Spring 2013

Cloud Based Complaint Management Service

Ajinkya Amrute
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Amrute, Ajinkya, "Cloud Based Complaint Management Service" (2013). *Master's Projects*. 298.
DOI: <https://doi.org/10.31979/etd.xd55-gcvd>
https://scholarworks.sjsu.edu/etd_projects/298

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

**Cloud Based Complaint Management
System as a Service**

A Writing Project

Presented to

The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment

Of the Requirements for the Degree
Master of Science

By

Ajinkya Amrute

May 2013

© 2013

Ajinkya Amrute

ALL RIGHTS RESERVED

ACKNOWLEDGEMENTS

I am extremely obliged to my project advisor, Dr. Soon Tee Teoh for guiding and helping me out in implementing my ideas in my Master's Project and also would like to thank my committee members, Dr. Robert Chun and Mr. Darpan Kathoke for their time and support.

I would like to thank my parents and my sister, for always motivating me and my friends who provided their help whenever it was needed.

Abstract

Cloud Based Complaint Management Service

By Ajinkya Amrute

Complaint Management is important from both customer as well as business point of view. Complaints contain direct voice of the customer which provides companies a huge volume of data which can be used to improve the quality of the product the company is manufacturing. Hence it is necessary for the organizations to harness the data received via complaints. However as the data received via complaints is enormous it is not an easy task to manage the data received via complaints as the data keeps on expanding and multiplying. For implementing this difficult task of complaint management, websites were built but it becomes necessary for the customer to have a desktop/ laptop for using the system built for complaint management. However with time, mobiles have taken place of the Laptops / Desktops due to their on the fly nature which helps an individual to communicate or work on his personal tasks while he/ she is travelling. Hence in my project I have provided mobile support to these kinds of systems which has to deal with huge volumes of data as it will be beneficial. However mobile devices on their own do not possess sufficient level of resources for dealing with such level of computations that are done on the data. Hence I have cloud computing utilized for this kind of systems which will take care of the heavy computational stuff leaving mobile phone to efficiently function while people are using such a system. Hence my project aims to develop a mobile based complaint management system which will be provided as a service to the users of the system. I have also added the feature of the multi-tenancy to the system using which the system can be configured according to the requirements of an organization.

Table of Contents

1.0 Introduction.....	9
1.1 Project Overview.....	9
1.2 Project Deliverables.....	10
2.0 Project Architecture.....	11
2.1 Upper-Level Architecture.....	11
2.2 Subsystems.....	12
3.0 Technologies Used.....	15
3.1 Client-Tier Technologies.....	15
3.2 Middle-Tier Technologies.....	16
3.3 Data-Tier Technologies.....	18
3.4 Other Technologies.....	19
4.0 Software System.....	21
4.1 User Characteristics.....	21
4.2 Conceptual System.....	22
4.3 Multi-Tenancy Implementation.....	26
4.4 Use Cases.....	35
4.5 Description of Sequence.....	37
5.0 Project Implementation.....	38
5.1 Generic Workflow Of The Application.....	38
5.2 Client Implementation.....	40
5.3 Server Implementation.....	56
6.0 Testing.....	62
7.0 Conclusion.....	66
References.....	67

List of figures

FIGURE 1.UPPER LEVEL SYSTEM ARCHITECTURE.....	11
FIGURE 2.MOBILE CLIENT ARCHITECTURE	12
FIGURE 3.CONCEPTUAL SYSTEM DESIGN	22
FIGURE 4.ER DIAGRAM FOR ENTIRE DATABASE SCHEMA.....	27
FIGURE 5.ER DIAGRAM FOR THE SYSTEM	28
FIGURE 6.ER DIAGRAM FOR THE MULTI-TENANCY SUPPORT	31
FIGURE 7.USE CASE DIAGRAM	35
FIGURE 8.MOBILE UI FOR LOGIN SCREEN FOR TENANT 1.....	40
FIGURE 9.MOBILE UI FOR EDITING PROFILE FOR TENANT 1.....	41
FIGURE 10.MOBILE UI FOR CHANGING PASSWORD IN A USER PROFILE FOR TENANT 1	42
FIGURE 11.MOBILE UI FOR BUILDING A USER PROFILE FOR TENANT 1	43
FIGURE 12.MOBILE UI FOR COMPLAINTS FOR THE CUSTOMER FOR TENANT 1.....	44
FIGURE 13.MOBILE UI FOR COMPLAINTS FOR THE CMS_STAFF FOR TENANT 1.....	45
FIGURE 14.MOBILE UI FOR REGISTERING A COMPLAINT BY A CUSTOMER FOR TENANT 1	46
FIGURE 15.MOBILE UI FOR VIEWING THE COMPLAINT AND ITS STATUS FOR TENANT 1.....	47
FIGURE 16.MOBILE UI FOR LOGIN SCREEN FOR TENANT 2	48
FIGURE 17.MOBILE UI FOR COMPLAINTS FOR THE CUSTOMER FOR TENANT 2.....	49
FIGURE 18.MOBILE UI FOR VIEWING THE COMPLAINT AND ITS STATUS FOR TENANT 2.....	50
FIGURE 19.THREAD GROUP SELECTION.....	63
FIGURE 20.TESTING RESULTS IN TABULAR FORM	64
FIGURE 21.TESTING RESULTS IN TABULAR FORM	65

List of Tables

TABLE 1.APPLICATION TO TECHNOLOGY MAPPING	15
TABLE 2.USE CASE DESCRIPTION	36
TABLE 3.TYPES OF TESTING METHODS USED	62

1. Introduction

1.1 Project Overview

One of the fundamental parts of Industry is customer complaint management. And for each of the functions from registering a complaint to resolution of complaint there are administrative bodies. Hence the process of filing a complaint by a customer till it gets resolved by a regulatory body is conventional. Complaints not only incur direct expenses but also indirect expenses. Hence complaints are expensive [7]. However as the complaints embrace the voice of the customer they provide critical knowledge about the company and its products which can be utilized for the improvement of the organization. Hence it is very crucial for companies to have a system which manages complaints for them and hence customer complaint management system (CCMS) is a step towards compliance management [4].

