## A REPORT

**ON**

**BILLING SYSTEM FOR SAAS PROVIDER**

###### BY

**K.N.V.L. SUNEETHA - 2009C6PS330H**

**P.RAJASHREE RAO - 2009A7PS012H**

**R.SHRAVYA - 2009C6PS535H**

**SUCHI NANDINI - 2009A7PS582H**

UNDER THE SUPERVISION OF

**Dr. N.L.BHANU MURTHY**

**Assistance Professor, Computer Science**

****

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**HYDERABAD CAMPUS**

**October,2011**

## A REPORT

**ON**

**BILLING SYSTEM FOR SAAS PROVIDER**

Submitted in partial fulfillment of the

Computer Projects BITS C331

###### BY

**K.N.V.L. SUNEETHA - 2009C6PS330H**

**P.RAJASHREE RAO - 2009A7PS012H**

**R.SHRAVYA - 2009C6PS535H**

**SUCHI NANDINI - 2009A7PS582H**

UNDER THE SUPERVISION OF

**Dr. N.L.BHANU MURTHY**

**Assistance Professor, Computer Science**

****

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**HYDERABAD CAMPUS**

**October,2011CERTIFICATE**

This is to certify that the report entitled, ----------------------------------------------------------------------------------------------------------------- and submitted by --------------------------------- ID No. ------------------ in partial fulfillment of the requirements of BITS C331 Computer Projects embodies the work done by him/her under my supervision.

Signature of the supervisor

Name:

Designation:

Date:

**ACKNOWLEDGEMENTS**

No project or research is done in a single day by a single person but it is a result of long experimentation and observation where the required environment consisting of many people play a vital role in initializing and completing a work.

First of all our heartfelt thanks to Dr.N.L.Bhanu Murthy, for constantly reviewing our project and resolving any difficulties we faced on our way. For giving us the initial guidance and encouragement regarding our project and insight regarding further research work possible in our area concerned.

We are indebted to BITS Pilani, for giving us the opportunity to experience project development.

Our words are incomplete without our acknowledgment towards our family and our friends, without whose encouragement and support we wouldn‟t have been able to complete our project.

**ABSTRACT**

Recognizing the disruption that cloud computing will have on the technology industry, vendors are rapidly shifting their offerings to the cloud. Emulating early movers like Amazon Web Services, Google App Engine, and Microsoft Azure, many vendors are shifting their business offerings to be more elastic, on-demand, and usage-based. If the business is making a similar shift, then they have to change their technical infrastructure to be able to deliver elastic computing to the customers. At the same time, the shift to cloud computing will require significant changes to the business model. Instead of making one-time, purchases of perpetual licenses, the customers in the cloud will demand to see multiple subscription pricing plans online, pick what is best suited to their needs, and pay on a recurring basis and only for what is used. This cloud business model will have considerable implications on the company‟s cloud commerce infrastructure.

This is shift from convention spreadsheet-billing records to cloud based billing has been implemented, though on a small scale, in our project. The project attempts to highlight all the considerations around metering, pricing, and billing that will need to go into any commerce system to support a successful cloud offering.

**TABLE OF CONTENTS**

**1.Introduction 7**

**2.The Cloud Stacks 12**

**3.Why SaaS 18**

**4.Billing system for cloud 30**

**5.Challenges in SaaS billing 30**

**6.Struts 2 45**

**7.Product details 46**

**8.Conclusion 30**

**9.References 30**

**1.INTRODUCTION**

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information are provided to computers and other devices as a utility over a network.

The “cloud” in the cloud computing comes from the terminology used to describe large networks such as internet. Rather than the data stored in and accessed from the drive of an individual computer, it is held off in the “cloud” of the network.

Cloud Computing is a *style of computing which* must cater to the following computing needs:

**1. Dynamism**

**2. Abstraction**

**3. Resource Sharing**

**Dynamism**

The growing of business exponentially demands more and more investment in hardware to meet the growing demands. The rapid changes in the business environment like recession or rapid bursts in the demand traffic, poses great a threat to business. Cloud computing reduces such risk as end-users just need to configure and providers will take care of fluctuating demand.

**Abstraction**

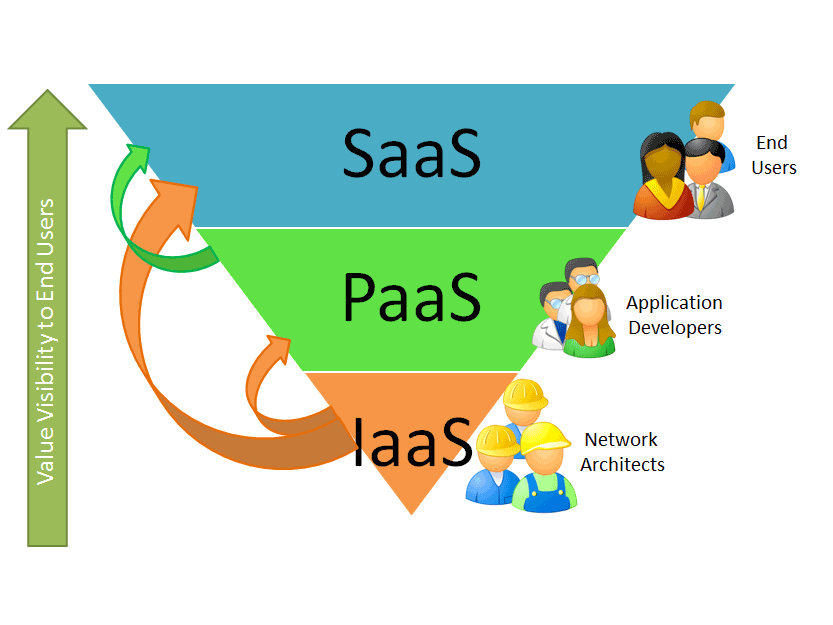
The business should focus on the core competency with least botheration about security, OS, software platform , updates and patches etc. With cloud computing these chores can be left to the providers. From an end users perspective, they don’t need to care for the OS, the plug-ins, web security or the software platform.

