Application Development Approach to Transform Traditional Web Application into a SaaS Model

Ruth Baptista, Deven Shah

Abstract - *With emergence of new technologies, enormous amount of applications are developed, Software as a Service (SaaS) is bringing new revolution to IT industry. SaaS has changed the way for Software development, deployment and maintenance. It enables users to use the software as a solution on the internet itself, there by excluding the prerequisite that are needed to download and install the application on user's own machine thus eliminating burden of maintenance, ongoing operation, support and machine load. Multi-tenancy for SaaS application is the most important feature which contributes for its success.*

This development approach consists of features that include authentication, authorization, tenant tracking, application platform controller and securing tenant data in shared schema. Any user who is willing to access and modify the existing web application for their own use can do so by registering and will be provided with unique URL specific to that user. The proposed website will be offering a service to various dentist who desire to use the web site on a pay per use basis.

Key Words: Design pattern, Tenants, Multi-tenancy, Software-as-a-service, User intervention.

1. INTRODUCTION

Software as a Service (SaaS) is an emerging business model in the software industry due to its advantages of flexibility, quick deployment, and scalability. Web is responsible for this emerging new technologies and models. The introduction of web and its current form has seen various phases. Broadly, these phases have been classified into three groups namely Web 1.0, 2.0 and 3.0. Web 1.0 is also associated with the era of static websites, During Web 2.0 stage, the websites grew in terms of interaction capabilities and Web 3.0 can be categorized as a "read- write-execute" web.

In last decade web has become a delivery platform for many of the software products. This has led the path for innovating new web based development application of software. Software as a Service is a perfect fit to the basic needs of new software industry requirement. Recently more enterprises have been attracted to build or upgrade their applications or services from local infrastructure to cloud. Multi-tenancy is a core concept in SaaS. In SaaS model, software applications usually are adopted by using a multi-tenant architecture, that is, a single application can serve multiple customers or tenants at same time.

Traditional three tier Web Architecture is a distinctive system where a web database application works around all the 3 tiers of the model. This inclusive 3 tier architecture module is the framework for most of the Web Applications on the Internet.[1] This architecture of the system helps to separate the Business Logic from the Application, Data Storage and database.

1.1 Design Idea

When we migrate the non-SaaS applications to a SaaS application, there will be certain issues that needs to be focused on, such as architecture, database partitioning, UI customization, scalability issues, and work-flow management.[1] The idea for migrating a three tier application into a saas application is surfaced from design pattern. In a software design pattern it is a generally a reusable solution which we use to a commonly occurring drawback or bugs in a given frame for a software design. It is not a finished design which can be transformed directly into the source. It is a template or description to how the problem can be solved which can be used in many different scenarios.

The application hosting can offer three ways by which we can serve the requirement i.e use of a dedicated VM, paas architecture and a saas architecture. To over come the disadvantages of the dedicated VM and paas we use saas model for our application. A dedicated VM has limitation of performance and in paas architecture we need to maintain our hosted application however on the other hand while using a saas architecture we need not maintain any of the application related code, and the application is ready to go as soon as you get your login and password. Therefore, all software and hardware are provided and managed by a vendor, so you don't need to install or configure anything.

1.2 Traditional three tier architecture

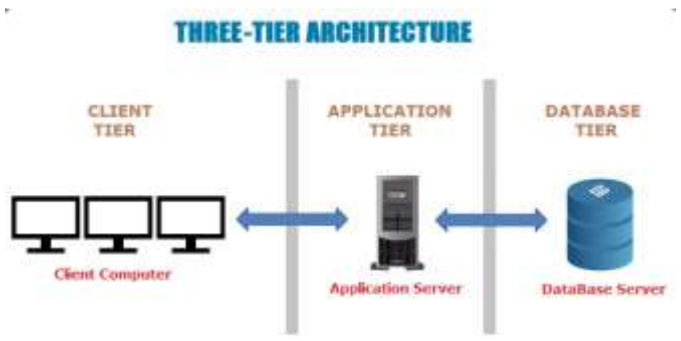

Figure1: Traditional three tier architecture

A3-tier architecture is asoftware architecture comprises of three “tiers” or “layers” for computing. This type of architecture are often used in applications for a certain type of client-server model. 3-tier architecture deliver many benefits in production and development environments by providing more modularity to the user interface, business logic, and data storage tiers. By providing above benefits it gives greater flexibility for development by allowing them to update a specific part of an application independently from the other parts. With this flexibility one can improve overall time-to-market and decrease development cycle time.