As the data accumulated from various complaints keeps on increasing it becomes difficult to efficiently manage the complaints. Moreover most of the complaint management systems are only web based and go through various problems when used on mobile phones, as they are not built specifically for mobile devices. However by the use of mobile cloud computing the all the hindrances of having a smooth and efficient complaint management system could be solved.

Hence I have developed a cloud based multitenant complaint management service for efficiently manage customer complaints which will used by different companies [5]. And in the system I have implemented the concept of multi-tenancy which provides the system with the property of designing the system according to terms and requirements of the organization.

1.2 Project Deliverables

Any company who manufactures and sells products may receive complaints about it from its customers. So for handling these complaints a company needs to have its complaint management system so that the complaints get handled effectively and as soon as possible. Hence the working of the complaint management system built is as discussed as follows. Registration of the customer complaints is done using an account that will be created by customer for registering a complaint. Customer selects the product for which he wants to issue a complaint and will issue the complaint for the same. After the registration of the complaint, the complaint automatically gets assigned to appropriate serviceman handling the complaints related to the product [7]. Both the customer of a product and the serviceman working on the complaints issued on that particular product has the facility to communicate with each other so that the complaint gets addressed quickly.

- Registration of the customer complaint through an android application.
- Automatic and immediate assignment of customer complaint to specific service person.
- Intercommunication between service man and customer.
- Displaying complaint status information for customer as well as serviceman.

The major deliverables of my project are:

- 1) Complaint management provided as a service which can be used by any organization.
- 2) Provided multi-tenancy at the database layer.
- 3) Optimized source code with proper comments.
- 4) Testing performed on the project.
- 5) Report encompassing information such as architecture of the system, system constitution, technologies that will be used for service implementation.

2. Project Architecture

2.1 Upper-level Architecture

The designed system is built on the basis of 3-tier client-server architecture. The database server is built using JAVA and is resident over cloud; the client side inputs are received from a mobile device and the interaction between the client and the server side is achieved using JSON messages which are sent using the REST calls. All the data is stored using MySQL database which is a relational database. And finally the application is hosted on Amazon EC2.

The architecture is as shown below:

FIGURE 1.UPPER LEVEL SYSTEM ARCHITECTURE

2.2 Subsystems

Client Tier – User interacts with the system using mobile user interface and the mobile UI is built using two technologies i.e. Android and Java. And the user interface first of all provides a platform for the user to communicate with the system and performs the system dedicated actions which the user desires. The interaction is performed by doing the sending and receiving of data which is in JSON format and the web technology used is REST web service.

FIGURE 2. MOBILE CLIENT ARCHITECTURE

Mobile Client GUI- This is where the user interacts with the system through the user interface that is built for mobile.

Authentication- The users of the system are Customer, staff working in the complaint management system and each user is assigned a unique role. So after successful authentication of the user, each user will have access to a particular set of modules in the system with respect to the role of the user.

Session Manager- The duty of this module is to have a continued session between a login and a log out of the user. This way we could keep track of the data and the functions performed in a particular session.

Administration- The administration modules endows the administrator of the system with the permission and accessibility which the admin uses to add/remove features, client and access.

Connectivity- Through this module the users utilizes the WiFi network to access the complaint management service.

Local Memory Storage – This module helps the user to store the data temporarily in the cache of the device which reduces the database round trips made to the database for bringing or loading the data which is requested.

Server Tier

All the business logic is written at the server side and REST web service is used by the server for communication with its clients. The system is hosted on the Amazon EC2 cloud and so uses the power and the resources provided by the cloud provider to perform its various functions.

This tier performs following tasks:

- Interaction between the client and the server done by sending and receiving data in JSON format which is done by using the REST web service.
- Ensures secure connection between the clients.
- Perform database related operations using the database connector.

Data Tier – The data tier consists of the database which is shared across companies using the feature of multi-tenancy so that more than one company can be handled at the database level. The type of database used for storing the data is relational and is a MySQL database.

3. Technologies Used

Application Components	Technologies
Server	Java
Mobile Client	Android
Application Server	Tomcat
Message Communication	REST
Cloud Provider	Amazon EC2
Database	MySQL

TABLE 1.APPLICATION TO TECHNOLOGY MAPPING

3.1 Client-Tier Technologies

The system is built for Android based Mobile Client and hence utilizes Android SDK (Software development Kit) and the Java programming language for its implementation [4]. For using the Android SDK with JAVA, the Integrated Development Environment (IDE) requires the ADT plugin [3]. And for compiling the Android based code we need a virtual machine DVM (Dalvik Virtual Machine) which is used for running the android application. The client only deals with the interaction of data with the server where the main computing and execution of the business logic takes place.

3.2 Middle-Tier Technologies

The middle tier comprises of the server which is hosted on Amazon EC2 cloud and runs an instance of the server on it [9]. The communication between the client side and the server side is done using REST (Representational State Transfer) calls. However it has to be used with addition of a plugin in the Eclipse (IDE) for utilizing the JAX-RS API which is needed for RESTful web services. And the working of JAX-RS utilizes annotation

Advantages of REST over SOAP Protocol are as follows:

- a) The Lightweight nature of REST makes it easy to use REST web services.
- b) REST services deals with both XML as well as JSON data format.
- c) REST is efficient while performing HTTP request-response operations.
- d) The development and deployment both are simple in REST rather than in SOAP.