**Resource Sharing**

Resource Sharing is the beauty of Cloud Computing. This is the concept which helps the cloud providers to attain optimum utilization of resources. Example, a company dealing in gifts may require more server resources during festive season. A company dealing in Payroll management may require more resources during the end or beginning of the month.

The cloud architecture is implemented in such a way that it provides the flexibility to share application as well as other network resources (hardware etc.). This will lead to a need based flexible architecture where the resources will *expand* or *contract* with little configuration changes.

**2.THE CLOUD STACKS**



**IaaS:**

This is the base layer of the cloud stack. It serves as the foundation for the other two layers, for their execution.

The keyword behind this layer is Virtualization. The applications are executed on a virtual computer (instance). The configuration of CPU, memory and storage that is optimal for the application has to be specified a priori. The whole cloud infrastructure viz. servers, routers, hardware based load-balancing, firewalls, storage and other network equipment are provided by the IAAS providers. The customer buy these resources as a service on a need basis.

**PaaS:**

It provides a [computing platform](http://en.wikipedia.org/wiki/Computing_platform) and a [solution stack](http://en.wikipedia.org/wiki/Solution_stack) as a service. PaaS offerings facilitate the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities, providing all of the facilities required to support the complete life cycle of building and delivering [web applications](http://en.wikipedia.org/wiki/Web_application) and [services](http://en.wikipedia.org/wiki/Web_service) entirely available from the Internet.

PaaS offerings may include facilities for application design, application development, testing, deployment and hosting as well as application services such as team collaboration, web service integration and [marshalling](http://en.wikipedia.org/wiki/Marshalling_(computer_science)), database integration, security, scalability, storage, persistence, state management, application versioning, application instrumentation and developer community facilitation. These services may be provisioned as an integrated solution over the web.

**SaaS:**

Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet.

 (HRM), [content management](http://en.wikipedia.org/wiki/Content_management) (CM) and service desk management.[[1]](http://en.wikipedia.org/wiki/Software_as_a_service#cite_note-0) SaaS has been incorporated into the strategy of all leading [enterprise software](http://en.wikipedia.org/wiki/Enterprise_software)companies. SaaS is becoming an increasingly prevalent delivery model as underlying technologies that support [Web services](http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci750567,00.html) and service-oriented architecture ([SOA](http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci929153,00.html)) mature and new developmental approaches, such as [Ajax](http://searchwindevelopment.techtarget.com/sDefinition/0,,sid8_gci1107521,00.html), become popular.  SaaS is closely related to the [ASP](http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci213801,00.html) (application service provider) and [on demand computing](http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci903730,00.html) software delivery models.

SaaS has become a common delivery model for most business applications, including [accounting](http://en.wikipedia.org/wiki/Accounting_software), [collaboration](http://en.wikipedia.org/wiki/Collaborative_software) , [customer relationship management](http://en.wikipedia.org/wiki/Customer_relationship_management) (CRM), [enterprise resource planning](http://en.wikipedia.org/wiki/Enterprise_resource_planning) (ERP), invoicing, [human resource management](http://en.wikipedia.org/wiki/Human_resource_management)

**3.WHY SAAS?**

* No upfront expenses.
* You just need a web browser to access the application. No other hardware purchase or software installation is necessary.
* Quick deployment or it’s already deployed and ready to use. Forget those sleepless nights of traditional application deployment - remember what happens when there’s a *go-live*at the production server!
* Unlike the traditional apps, cloud architecture makes SaaS highly *scalable*.
* *Multi-tenant* architecture makes SaaS highly efficient as the *source code is the same* for every customer.
* Unlike traditional apps where *customization* is the key; a true SaaS can meet any requirement by simple *configuration*.
* Upgrades are applied directly at the SaaS provider’s end. No headache at the customer end.
* Since all the customers are using the same code base, any new tech-innovation is easily integrated by the provider, and is available for all the subscribers.

**4.BILLING SYSTEM FOR CLOUD**

Cloud computing is based on a usage model where access to computing resources is delivered through Internet technologies. The user pays per usage, rather than buying a license and annual maintenance. Infrastructure costs (servers, DASD, network costs) are typically included in SaaS. Usage can fluctuate and a billing system needs to reflect the billing usage granularity per the cloud provider’s business model.

**Infrastructure as a service (IAAS):**

Different hardware resources are provided through Infrastructure as a Service (IaaS).

Charging model examples include the following:

•Cpus: CPUs are differentiated by power and number of CPU cores and, consequently, price. CPU power may be differentiated by time zone, e.g., static (based on peak and off peak resources) or dynamic, where price is determined by demand at the time.