2. Literature Review

There is huge amount of research that is being carried out since past decades for SaaS and multi-tenancy. But there are only a few who proposed a complete framework for migration of traditional web applications into SaaS application. The below literature survey analyzes the work done by various researchers and scholars for Saas and Multi-tenancy.

In paper [1], authors Eyad Saleh, Nuhad Shaabani, and Christoph Meinel have proposed a framework for re-engineering a traditional web applications from scratch into saas which requires tremendous efforts in terms of cost, manpower, and time. This paper provides a framework to migrate traditional web applications into multi-tenant SaaS applications.

In paper [2], the authors Jianbo Zheng and Weichang Du explores the whereabouts of many well used conventional software applications, especially client-server applications. Reusing these application in cloud platforms will benefit both enterprises and the customers.

In paper[3], Chang Jie Guo, Wei Sun, Ying Huang, Zhi Hu Wang, Bo Gao have provided a framework where a set of multi-tenancy common services to help people design and implement a high quality native multi-tenant application more efficiently. Requirements and challenges for a native multi-tenancy pattern which have the potential of serving a large volume of clients simultaneously are explored.

In paper[4], authors Manoj A. Thomas, Richard T. Redmond, H. Roland Weistroffer state, revisions to the conventional architecture model are suggested and two examples of information systems applications. A traditional three-tier client-server architecture requires a major remodeling to address the continuously changing and rapidly increasing information processing and services needs of consumers.

In paper [5],authors summarize Nadir K.Salih, Tianyi Zang, A new model for saas application is presented with layers showing the associations and the dependencies of the layer elements. The necessity of sharing the workflow in each level and how it can improve efficiency and better control for customer service.

In paper [6],authors Ekasari Nugraheni have re-engineered web-based application SIMA into the SaaS model which can be used by many government agencies for administering state-owned inventory of the local government in Indonesia, so they do not need to build network infrastructure and provide hardware.

3. Proposed System

The Proposed system is a SaaS based Web Site for dentist where they do not need to build their own website, instead they use existing website to login or register and then can use it as their own website. Thus, the system provides a common application start-up UI. The dentist can either register or login into the application/website. If it is a new doctor who wants to register, the doctor has to register by providing the application with a logo which they wish to appear on their web-page. Once the doctor registers the they will be provided with a unique URL which they can use. Here the users are dentist, where they can store and view their patients, inventory, appointments, etc data that can be easily managed. The doctors will be charged on the basis of a pure utility based model along with other cloud based facilities. The website would offer UI customization and configuration such as name, logo, etc. for the tenants or the doctors.