Cloud Service:

Software applications are built independently and then are hosted on cloud and once hosted can be provided as a service to the users. And the space required hosting an application or a system is given by the service providers. I have used Amazon EC2 which is a web services cloud provider. Amazon Elastic Compute Cloud (Amazon EC2) provides the computing resources to developers which help them focus on other development aspects of the system as they don't need to care about managing resources for the smooth functioning of the system [1]. For using the Amazon web services we need to host our application the server, appropriately configure it and finally run a server instance.

Properties and advantages of Amazon EC2

- a) Flexibility –We have two options to configure the system in the cloud. We can either use the private cloud or use the public cloud. Out of which I have used the public one.
- b) Elasticity – Multiple instances of a system can be running on a cloud server which provides better availability and scalability to the application.
- c) Reliability – Amazon EC2 is built on a very secure platform and the resources used for building it are reliable and organized which minimizes the chance of server going down or any such kind of problem.
- d) Compatibility – Like Amazon EC2 there are other services too like the Amazon S3, Amazon AQS. And even these services are compatible with EC2.
- e) Security –Amazon EC2 is secure as it employs many network security principles for its security.

3.3 Data Tier Technologies

MySQL: MySQL database is a relational database which I have used for storing and processing the complaint management data. Specifically I have used MySQL workbench which is an IDE for writing SQL queries and work with the databases. There is a service in Amazon called the Amazon RDS which provides MySQL databases. So even the database related operations are taken care of by the service.

Properties and Advantages of Amazon RDS:

- a) Scalability – Multiple instances can be created which scales the Application on the database side.
- b) Easy deployment – Due to the easy interface managing the AWS console for using the Amazon Web Services becomes easy.
- c) Compatibility – MySQL has been provided by Amazon RDS provides which allows us to directly connect and work on the databases.
- d) Reliability – Amazon RDS is reliable considering the steps taken for ensuring safety and employs some common safety measures for its safety.

3.4 Other Technologies

Java:

Java language is used for performing object oriented programming. It also provides the developers with lots of class libraries using which the developers are able to produce better Client as well as Server side systems used for building a system. Also android uses the Java language for developing applications.

Windows Operating System:

I have used to the Windows operating system to build the System since it is one of the most widely used operating systems and hence it would reduce any kind compatibility issues that are operating system specific.

Eclipse IDE:

Eclipse is an IDE (Integrated Development Environment) which is used by me for developing Java based applications. It also provides plugins for development of application for other platforms. Android application development can be performed using the ADT plugin when attached with Eclipse.

Android SDK, JDK and JRE:

Android SDK (Software development Kit) is used for implementing the mobile client for our system but Android programming can't be done independently it needs JDK and JRE and hence is developed using java [3]. Android uses XML for creating layouts/ screens which is needed for Mobile User Interface.

REST Web Services and REST API:

All the HTTP request-response methods like the GET, POST and DELETE etc. are taken care of by the REST based Web Services.

4. Software System

4.1 User characteristics

There are 3 types of users in the system

1. Administrator:

Administrator has a central control over the system.

2. Service Man:

Service man is the person to whom the complaint filed is sent. Once a complaint received and service person can start its working. If possible service man can solve a complaint just by simple message communication with the user.

3. Customer:

Customer can register his complaint. He/ She communicate with the service man of the company regarding the complaint.

4.2 Conceptual System

The Conceptual system Design gives us an overview of the system. It explains the various components involved in the "COMPLAINT MANAGEMENT SYSTEM" and their interaction with each other to achieve the process of Complain Handling.

FIGURE 3.CONCEPTUAL SYSTEM DESIGN

Subsystem Description:

1. Subsystem for User Interface

Various users of the system use the User Interface Subsystem for logging into the system and depending on the user role only relevant functions are visible to the user. The objective of this module is to provide a user friendly interface which provides user an easy interaction with the system.

2. Subsystem for Authentication

Validation of the user is the main objective of this module which authenticates the user depending on the username and password supplied by the user. Once validation is phase is over it provides a channel through which a user could access the principal functions of the system.

3. Subsystem for Membership

This subsystem is relevant to the member details of the customer which are stored while registering a user. And then this information saved during registration is used by the authentication subsystem for authenticating a user. It creates the valid member profile which is stored in the database.

4. Subsystem for Serviceman

After the registration of the complaint is done, the complaint is assigned automatically and quickly to appropriate serviceman by the complaint handling subsystem .Then the service person subsystem works on the complaint and if necessary can directly communicate with the customer for resolving the complaint as soon as possible.

5. Subsystem for Complaint Registration

A complaint is registered by the customer using the complaint registration subsystem of the mobile client. Mobile user interface is used by the customer for filling out the form about the complaint and selects the product name and files the complaint with a message if necessary. All the information is saved in the database. This way the complaint registration subsystem works.

6. Subsystem for Storage

Storage subsystem keeps all the users of the system. It also keeps the data about the communication which takes place between the customer and the serviceman so that the data might be helpful for solving other complaints. The administrator has full control over the system and can modify the system depending upon the requirements. It also keeps data regarding the complaints that were solved previously for the future use.

7. Subsystem for Administrator

The subsystem is mainly controlled by the administrator hence it provides administrator all the rights and privileges to modify a database or add a client to the system etc. which will be helpful for implementing multi-tenancy depending upon the requirements.