• server type: Because the same CPU can be deployed either in a low cost server or in a top-of-the-range server with high availability and a significantly different cost point, the customer price must reflect this variation.

• system administration: The same server type resource may be charged at a different rate depending on the operating system (e.g., Windows or Linux).

•storage (dAsd): Different storage capacity (including mirroring) is available, as well as different types of storage reflecting different price points from disk storage suppliers. The same variability exists as in the case of CPUs. For example, the cost for 1GB will vary depending on whether it’s provided in a low- or high- end unit.

• disaster recovery: This involves the time window within which SaaS would need to be available should a disaster take out a data center from which SaaS is provided. For short time window an active-active deployment in two data centers may be required.

•other: Charges for space, power, network capacity, security, operating system and so on are built into the infrastructure pricing.

•service level agreements (slAs): If high availability is part of the agreement,SLAs may impact the price (e.g., refunds when contractual SLAs are not achieved).

Billing for IaaS may be done based on the quantity and quality of the infrastructure Resources provided.

**Platform as a service (PAAS):**

PaaS includes software frameworks and the necessary hardware in which to develop and deliver Software as a Service (SaaS). Examples of such frameworks include the following:

•Different hardware architectures with different server sizes—from small, Intel-based servers to mid- or top-range servers and mainframes—utilizing different chips

• Various software operating systems (e.g. Windows, Linux , MAC OS, Solaris )

• Various development and application frameworks (e.g., Java,.Net)

• Solution stacks(e.g., LAMP,MAMP,WINS,and so on)

Billing must take into account Infrastructure as a Service (IaaS) costs, as well as software features and product offerings provided in the PaaS layer. Different frameworks have different prices and may include different infrastructures. All of this, together with usage, needs to be taken into account.

**5.CHALLENGES IN SaaS BILLING**

The different software modules within a single cloud offering , are likely to follow different pricing models. Example: Consider a company offering messaging,VoIP, videoconferencing( low and high grade) etc. modules. These offerings provided as SaaS will need to reflect different pricing based on number of such modules used and each module usage. These offerings need servers, handsets and communications “pipes” from communication service providers, which, in turn, requires a data center infrastructure, including system administration, database administration, security, SLAs, and so on. The amount of computing power will depend on the number of users, usage and the number of these applications. Different application modules may have a different number of users and a different infrastructure utilization per user. For example, VoIP and high resolution videoconferencing will have different network capacity requirements. SaaS provided by a vendor or a systems integrator would thus include the price for an integrated solution. In a “pay per use” environment, this price would need to reflect the above costs directly—through a breakdown on the bill—or indirectly by abstracting some of the detail to the end user. However, the cloud provider would still likely want to understand the detailed usage breakdown, so as to establish the right business model and, in turn, plan the capacity of different “tiers”. Such cloud offerings may have different ServiceLevelAgreements and associated penalties for non-compliance.

Also, such modules need to be updated over time as product suppliers issue new releases. Each upgrade cost may vary depending on the implementation effort required.

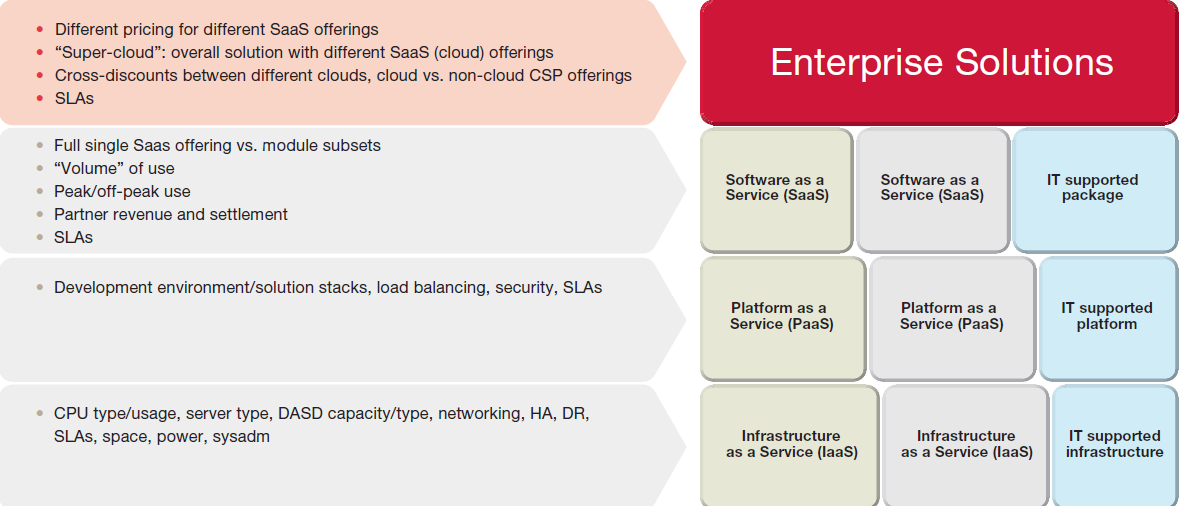
Another recurring cost is maintenance, which splits into two parts:

1. Product supplier maintenance cost (equivalent to an annual maintenance contract

in a licensed environment)

2) Systems integration support and maintenance of the specific implementation.

The pricing for the SaaS may be based on the number of service invocations in the crudest form.