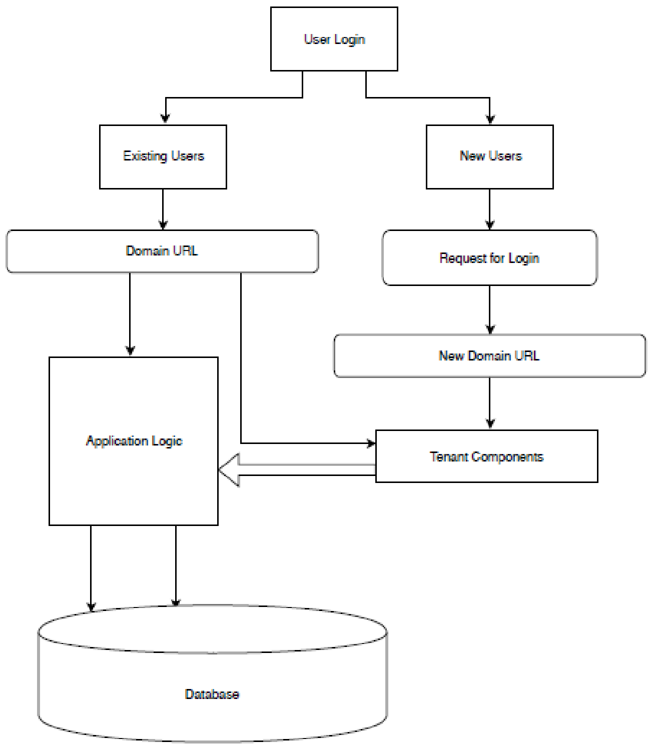

Figure2: Proposed System

4. Design Methodology

The intended system's presentation layer is same as the 3-tier architecture with a little modification in the presentation layer which includes the idea from design patterns. This layer is presented in such a way that it will appear with a wrapper on the top which will be asking for the details of the doctor. Once the details for the doctor are accepted, these details then will be the UI for application where it will change to the name and logo of that particular doctor, thus giving the look and feel of using their own website to the doctors. The doctor(dentist) when registered will be provided with a unique url where they can update and add patients data. This unique url can then be provided to the patients of their respective doctors to register or book a appointment.

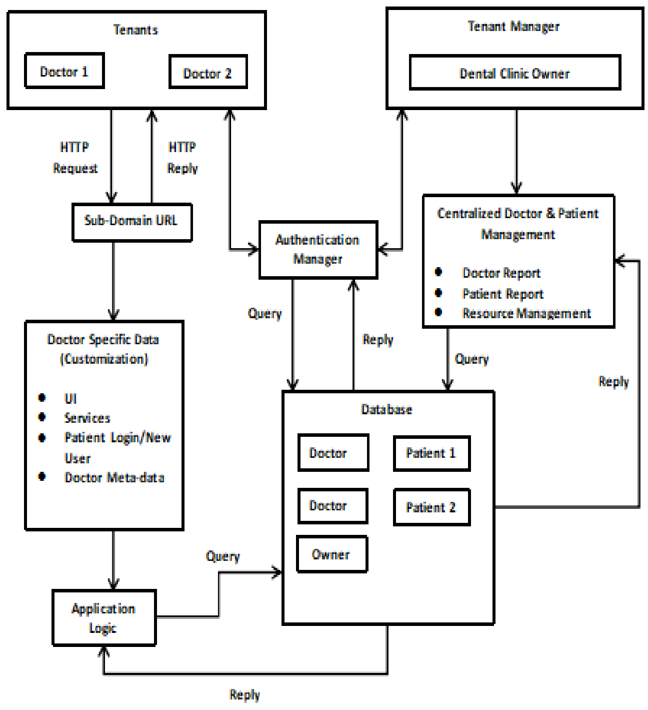

Figure3: Architecture Approach

This multitenant application is devloped by using HTML, CSS for the frontend i.e the UI and Python-django for backend i.e the database. The application can be deployed to AWS (Cloud Platform) and can be accessed by users of various tenants over the internet by using web browsers. The above diagram shows us the application architecture for the multitenant SaaS application on cloud environment. The architecture is divided into five main parts. Each of these parts is devloped/designed to support multitenant SaaS application.

4.1 Tenant Manager

In multitenant environment, Tenant manager is the one who usually shares their basic application on the internet and makes it available to the worldwide public. An application needs to have some levels of self-service sign up, even if it's just a request mechanism needed in a business process for the addition of a tenant.

4.2 Customization Component

For any multitenant application developed, configuration or customization per tenant is the most important module of the multitenant applications. Our application includes various components like logo for the doctor, name, address, etc. These components are scoped according to tenants and identified using tenant_id in the database. In our application doctors are the main configurable components. Each tenant user (doctor) is allowed to access configuration components of the system. The user can access the components that belongs to its own tenant only and is unable see other tenant's components. The tenant metadata includes the tenant_id, name of the tenant and subdomain.