8. Subsystem for Communication

The subsystem mainly is used for communication that will take place between the Customer and the CMS_Staff. Whenever a new message is sent by the any of the two users the communication subsystem will store it on the Database and whenever a user might be viewing his/hers previous conversations, the communication subsystem will populate the data in the front-end from the database.

9. Subsystem for Complaint Handling

This complaint handling subsystem keeps the track of complaints stored in database. After registration of the complaint by the customer the complaint is assigned to an appropriate serviceman depending on the product id and the complaint is in opened state by default. The serviceman can choose to work on a particular complaint depending on the severity of the complaint and while working on the complaint the status of the complaint is in progress. And once the complaint gets resolved the status is updated to Closed.

4.3 Multi-Tenancy Implementation

Multi-tenancy provides the advantages of scalability at any layer it has been implemented i.e. The User Interface layer, Business logic Layer and the Data Layer.

Traditionally when a service is allocated to an organization a separate instance of the application is created and resources like the databases required for the system, the front-end files developed for the system and the files that implement the business logic are allocated by the cloud service provider specifically and separately for that particular system. This consumes lots of resources of the cloud as hundreds of systems are going to be using cloud and allocating same type of resources to these many systems duplication of resources if they are used for a similar type of systems eventually causing the wastage of resources. Hence in order to optimize the resource utilization we can use this Multi-tenant system which helps system to serve to similar systems all at once as the resources are shared between the systems of same type which also reduces the cost of the service to great extent as the resources are shared between many tenants [10].

I have implemented and displayed the concept of Multi-tenancy at the Database Layer. For this I have developed a scalable database schema which can support Multi-tenancy which helps the system to maintain a dynamic configuration of the database which is configured according to the requirements of the Tenant. The Multi-Tenant Database schema is explained below in detail.

The Overview of the Database Schema in the complaint management system is as shown below:

FIGURE 4.ER DIAGRAM FOR ENTIRE DATABASE SCHEMA

The above database schema can be superficially divided into two sub database schemas.

- A) The first part contains the tables which contain the data with respect to the business in our case it is complaint management system.
- B) The second part in the system consists of the tables that maintain the configuration of the required business tables which the organization desires to work with.

A) Database Schema for the first part is explained as follows:

FIGURE 5.ER DIAGRAM FOR THE SYSTEM

The above schema contains the following tables which are explained below and every table contains a field called `tenant_id` which is helpful for maintaining the configuration of that particular table with respect to a particular tenant.

❖ user:

The user table contains the authentication details of every user in the system like the Username and Password. It also contains the `role_id` of the user which states whether a User is customer or `cms_staff`.

❖ role:

The role table contains the details of the role a user is playing as in the role of the customer or a role of the `cms_staff`.

❖ customer

The customer table customer table inherits from the user table. It contains the details of the customer for that particular system, details of the customer like the `customer_id`, name etc.

❖ admin

The admin table contains the details of the admin of the system.

❖ cms_staff

The `cms_staff` table inherits from the user table and contains the details of the staff working in the system, details of the `cms_staff` like `cms_staff_id`, name etc.

❖ complaint

The complaint table consists the details of the complaint logged by the customer.

Details of the complaint like the complaint_id, who has sent the complaint who will be receiving the complaint. The message communication taken place between the customer and the receiver etc.

❖ organization

The organization table contains the data regarding the organizations using this system.

Each organization will be considered as a different tenant.

❖ product

The product table consists of the details of the product.

B) Database Schema for the second part is explained as follows:

FIGURE 6.ER DIAGRAM FOR THE MULTI-TENANCY SUPPORT

The above schema contains the following tables which are explained below. These tables actually store the configuration of the system tables with respect to the tenant id stored in the system tables.

❖ tenant_details

The **tenant_details** table mainly stores the name and id of the tenant. Every organization using the system will be having a different **tenant_id**.

❖ table_data

The **table_data** stores the ids and the names of each tables used in the system.

❖ field_details

The field table stores the field specific information like which table the field belongs to and the name of the field in that table.

❖ shared_fieldset

The share_fieldset table stores the values of the fields of the table which are shared. Usually these field are the fields which every organization of the system should contain.

❖ configurable_fieldset

The configurable_fieldset stores the values of the fields of the table which are configurable i.e. the fields that some organization may require and some may not.

❖ tenant_metadata

The tenant_metadata table stores the mapping of each tenant to table and each table to field mapping. The administrator of the system here decides which tenant will be having which tables and which tables will be having which fields.

So in this way a Multi-tenant database is created which could support multiple organizations as the database built can be shared between the systems of the same type.

Now for interacting with the Multitenant database the most important thing of all is the tenant_id as it will be unique with respect to an organization. Hence whenever a client is interacting with the system a tenant_id is sent by the client corresponding to the organization to which the user is customer. And as we are using the Rest web service whenever the client is initiating communication with the server the client uses the URL which stores the address of the server and the tenant's id through which only the corresponding values in the database schema is accessed by the client.

Below is the code snippet which has the URL of the server as well as the tenant_id is embedded in the URL. The tenant_id is 1 which is highlighted in yellow. So this is how communication takes place.

```
login.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        String uname,pwd;  
        uname=username.getText().toString();  
        pwd=password.getText().toString();  
  
        HttpClient httpclient = new  
DefaultHttpClient();  
  
        String  
url="http://10.0.2.2:8080/CMS/rest/users/1/"+uname;  
  
        HttpGet request = new HttpGet(url);  
        request.addHeader  
("Accept","application/json");  
        HttpResponse response;
```

And once the request/ control have reached the server, depending upon the parameters present in the URL appropriate function will be called of the respective tenant.