**6.STRUTS 2**

Apache Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications. The framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time.

Apache Struts2 was originally known as WebWork 2. After working independently for several years, the WebWork and Struts communities joined forces to create Struts2. This new version of Struts is simpler to use and closer to how Struts was always meant to be. Struts 2 is a pull-MVC framework. i.e. the data that is to be displayed to user has to be pulled from the Action.

**MVC Framework:**

Following are the components of MVC framework:

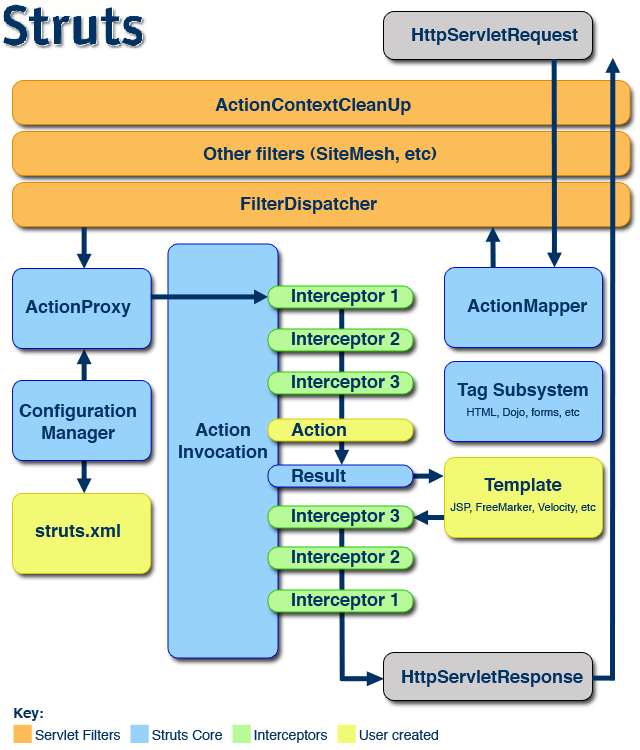
**1.Model:** The model object knows about all the data that need to be displayed. It is model who is aware about all the operations that can be applied to transform that object. It only represents the data of an application. The model represents enterprise data and the business rules that govern access to and updates of this data. Model is not aware about the presentation data and how that data will be displayed to the browser.

**2. View :** The view represents the presentation of the application. The view object refers to the model. It uses the query methods of the model to obtain the contents and renders it. The view is not dependent on the application logic. It remains same if there is any modification in the business logic. In other words, we can say that it is the responsibility of the of the view's to maintain the consistency in its presentation when the model changes.

**3.Controller:**  Whenever the user sends a request for something then it always go through the controller. The controller is responsible for intercepting the requests from view and passes it to the model for the appropriate action. After the action has been taken on the data, the controller is responsible for directing the appropriate view to the user. In  GUIs, the views and the controllers often work very closely together.

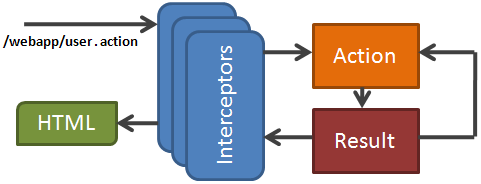
In struts 2 the model, view and controller are implemented by the action, result and filter Dispatcher respectively. The controller’s job is to map the user request to appropriate action. In struts 2 filterDispatcher does the job of controller. Model contains the data and business logic. In struts 2 the model is implemented by the action component. View is the presentation component of the MVC pattern. In struts 2 the view is commonly implemented by using JSP, Velocity template, Freemarker or some other presentation-layer technology.

**Architecture of Struts 2**

Struts 2 Architecture is based on WebWork 2 framework. It leverages the standard JEE technologies such as Java Filters, JavaBeans, ResourceBundles, Locales, XML etc in its architecture. Following is its framework diagram.  


1. The normal lifecycle of struts begins when the request is sent from client. This results invoke the servlet container which in turn is passed through standard filter chain.
2. The FilterDispatcher filter is called which consults the ActionMapper to determine whether an Action should be invoked.
3. If ActionMapper finds an Action to be invoked, the FilterDispatcher delegates control to ActionProxy.
4. ActionProxy reads the configuration file such as struts.xml. ActionProxy creates an instance of ActionInvocation class and delegates the control.
5. ActionInvocation is responsible for command pattern implementation. It invokes the Interceptors one by one (if required) and then invoke the Action.
6. Once the Action returns, the ActionInvocation is responsible for looking up the proper result associated with the Action result code mapped in struts.xml.
7. The Interceptors are executed again in reverse order and the response is returned to the Filter (In most cases to FilterDispatcher). And the result is then sent to the servlet container which in turns send it back to client.

**Request Processing Lifecycle**



1. Request is generated by user and sent to Servlet container.
2. Servlet container invokes FilterDispatcher filter which in turn determines appropriate action.
3. One by one Intercetors are applied before calling the Action. Interceptors performs tasks such as Logging, Validation, File Upload, Double-submit guard etc.
4. Action is executed and the Result is generated by Action.
5. The output of Action is rendered in the view (JSP, Velocity, etc) and the result is returned to the user.

