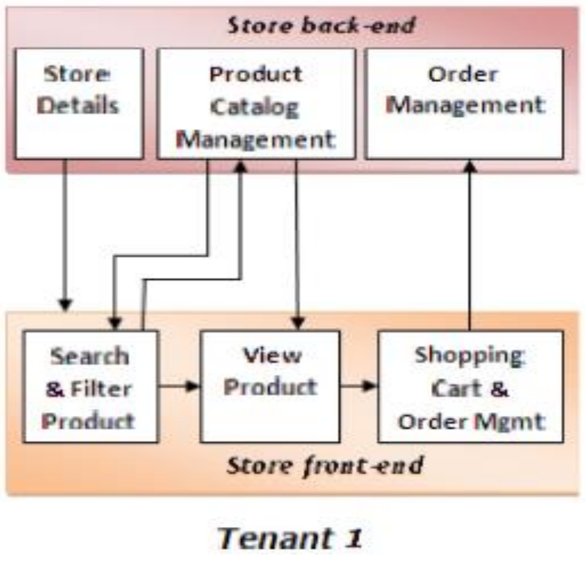


Fig 2 : Tenant's Client UI & Backend Interaction

E. DEPLOYMENT OF MICROSERVICE AND APPLICATIONS TO CLOUD SERVERS

In this module, the micro service and the user interfaces are deployed onto the cloud, so that Software itself can be provided as a Sevice, thus the goal business model SaaS. The micro service along with the database is deployed on to the Google Cloud Platform so that the server scales with as the number of requests and responses increase. The tenants and their e-commerce sites are hosted on GoogleFirebase with distinct domains or sub domains based on their subscription.Thus, Single Codebase and Database support multiple tenants.

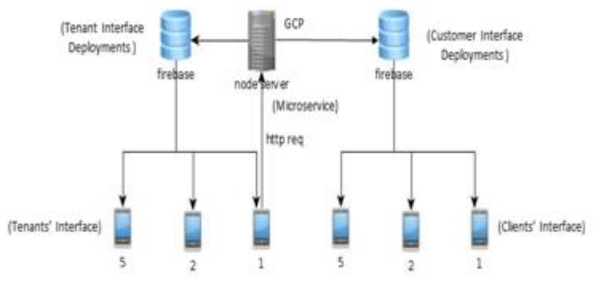


Fig3 : Deployment Interaction

4. RESULTS

The system model explained as above is implemented in real time and gained immense responses from wide variety of localized businesses. It gained positive feedbacks and took the suggestions from real time users as requirements. The snapshots from the user interfaces are presented below for a clear view.