Below is the code snippet of two such functions.

```
// Return the list of users in the browser
@GET
@Path("/{tenantid}")
@Produces({ MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON })
public List<User> getUsers(@PathParam("tenantid") int
tenantid) {
    List<User> userObjs = new ArrayList<User>();
    userObjs.addAll(UserDao.instance.getAllUsers
(tenantid).values());
    return userObjs;
}

@GET
@Path("/{tenantid}/{username}")
@Produces({ MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON })
public User getUser(@PathParam("tenantid") int tenantid,
@PathParam("username") String username)
{
    return UserDao.instance.getUserByUsername(username,
tenantid);
}
```

In the above code snippet whenever a URL contains tenant_id at the end and if the URL has reached the appropriate file on the server. The function getUsers will be fired which will fetch the user of the corresponding tenant of which the tenant_id is supplied. And when the tenant_id is supplied with the username the function getUser will be fired and will fetch the corresponding user. This is how the Multi-tenant system works in my system.

4.4 Use cases

FIGURE 7.USE CASE DIAGRAM

Use cases	Description of use cases
Registration	Customer creates an account in the system in using this function.
Give complaints	Customer logs his/ her complaint using this module once he is authenticated.
Handles complaints	The complaint is handled by the serviceman.
Communication	With the help of this module the customer as well as the serviceman communicated with each other.
Access complaints	The complaint is directly routed in the complaint inbox of the service man.
Modifications in data base	The administrator is provided with full access to the system so that he can make necessary changes as when required.

TABLE 2.USE CASE DESCRIPTION

4.5 Description of Sequence

The sequence of the events that takes place in the system is as follows:

- Customer signs in.
- Validation of the user takes place through Authentication Subsystem
- Registration of the complaint takes place.
- The database stores the complaints.
- The complaint handling system assigns the complaint to an appropriate serviceman.
- Customer communicates with the customer for fast resolution of the complaint.
- Once the complaint is resolved the serviceman updates the status of the complaint.
- Administrator manages the system

5. Project Implementation

5.1 Generic work flow of the application

User interacts with the system using the mobile client. User enters the data on the mobile. For example: user enters the Username and Password on the mobile client for authentication. The values in the Username and Password variables are sent through the HTTP request-response mechanism that is governed by Rest calls. The Rest calls contain a URL to the server which is stored in a string on the client.

For example: `String url="http://10.0.2.2:8080/CMS/rest/users/1/"+uname;` The Client sends the appropriate data with the `tenant_id` of the organization that the user is customer of and the information is sent to the desired server using the above URL. The data standard used for communication is either in XML or JSON format. When the data reaches the server, the Resource modules created on the server first filters out the module that needs to be executed i.e. in our current case the username along with the `tenant_id` first reaches the application through this link `http://10.0.2.2:8080/CMS` and then going through the address embedded in the URL it eventually reaches the module where the Username is validated. The User resource module then calls the appropriate function mentioned in the data access object which opens a database connection and uses SQL queries for fetching the data i.e. in our case the user specific data will be fetched and stored in a user object and then the user object is returned to the resource module. After the reception the data in the user object will be converted in XML or JSON data format using the Jersey framework and is sent as a result string to the client. After the data is received on the client the data is parsed in a way that it extracts the data only in JSON format and stores it in appropriate variables. And then the data in the variables is

compared to for further operations. For example in our case all the user specific data is received, distributed and stored in variables such as the Password, role_id and user_id and then operations are performed accordingly for instance if the password is matched the user exists and depending on the role the respective home page will be opened otherwise a message of “Invalid Username/ Password” is displayed. This is how the general work data flow in the modules operates and is similar for other entities present in the system.

5.2 Client Implementation

FIGURE 8. MOBILE UI FOR LOGIN SCREEN FOR TENANT 1

FIGURE 9.MOBILE UI FOR EDITING PROFILE FOR TENANT 1

FIGURE 10. MOBILE UI FOR CHANGING PASSWORD IN A USER PROFILE FOR TENANT 1

FIGURE 11. MOBILE UI FOR BUILDING A USER PROFILE FOR TENANT 1

FIGURE 12.MOBILE UI FOR COMPLAINTS FOR THE CUSTOMER FOR TENANT 1

FIGURE 13. MOBILE UI FOR COMPLAINTS FOR THE CMS_STAFF FOR TENANT 1

FIGURE 14.MOBILE UI FOR REGISTERING A COMPLAINT BY A CUSTOMER FOR TENANT 1

FIGURE 15. MOBILE UI FOR VIEWING THE COMPLAINT AND ITS STATUS FOR TENANT 1

FIGURE 16.MOBILE UI FOR LOGIN SCREEN FOR TENANT 2

FIGURE 17. MOBILE UI FOR COMPLAINTS FOR THE CUSTOMER FOR TENANT 2

FIGURE 18.MOBILE UI FOR VIEWING THE COMPLAINT AND ITS STATUS FOR TENANT 2

Some of the Code snippets for implementing the client are as follows:

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Login extends Activity {
    EditText username;
    EditText password;
    Button login,signup;
    String result;
    int code;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        username=(EditText) findViewById(R.id.etUsername);
        password=(EditText) findViewById(R.id.etPassword);
        login=(Button) findViewById(R.id.bLogin);
        signup=(Button) findViewById(R.id.bSignup);
```

```

        login.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String uname,pwd;
                uname=username.getText().toString();
                pwd=password.getText().toString();

                HttpClient httpclient = new
DefaultHttpClient();