**Advantages of struts**

1.Centralized File-Based Configuration:

Struts values/mapping are represented in XML or property files. This loose coupling means that many changes can be made without modifying or recompiling Java code, and that wholesale changes can be made by editing a single file. This approach also lets Java and Web developers focus on their specific tasks (implementing business logic, presenting certain values to clients, etc.) without needing to know about the overall system layout.

2.FormBeans

3. Bean Tags:

Struts provides a set of custom JSP tags that let you easily output the properties of JavaBeans components.

4. HTML Tags:

Struts provides a set of custom JSP tags to create HTML forms that are associated with Java Beans components. This bean/form association serves two useful purposes:   
+ It lets you get initial form-field values from Java objects.   
+ It lets you redisplay forms with some or all previously entered values intact.

5.Form Field Validation:

Struts has a robust, extensible validator that can be used to uniformly validate your form fields. This validation can be performed on the server (in Java), or both on the server and on the client (in JavaScript).

**7.PRODUCT DETAILS**

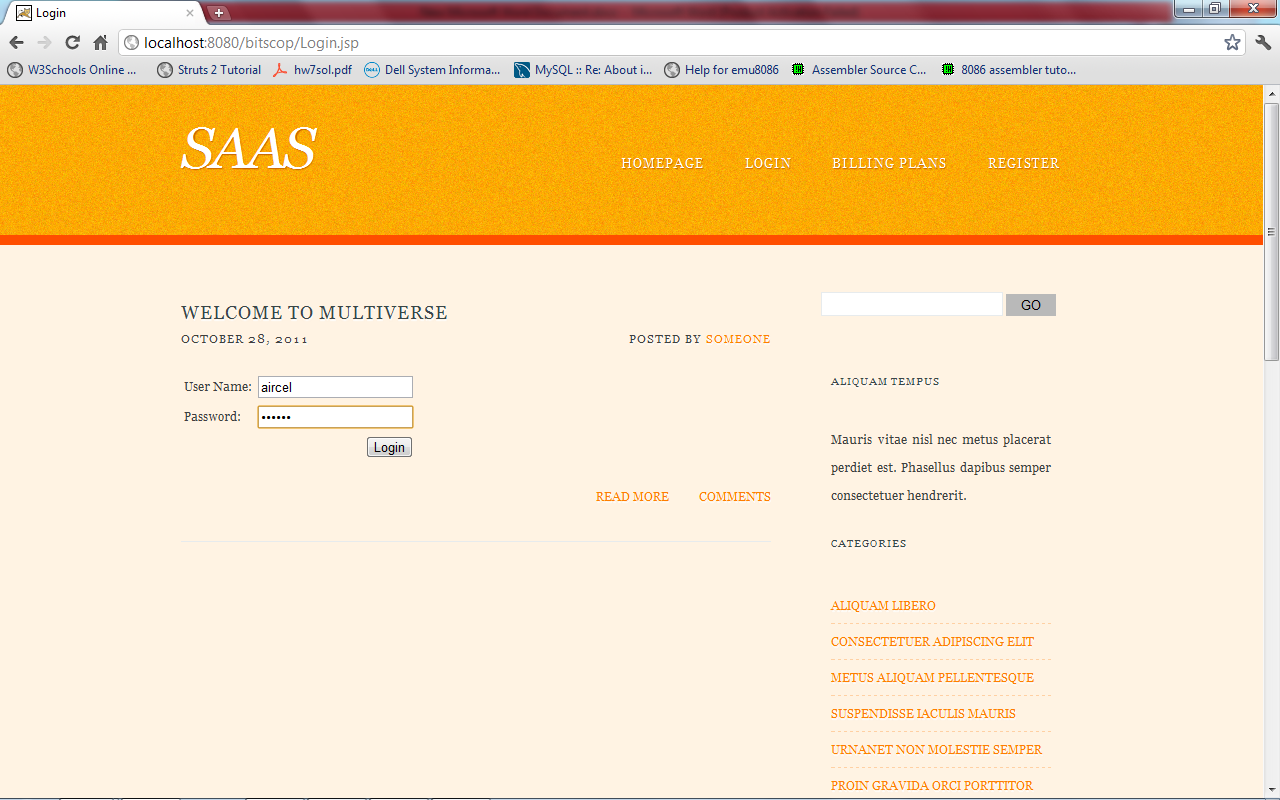
Entire product has been divided into two modules registration and login modules. Homepage gives a brief description about the advantages of automated billing system for a business and features of our billing system.