4.3 Subdomain URL

Generally, a simple web application URL recognize the address of the application where it is stored. In this case subdomains play a vital role to identify the component of a main application. These subdomains are handled by DNS servers to locate the server where actual application is stored. In this case the subdomain is a parameter of the tenant which is used to locate the registration of a particular tenant. This identifies the tenant_id of a tenant and all operations in this store are carried out according to the application logic. This system sustains multiple tenants with multiple subdomains for a single domain.

4.4 Application Logic

The application logic is a main module to execute the application. This module mainly contains the real code for application, which is being used by each tenant and its users from login to the logout process. The application logic includes the functionality for a dental application. The module is developed using Python-django framework. The application logic continuously running on the server in the cloud environment. It accepts all the requests (http) from users who are using the application and gives the response with html files.

4.5 Database

Database is the core part for each application where data storage for application is done. For handling data of each tenant has some issues related to storage and access. However particularly for multitenant databases Li Heng, Yang Dan in [17] discussed three approaches by which we can implement database in multitenancy, they are separate databases, shared database - separate schema (semi-multitenancy) and finally shared database - shared schema (pure multitenancy). For our implementation tenant_id is shared by all configurable components tables and they are accessed by identifying tenant_id from the subdomain of the tenant. This pure multitenancy scheme helps to isolates the data of each tenant.

5. Result

The output for the proposed system of multitenant application can seen in below snapshots of various tenants and comparing them. We have taken results by running our application on local environment. The environment can also be hosted on the AWS environment where it can be accessible to all the public. In case of single tenant application, it has stand alone UI for the single user.

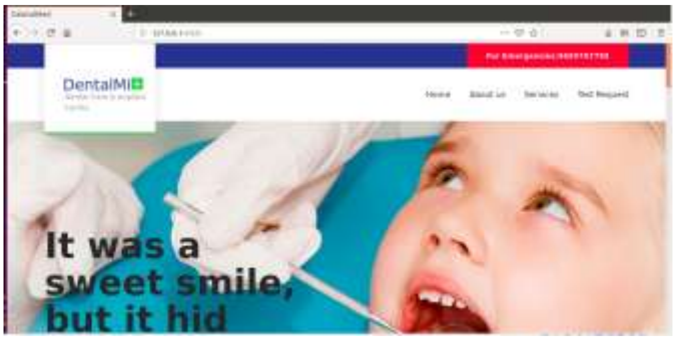

Figure 4: Application before multi-tenancy

The above image shows us the web application before converting it to the multi-tenant application.

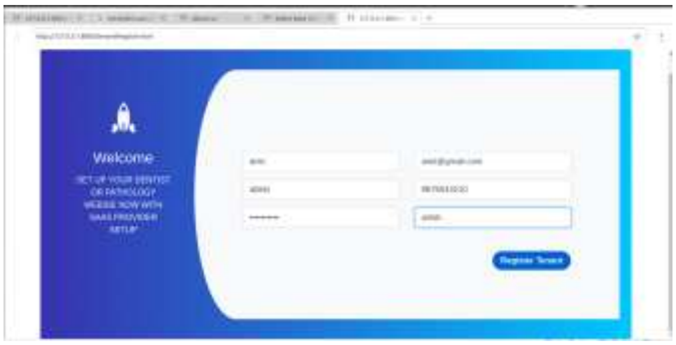
Welcome
SET UP YOUR DENTIST OR PATHOLOGY WEBSITE NOW WITH SAAS PROVIDER SETUP
Register Tenant

Figure 5: Tenant registration for website

Once the application has been transformed for multi tenancy, many users can register to use the application. The above screenshot is where the new tenant (user) can register

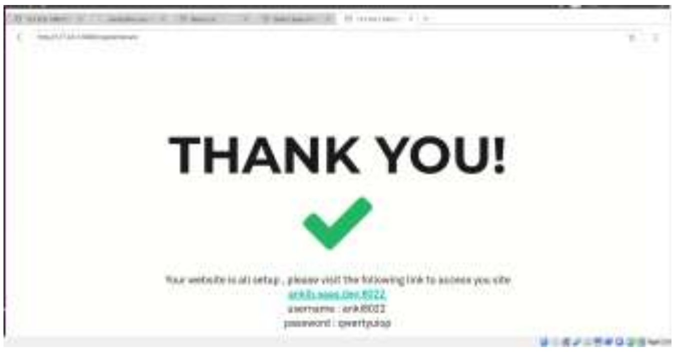

Figure 6: Tenant after registration.