                String
url="http://10.0.2.2:8080/CMS/rest/users/6/"+uname;
                |
                HttpGet request = new HttpGet(url);
                request.addHeader
("Accept","application/json");
                HttpResponse response;

        signup.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                Intent browserIntent = new Intent
(getApplicationContext(),Signup1.class);
                //browserIntent.putExtra("username",
uname);

                //browserIntent.putExtra("userid", userid);
                startActivity(browserIntent);

            }

        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_login, menu);
        return true;
    }

}

```

```

public class Signup1 extends Activity {

    EditText username, password, cpassword;
    Button next, cancel;
    String result;
    int code;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup1);

        username = (EditText) findViewById(R.id.etUsername);
        password = (EditText) findViewById(R.id.etPassword);
        cpassword = (EditText) findViewById
(R.id.etConfirmPassword);

        next = (Button) findViewById(R.id.bNext);
        cancel = (Button) findViewById(R.id.bCancel);

        next.setOnClickListener(new View.OnClickListener() {

            else {

                String details= uname+":"+pass;
                Intent browserIntent = new
Intent(
                                getApplicationContext
(), Signup2.class);
                browserIntent.putExtra
("userdetails", details);
                startActivity(browserIntent);

                // already exists
            }

        } catch (Exception e) {
            // TODO: handle exception
        }

    }

});

```

```

        try {
            HttpPut request = new HttpPut
("http://10.0.2.2:8080/CMS/rest/customers");
            StringEntity params =new
StringEntity(json);
            request.addHeader("Content-Type",
"application/json");
            request.setEntity(params);
            HttpResponse httpResponse =
httpClient.execute(request);

            if(httpResponse.getStatusLine
().getStatusCode() == 201)
            {
                Toast toast = Toast.makeText
(getApplicationContext(), "Object Created!!",
Toast.LENGTH_LONG);
                toast.show();

                Intent browserIntent = new
Intent(getApplicationContext(), Login.class);

                startActivity
(browserIntent); }
            else
            {
                Toast toast = Toast.makeText
(getApplicationContext(), "Some Error!!",
Toast.LENGTH_LONG);

                toast.show();
            }
        }
    }
}

```

```

public class Edit_profile extends Activity {

    Button cpassword,updateprofile;
    String userid;
    String user;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_profile_cmss);

        userid=getIntent().getExtras().getString("userid");
        user=getIntent().getExtras().getString("username");

        cpassword=(Button)findViewById(R.id.bChangePassword);
        updateprofile=(Button)findViewById(R.id.bUpdateProfile);

        cpassword.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent browserIntent = new Intent
(getApplicationContext(),Change_password.class);
                browserIntent.putExtra("userid", userid);
                browserIntent.putExtra("username", user);
                startActivity(browserIntent);
            }
        });
    }
}

```


5.3 Server Implementation

Some of the code snippets used for development of the server is as follows:

```
public class SqlConstants {

    public static String getAllUsers="SELECT * FROM cmss.user
where tenantid=?";
    public static String getUser = "SELECT * FROM cmss.user
where username=? and tenantid=?";
    public static String insertUser = "INSERT into cmss.user
(username,password,roleid,tenantid) VALUES(?,?,?,?)";
    public static String deleteUser = "DELETE from cmss.user
where userid = ?";
    public static String getUserById ="SELECT * FROM cmss.user
where userid=? and tenantid=?";
    public static String updatePassword = "UPDATE cmss.user SET
password=? WHERE userid = ?";

    public static String insertCustomer ="INSERT into
cmss.customer
(fname,lname,email,phone,productid,userid,gender,addr,city,state,
country,zipcode) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)";
    public static String updateCustomer = "UPDATE cmss.customer
SET
fname=?,lname=?,email=?,gender=?,phone=?,productid=?,addr=?,city=
?,state=?,country=?,zipcode=? WHERE customer_id = ?";
    public static String getAllCustomers = "SELECT * FROM
cmss.user u, cmss.customer p where u.userid = p.userid and
u.tenantid=?";

import java.sql.Connection;
import java.sql.DriverManager;

// TODO: improve DBConnection class. Include connection pooling
public class DBConnection {
    final static String url = "jdbc:mysql://localhost:3306/";
    final static String dbName = "cmss";
    final static String driver = "com.mysql.jdbc.Driver";
    final static String userName = "root";
    final static String password = "root";
    |

    public static Connection getConnection() throws Exception {
        Class.forName(driver).newInstance();
        Connection conn = DriverManager.getConnection(url +
dbName, userName,
                password);
        return conn;
    }
}
```



```

public class User implements Serializable {
    private int userid;
    private String username;
    private String password;
    private int tenantid;
    private int roleid;

    public void setUserid(int userid) {
        this.userid = userid;
    }
    public int getUserid() {
        return userid;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getUsername() {
        return username;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getPassword() {
        return password;
    }
}

```

```

public class CmsStaff extends User implements Serializable{

    private static final long serialVersionUID = 1L;
    private int cmsStaffId;
    private String fname;
    private String lname;
    private String details;
    private String speciality;
    private int Product_id;
    private int userId;
    private int organization_id;|
    private String addr;
    private String city;
    private String state;
    private String country;
    private int zipcode;
    private int phone;
}

```

```

public void setAddr(String addr) {
    this.addr = addr;
}
public String getAddr() {
    return addr;
}
public void setPhone(int phone) {
    this.phone = phone;
}
public int getPhone() {
    return phone;
}

public String getCity() {
    return city;
}
public void setCity(String city) {
    this.city = city;
}
public String getState() {
    return state;
}
public void setState(String state) {
    this.state = state;
}
public String getCountry() {
    return country;
}
public void setCountry(String country) {
    this.country = country;
}

```