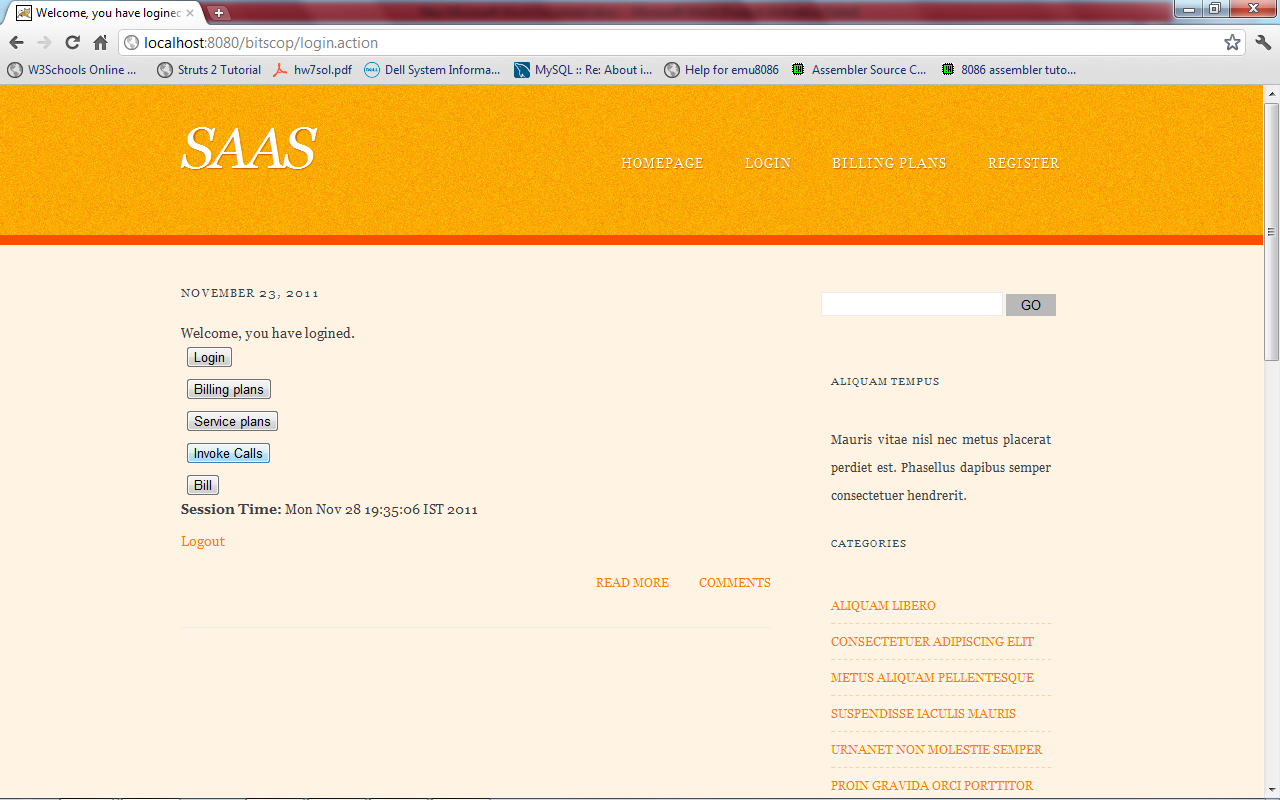


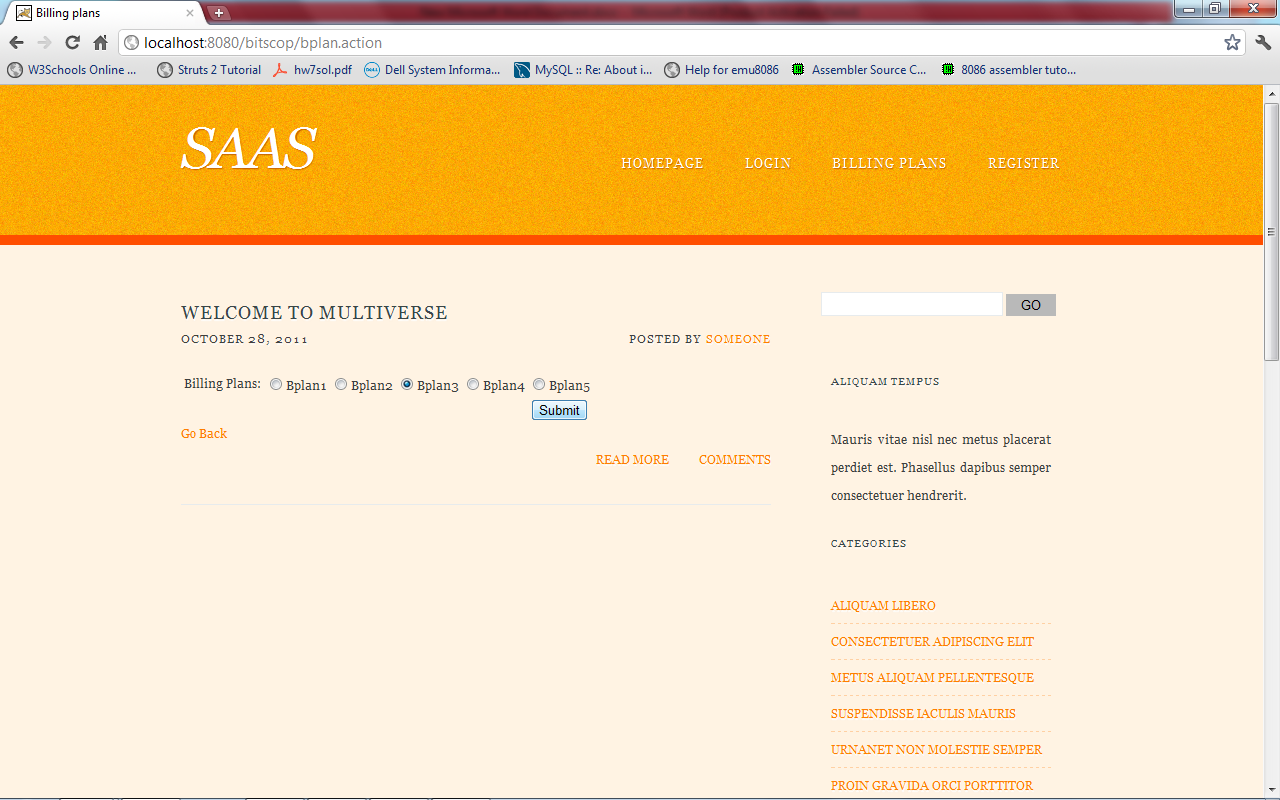
In order to avail our product the customer has to register wherein he will be given a unique userid and password so that he can login anytime. Along with these, additional details such as company name and address are taken. These details about the customer are sent into the database for future reference.