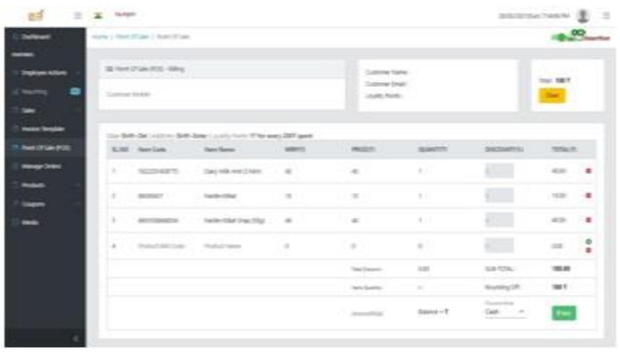


Fig 4. Point of Scale

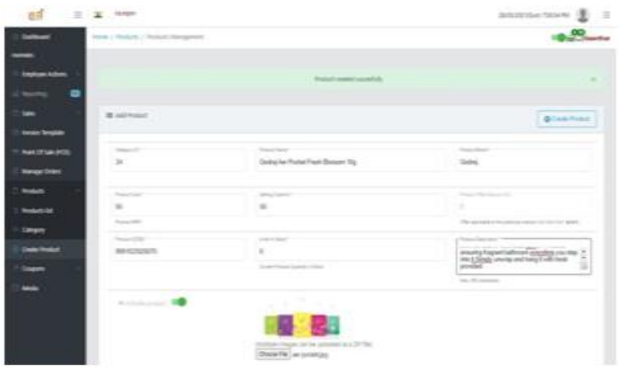


Fig 5: Product Creation

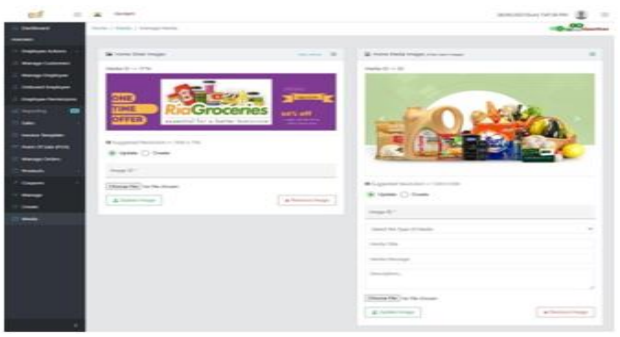


Fig 6: Client UI Banner Change Option in Admin UI

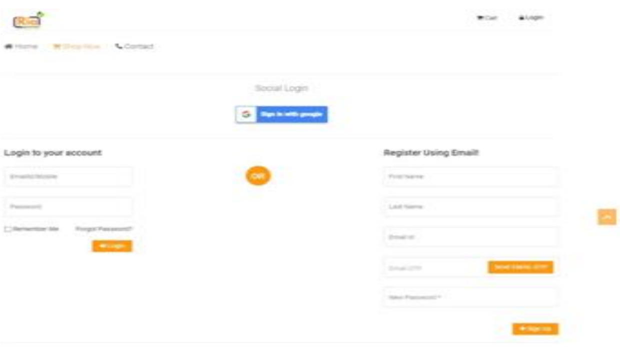
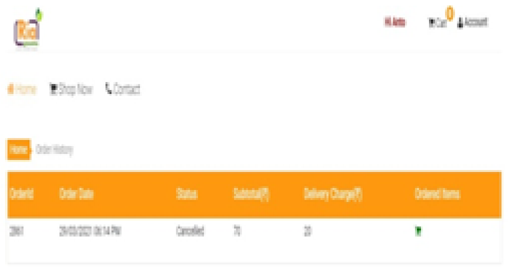


Fig 7: Sign in/Sign up

Fig 8: Add Cart



OrderId	Order Date	Status	Subtotal(₹)	Delivery Charge(₹)	Ordered Items
2961	29/03/2021 04:14 PM	Cancelled	70	20	

Fig 9: Order Status

5. CONCLUSION

In recent years, native multi-tenancy model, as exemplified in SaaS (Software as a Service), achieves great successes. Pioneers in this domain started building their solutions of hosted applications around the multi-tenancy rather than simply leveraging the on-premise application hosting model. Taking everything into account, a multi-tenant SaaS architecture offers more long-term benefits both in terms of development and investment than a single-tenant one.

It gives excellent data capacity and a higher ceiling for businesses. That is why most SaaS services operate on multi-tenancy. In comparison to a single-tenant architecture, it is cheaper, it is easier to maintain and update and makes use of resources more efficiently. What is more, multi-tenant software is secure and easily scalable. Thus, we make high business profits from our customers affordable investment, thereby providing high levelservices.

6. REFERENCES

[1] Bo Gao, Wen Hao An, Xi Sun, Zhi Hu Wang, Liya Fan, Chang JieGuo, A Non-intrusive Multi-tenant Database for Large Scale SaaS Applications,WeiSun IBM China Research Lab, Beijing, China,8th IEEE International Conference on e-Business Engineering, 2011

[2] Keshav Gupta, Sandeep Kumar, OjaswiAgnihotri, Data isolation in multi-tenant SaaS environment, School of Computing Science andEngineering, Galgotias University, Greater Noida, IEEE International Conference on Computing Communication and Automation,2016

[3] Huixin Chen, Architecture Strategies and Data Models ofSoftware as a Service: A Review, Library, Party School of the Shandong Provincial Committee of the CPC, Jinan, Shandong, China, 3rd International Conference on Informative and Cybernetics for Computational SocialSystems, 2016

[4] Farzana Shaikh, Dipti Patil, Multi-Tenant E-commerce based on SaaS Model to Minimize IT Cost, Department of Information Technology, Pillai Institute of Technology, Panvel, Maharashtra, India, IEEE International Conference on Advances in Engineering & Technology Research, 2014

[5] Hui Song, Phu H. Nguyen, Franck Chauvel, Jens Glattetre, Thomas Schjerpen, Customizing Multi-Tenant SaaS by Microservices: A Reference Architecture, IEEE International Conference on Web Services, 2019

[6] Deng Zhonghua, HaiErfeng, &quot;Analysis of Sa as-Based E-Commerce Platform&quot;. IEEE 20 I 0 International Conference on E-Business and E- Government, pp. 9 - 12,7-9 May 2010.