```

    public Map<String, User> getAllUsers(int tenantid) {

        Map<String, User> usersList = new HashMap<String,
User>();

        Connection con = null;
        ResultSet rs = null;
        PreparedStatement prest = null;
        try {
            con = DBConnection.getConnection();
            String sqlStatement = SqlConstants.getAllUsers;
            prest = con.prepareStatement(sqlStatement);
            prest.setInt(1, tenantid);
            rs = prest.executeQuery();
            if (rs != null) {
                List<User> resultList = fetchMultiResults
(rs);

                for (User user : resultList)
                    usersList.put(String.valueOf
(userCount++), user);

            }

        }

        public int putUserDetails(User user) {
            Connection con = null;
            ResultSet rs = null;
            PreparedStatement prest = null;
            int result = 0;
            User obj = getUserByUsername(user.getUsername
(),user.getTenantid());
            if (obj == null) {
                // insert
                System.out.println("calling insert user");
                try {
                    con = DBConnection.getConnection();
                    String sqlStatement =
SqlConstants.insertUser;
                    prest = con.prepareStatement(sqlStatement);
                    prest.setString(1, user.getUsername());
                    prest.setString(2, user.getPassword());
                    prest.setInt(3, user.getRoleid());
                    prest.setInt(4, user.getTenantid());

                    result = prest.executeUpdate();
                }
            }
        }
    }

```

```

public User getUserByUserid(int userid, int tenantid)
{
    User userObj = null;
    Connection con = null;
    ResultSet rs = null;
    PreparedStatement prest = null;
    try {
        con = DBConnection.getConnection();
        String sqlStatement = SqlConstants.getUserById;
        prest = con.prepareStatement(sqlStatement);
        prest.setInt(1, userid);
        prest.setInt(2, tenantid);
        rs = prest.executeQuery();
        if (rs != null) {
            userObj = fetchSingleResult(rs);
        }
    }

@Path("/users")
public class UsersResource {
    UriInfo uriInfo;
    Request request;
    String id;

    // Return the list of users in the browser
    @GET
    @Path("/{tenantid}")
    @Produces({ MediaType.APPLICATION_XML,
        MediaType.APPLICATION_JSON })
    public List<User> getUsers(@PathParam("tenantid") int
tenantid) {
        List<User> userObjs = new ArrayList<User>();
        userObjs.addAll(UserDao.instance.getAllUsers
(tenantid).values());
        return userObjs;
    }
}

```

```

    @GET
    @Path("/{tenantid}/{username}")
    @Produces({ MediaType.APPLICATION_XML,
    MediaType.APPLICATION_JSON })
    public User getUser(@PathParam("tenantid") int tenantid,
    @PathParam("username") String username)
    {

        return UserDao.instance.getUserByUsername(username,
tenantid);
    }

    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    public Response putUser(User user) {
        String result = "Track saved : " + user.getUsername();
        // return Response.status(201).entity(result).build();
        return putAndGetResponse(user);
    }