Coming to the login module only registered user can login with correct username and password. Newly registered customers can select the billing plan and services by clicking on the respective tabs. For their convenience detailed description of each billing plan and service is provided. Billing plan selected and services chosen are stored in the database.

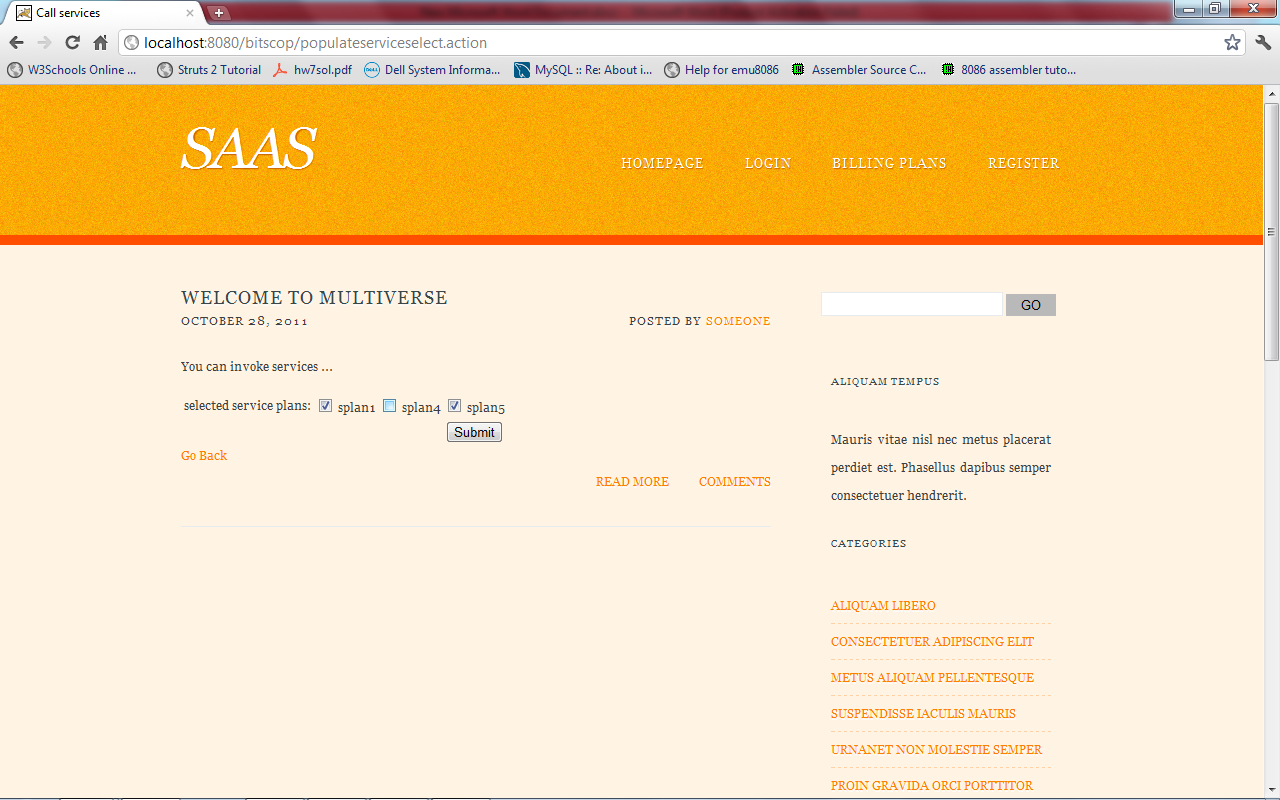
This is login page. Below snapshot shows the options after the user has loginned.