The registered tenant gets the user specific url, username and password.

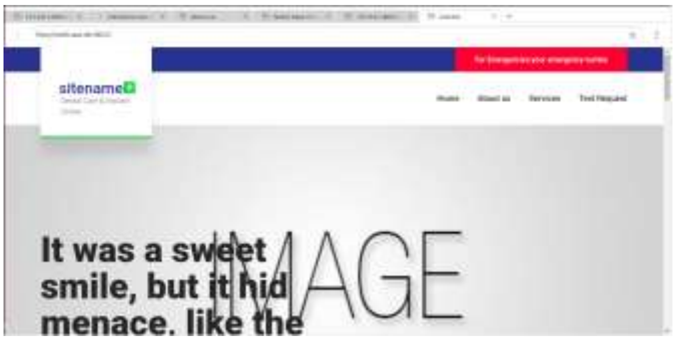

Figure 7: Tenant specific website after the registration.

The above screenshot demonstrates the view of the application after a new user has registered.

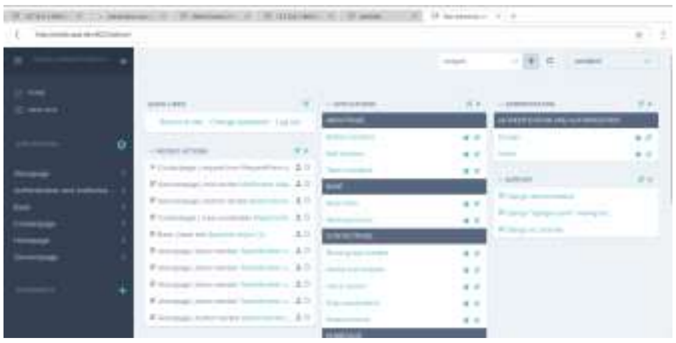

Figure 8: Tenant after registration.

The admin control panel of the specific website, where all the UI and data can be controlled. Here we can edit the images and the contact info for a particular tenant.

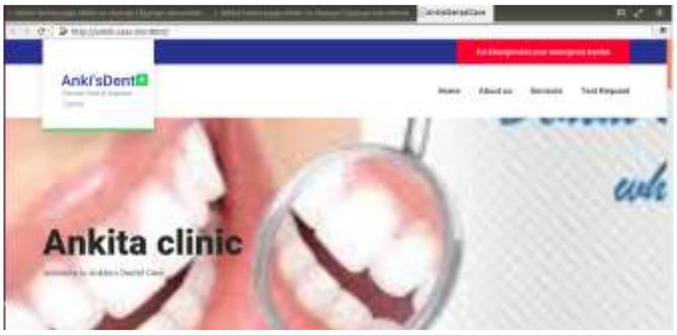

Figure 9: Tenant after registration.

The final website of the newly registered tenant with its unique URL and displaying its own name and images as required.

Conclusion

The advancement in IT industry for SaaS is ongoing and has already started to cause foremost changes in the setting. Cloud computing is emerging at far greater rates than any IT wave and SaaS has been the driver for that growth. The main concept of using software as a service is multi-tenancy. The paper suggests a approach for the multi-tenant SaaS application implementation it also describes the procedure for the implementation of multitenant SaaS application.

This approach identifies tenant from subdomain URL, providing configuration in multitenancy and most important a data separation for each tenant in cloud environment. This architecture can be used to implement any other multitenant SaaS application. Developing a new web application from scratch is more easy then re-engineering the already existing web application, doing such change requires lot of time, money and man-power. It is difficult to re configure the entire present system.

7. Future Work

The architecture can be modified as per the needs of an application that is to be implemented. Following with a dashboard that can be designed to make the tenant able to choose the configurable components for his applications in multitenant application. This will make the multitenant application highly configurable. There can also be various alterations to the SaaS project to make it highly scalable, increase performance and depending on the sensitivity of the data which will be stored various database approaches could be adopted.