    private Response putAndGetResponse(User user) {
        Response res;
        |
        if (UserDao.instance.putUserDetails(user) != 0) {
            String result = "User inserted";
            return Response.status(201).entity(result).build
();
        } else {
            return Response.noContent().build();
        }
    }

```

6. Testing

In my project I have performed development and testing simultaneously i.e. whenever a new module was implemented I used to verify whether that module performs smoothly with the entire system.

Below is the type of testing performed on the system.

No.	Testing Method	Purpose behind using this testing
1.	Regression Testing	Whenever a new module or functionality was implemented independently it needs to be checked whether it is smoothly running in sync with the system. Hence for this type of testing regression testing was used.
4.	Integration Testing	It is necessary for the system to run properly as expected when different modules of the system are integrated. Hence for this type of testing Integration testing was used.

TABLE 3.TYPES OF TESTING METHODS USED

JMeter was the tool used for performing testing. And the performance of the web project is measured via HTTP requests handled by a System.

In JMeter testing tool, a test case is represented by a Thread group. And a thread group performs the testing on a particular amount of threads. Number of threads represents the number of users for which the test case will be executed and the testing is performed. In the below testing the test case has and the loop of 1000 that means the loop that work with HTTP transactions runs for 1000 times and the test case is ran for 10 threads which corresponds to 10 users issues this requests. The above values as described above is shown in the below screenshot.

FIGURE 19.THREAD GROUP SELECTION

For running the test we will need to setup the path for HTTP Requests and the port number in the JMeter tool so that we could perform testing on the HTTP requests made by the application and after configuring the path when we run the test we get the following results.

As per Results shown in the graph the threads (Users) are divided in samples and each sample is ran for some duration until the requests is executed successfully. And we get an average of 57 for the sample time and a standard deviation of 64 which states and the sample time fluctuates by this standard deviation. Also the Status field is checked with a green tick which shows that the HTTP requests were successfully handled by that particular thread sample.

View Results in Table							
Name: View Results in Table							
Comments:							
Write results to file / Read from file							
Filename		Browse...	Log/Display Only: ☐ Errors ☒ Successes		Configure		
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	23:46:12.076	Thread Group 1-7	HTTP Request	93		1395	93
2	23:46:11.983	Thread Group 1-3	HTTP Request	223		1395	222
3	23:46:11.984	Thread Group 1-2	HTTP Request	198		1395	198
4	23:46:11.983	Thread Group 1-1	HTTP Request	191		1395	191
5	23:46:11.963	Thread Group 1-4	HTTP Request	209		1395	209
6	23:46:11.979	Thread Group 1-6	HTTP Request	255		1395	255
7	23:46:11.982	Thread Group 1-5	HTTP Request	254		1395	254
8	23:46:12.177	Thread Group 1-8	HTTP Request	95		1395	95
9	23:46:12.252	Thread Group 1-4	HTTP Request	57		1395	57
10	23:46:12.259	Thread Group 1-6	HTTP Request	55		1395	55
11	23:46:12.250	Thread Group 1-1	HTTP Request	72		1395	72
12	23:46:12.249	Thread Group 1-2	HTTP Request	87		1395	87
13	23:46:12.262	Thread Group 1-5	HTTP Request	184		1395	184
14	23:46:12.280	Thread Group 1-8	HTTP Request	168		1395	168
15	23:46:12.378	Thread Group 1-2	HTTP Request	88		1395	88
16	23:46:12.317	Thread Group 1-4	HTTP Request	156		1395	156
17	23:46:12.242	Thread Group 1-7	HTTP Request	273		1395	273
18	23:46:12.276	Thread Group 1-9	HTTP Request	243		1395	243
19	23:46:12.519	Thread Group 1-7	HTTP Request	49		1395	49
20	23:46:12.327	Thread Group 1-1	HTTP Request	243		1395	243
21	23:46:12.476	Thread Group 1-4	HTTP Request	97		1395	97
22	23:46:12.522	Thread Group 1-9	HTTP Request	60		1395	60
23	23:46:12.243	Thread Group 1-3	HTTP Request	343		1395	343
24	23:46:12.319	Thread Group 1-6	HTTP Request	301		1395	301
25	23:46:12.581	Thread Group 1-1	HTTP Request	78		1395	78
26	23:46:12.596	Thread Group 1-3	HTTP Request	88		1395	88
27	23:46:12.579	Thread Group 1-7	HTTP Request	107		1395	107
☐ Scroll automatically? ☐ Child samples? No of Samples 12000 Latest Sample 7 Average 57 Deviation 64							

FIGURE 20. TESTING RESULTS IN TABULAR FORM

We can also view the above “View results in a table” in graphical form as shown below:

The graph shows the clear mapping of the different parameters like Average, Median, Deviation, and Throughput for the thread samples we have tested in graphical form. And we can see the Throughput which gives the number of requests / total time is 595.663 per minute. This Throughput is very good as the server can handle approximately 596 requests per minute for 10 users.

FIGURE 21. TESTING RESULTS IN TABULAR FORM

7. Conclusion

Since mobile device contains limited computing resources hence accessing websites efficiently or performing heavy computations on mobile phones is not an easy task unless we perform the computation on the server side. Therefore to overcome this limitation, cloud computing was used where in most of the computation and processing was done on the server which is hosted on the cloud where the problem of heavy computation won't arise as cloud computing will take care of it by managing the resources leaving the mobile to display the final outcome of the computation performed. Hence this system has successfully used cloud computing at the server end in combination with mobile computing at the front-end.

While working on building such cloud based system has helped me learn various technologies like the Amazon EC2 which is a web service that provides flexible processing capacity, RESTful architecture which is used for communication between the client and the server, and Android which has served as the platform for creating our client. It has also given me the knowledge of the concept of multi-tenancy which helps to scale the system across multiple organizations.

References

- [1] Adam Snook (2012). *Mobile Cloud Computing*. Retrieved from: <http://iitweb.bloomu.edu/blogs/groupprojects/2012/06/20/mobile-cloud-computing/>.
- [2] -"Cisco Visual Networking Index, "Global Mobile Data Traffic Forecast Update, 2011-2016", February 14, 2012, pp. 1-3.
- [3] Mednieks, Zigurd, G. Blake Meike, Laird Dornin, Masumi Nakamura, (2011). *Programming Android*. 1st ed. : O'Reilly.
- [4] (1999). *i-Sight Service and Complaint Management Software*. Retrieved from: [http://www.customexpressions.com/cex/cexweb.nsf/\(GetPages\)/6362FC56F53DB37185256FE8004F508B](http://www.customexpressions.com/cex/cexweb.nsf/(GetPages)/6362FC56F53DB37185256FE8004F508B).
- [5] Larry Aiken (2011). *Why Multi-Tenancy is Key to Successful and Sustainable Software-as-a-Service (SaaS)*, *Cloudbook Journal Vol 2 Issue 1*. Retrieved from: <http://www.cloudbook.net/resources/stories/why-multi-tenancy-is-key-to-successful-and-sustainable-software-as-a-service-saas>.
- [6] – Jingjing Tao; Xi Chen; , "Web Service Based Enterprise Mobile Information System," *Multimedia Information Networking and Security (MINES), 2010 International Conference on* , vol., no., pp.320-323, 4-6 Nov. 2010 doi: 10.1109/MINES.2010.216 Retrieved from: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5670833&isnumber=5670801>
- [7] (2011). *Complaints Management System*. Retrieved from: <http://www.workpro.com/complaints-management-system>.
- [8] Michael Armbrust, Armando Fox, Rean griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia (APRil 2010). A View of Cloud Computing. *Communication of ACM*, Vol. 53, No. 4.
- [9] Weiguang SONG, Xiaolong SU (2011). Review of Mobile cloud computing. *IEEE*, 978-1-61284-486-2/111\$26.00
- [10] - Sungjoo Kang; Sungwon Kang; Hur, S.:(May 2011) , "A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform," *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on* , vol., no.pp.462-467,doi:10.1109/CNSI.2011.56 Retrieved from: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5954363&isnumber=5954266>