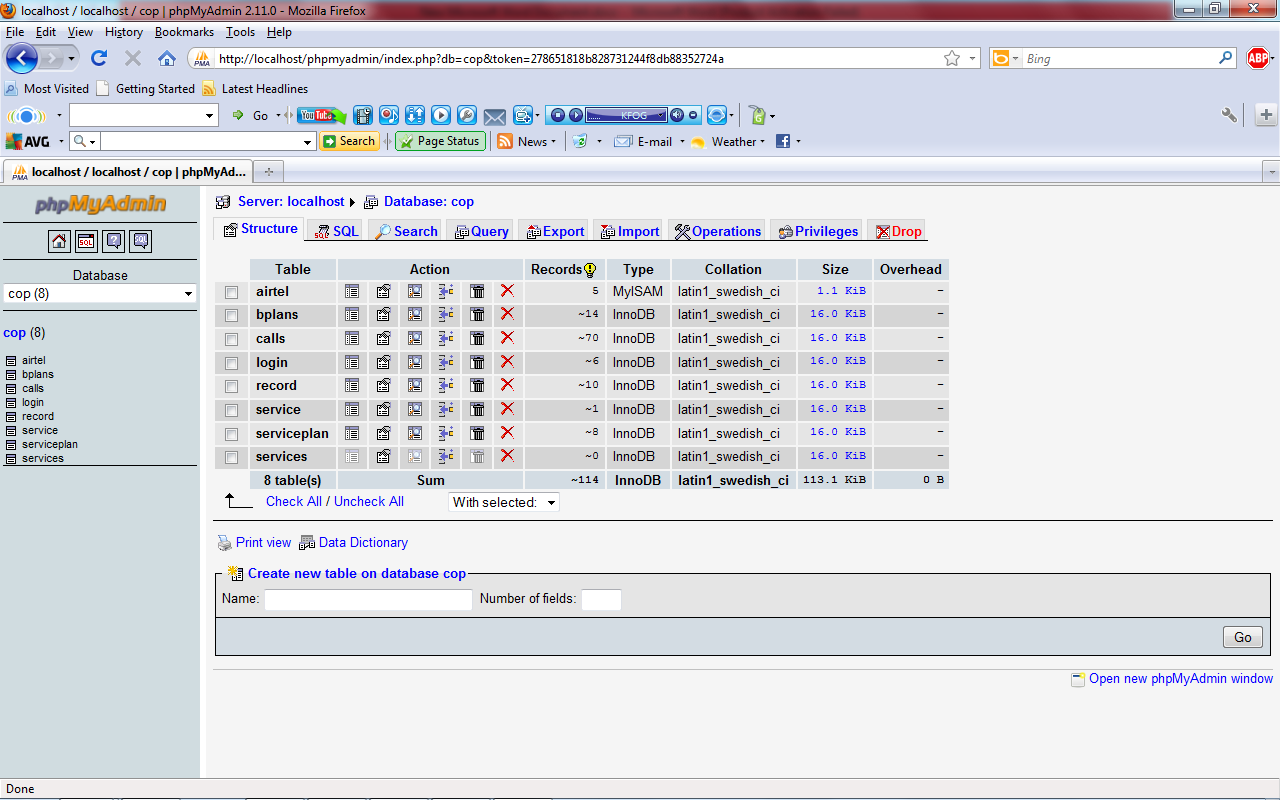


Selection of billing plans



Invoking calls

Already registered users can invoke calls and they can check their bill at the end of month.



In order to enact a real time situation we have created some files. Each file contains username and the service (actually service code given by us to each service) they selected along with time at which the service has been invoked. Several threads have been created to read each file. After reading, threads send data to a queue from where they are stored temporarily. From this queue they are finally sent to database where userid, service invoked by him and the time of invocation are recorded. This represents a real time situation where several users invoke different services. Finally with these data bill is calculated and sent to view page.

**8.BILLING PLANS**

In each of the billing plans the client has to pay small amount of advanced charges in order to get services from the SAAS service provider

**1.Fixed price billing plan:** In these scheme services are grouped depending on their complexity level. Amount charged per call for each service within the group is same but will vary among different groups. Based on the amount generated at the end of the day discount is provided.

For example let the initial amount for registering be Rs4,00,000. Let there be three types of services. Following table gives cost per call per service in each group:

|  |  |
| --- | --- |
| Service | Cost per call |
| 1. Service1 | Rs.5/call |
| 1. Service2 | Rs.6.5/call |
| 1. Service3 | Rs.4.5/call |

Discounts provided are as follows:

Rs.5,000<bill amount<Rs.10,000—discount is 1.2%

Bill amount>Rs.10,000—discount is 1.8%

The calls invoked for one month are as follows:

|  |  |
| --- | --- |
| Service | Number of calls |
| 1. Service1 | 500 |
| 1. Service2 | 500 |
| 1. Service3 | 300 |

Bill amount=Rs.7,100

After applying discount, bill amount=Rs.7,185.2

Finally the customer has to pay Rs.4,07,185.2

**2.Billing plan based on number of calls:** In this scheme depending upon number of calls cost per call varies. Based on the amount generated at the end of the day discount is provided.

For example let the initial amount for registering be Rs4,00,000. Let us consider five ranges for the number of calls.

|  |  |
| --- | --- |
| Call ranges | Cost per call |
| 1-500 calls | Rs.5/call |
| 501-1000 calls | Rs.4.5/call |
| 1001-2000 calls | Rs.4/call |
| >2000 calls | Rs.3/call |

Discounts provided are as follows:

Rs.2,000<bill amount<Rs.10,000—discount is 1.2%

Bill amount>Rs.10,000—discount is 1.8%

Suppose a particular customer has invoked 550 calls in a month. Bill generated is Rs.2725

After applying discount, bill amount=Rs.2,757.7

Finally the customer has to pay Rs.4,02,757.7

**9.CONCLUSION**

Many cloud items need to be sorted out to fully meet the promise of cloud computing. Flexibility in billing is one of them. Ignoring this aspect will mean that the cloud provider doesn’t reflect its true costs in the customer model, which in turn is likely to either impact the provider’s competitiveness through overcharging or its survival through undercharging. Supporting this functionality allows the provider to aggregate data and to understand usage patterns. This is important for capacity planning as well as offering analysis for sales and marketing. To fulfill the promise of cloud computing and charge per usage, flexibility in billing—the complexity of which is comparable to convergent telecom billing—is a key ingredient for cloud providers