This work can also be extended with the idea of developing SaaS applications to provide better service to clients. Endeavoring to help others come up with their own SaaS business where existing traditional applications can be re-engineered and given a face lift to generate monetary advances. The SaaS model can be applied in a wide-range of domains and there is endless prospect in terms of SaaS businesses. As SaaS gives the companies an alternative, where they can login and subscribe for services built on shared infrastructure over the Internet. The SaaS models are yet to develop to a greater extend in few years as it offers many benefits to the businesses.

REFERENCES

[1] Eyad Saleh, Nuhad Shaabani, and Christoph Meinel "A Framework To Convert Traditional Application To SaaS Model", Hasso-Plattner-Institute University of Potsdam,Germany,INFOCOMP2012.

[2] Jianbo Zheng and Weichang Du "Cloud Services for Deploying Client- Server Applications to SaaS", Faculty of Computer Science, University of New Brunswick,Fredericton,Canada

[3] M. Armbrust, A. Fox,R. Griffith, A.Joseph,R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, and M. Zaharia "Above the clouds: A Berkeley view of cloud computing", Technical report, University of California, Berkeley,USA, 2009.

[4] C. Guo et al. "A Framework for Native Multi-Tenancy Application Development and Management", 9th IEEE Intl. Conf. on E-Commerce Technology and 4th IEEE Intl. Conference on Enterprise Computing, Ecommerce andE-Services(CEC-EEE), 2007.

[5] Manoj A. Thomas, Richard T. Redmond, H. Roland Weistroffer "Moving To The Cloud: Transitioning From Client-Server To Service Architecture" , VirginiaCommonwealth University,USA

[6] Nadir K.Salih, Tianyi Zang "Modeling and Self-Configuring SaaS Application", School of Computer Science and Engineering, Harbin Institute ofTechnology, Harbin,Heilongjiang,China

[7] Ekasari Nugraheni "Migration of Web Application SIMA into Multi-tenant SaaS" Research Center for Informatics, Indonesian Institute of Sciences Jl. Sangkuriang 21/154D,Bandung 40135,Indonesia

[8] Yangpeng Zhu "A Platform for Changing legacy Application to Multi- tenant Mode", International Journal of Multimedia and Ubiquitous EngineeringVol.9.No.8(2014). pp.407-418

[9] Software as a Service, http://en.wikipedia.org/wiki/-Software_as_a_service

[10] Cor-Paul Bezemer, Andy Zaidman "Challenges of Reengineering into Multi-Tenant SaaS Applications" Delft University of Technology Software EngineeringResearchGroupTechnicalReport Series

[11] Ganesh Olekar,Vikram Sreekumar "Cloud Computing: Migration from Traditional Systems to the Cloud" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 3,March 2013

[12] Craig D. Weissman and Steve Bobrowski, "The design of the force.com multitenant internet application development platform", In Proc. of the SIGMOD, 2009,pp. 889-896

[13] Suhas Gajakosh, Mukta Takalikar, "Multitenant Software as s Service: Application Development Approach", International Journal of Advanced Computer Research (ISSN) Volume 2, Issue 11, September 2013.

[14] Leo Technosoft - Cloud Computing R&D Center for product development, IT services and infrastructure management, http://www.leotechnosoft.net

[15] D. Jacobs and S. Aulbach, "Ruminations on multitenant databases," BTW Proceedings, 2007.

[16] F. Chong, G. Carraro, and R. Wolter, "Multi-Tenant Data Architecture," MSDN Library, Microsoft Corporation, 2006.

[17] Li heng, Yang dan, Zhang xiaohong, "Survey on Multi-Tenant Data Architecture for SaaS", International Journal of Computer Science Volume 9 Issue 6, November 2012.

[18] Xuequan Zhou,Dechen Zhan, Lanshun Nie, Fanchao Meng, Xiaofei Xu, "Suitable Database Development Framework for Business Component Migration in SaaS Multi-tenant Model", International Conference on Service Science 2013.