UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

Comparative Study of REST and SOAP: Case of Registrar of Political Parties' Kenya

By

Christopher Wambua Kiama

P58/61788/2010

Supervisor: Mr. Lawrence Muchemi

August 2013

A Research Project Submitted in Partial Fulfillment of Requirements for the degree of Master of Science in Computer Science

Declaration

The Research Project as presented in this report is my original work and has not been presented for any other University Award. Materials of work done by other researchers are mentioned by clear reference and citations.

Signature: ……………………………

Date: ………………………………….

Christopher Wambua Kiama

P58/61788/2010

The Research Project has been submitted in partial fulfillment of the Requirements for the Degree of Master of Science in Computer Science at the University of Nairobi with my approval as the University supervisor.

Signature: ……………………………

Date: ………………………………….

Mr. Lawrence Muchemi

School of Computing and Informatics

Abstract

This project focuses on the comparative studies of Representational state Transfers (REST) and Simple Object Access protocol (SOAP), it demonstrates on how REST can perform better compared to SOAP in terms of consolidating data from different sources and presenting it in a common relational databases that can be used on a day to day activities. The data is presented in a web service in the form of HyperText MarkUp Language (HTML), Extensible Markup Language (XML) and JavaScript Object Notation (JSON).

It is also show how applications built on REST consume less memory and load fast compared to the ones build on SOAP, in addition to performance and scalability which is a key feature of REST.

They differ in context and usage; SOAP is a protocol while REST is architecture though they are all based on web service.

The comparative study of the two technologies is based on the data from different sources submitted by political parties to the registrar of parties.

Experiments were done to measure how much time and memory a SOAP and REST clients in PHP take in accessing the web service applications. The response messages and results showed that a REST client performs better on this than a SOAP client. Also experimented on scalability to determine how the two grow to accommodate increasing number of users, applications and systems.

Acknowledgement

My heartfelt gratitude goes to my Supervisor Mr. Lawrence Muchemi for his unrelenting intellectual guidance, motivation, suggestions and support at all the stages of my research project. Without him, this project would not have been possible. I would also wish to extend my sincere appreciation to my presentation panel members, Dr. E. Opiyo, Mr. A. Mwaura, Mrs. C. Ronge for their time, input and giving me ideas on the areas that required improvement on the project.

Special thanks to my parents Mr. and Mrs. Kiama who inspired and pushed me throughout the course and more so to complete the project

Above all, I owe this achievement to the Almighty God for the strength, gift of life and for enabling me pay my fees.

May God bless you all!

Dedication

I dedicate this Master of Science Research Project to my beloved parents Mr. and Mrs. Kiama and my daughter Sally Ngina. There is no doubt in my mind that without their inspiration and counsel, I couldn’t have completed this process.

Table of Contents

List of figures

List of tables

Definition of Acronyms

SOAP	–	Simple Object Access protocol
REST	-	Representational state Transfers
XML	-	Extensible Markup Language
ECK	-	Electoral Commision of Kenya
IEBC	-	Independent Electoral and Boundaries Commission
CSV	-	Comma Separated Values
HTTP	-	HyperText Transfer Protocol
HTML	-	HyperText MarkUp Language
JSON	-	JavaScript Object Notation
SQL	-	structured Query Language
URLs	-	Uniform Resource Locator
CRUD	-	Create, Read, Update, Delete
API	-	Application Programming Interface
RPC	-	Remote Procedure Call
ACL	-	Access Control List
URI	-	Uniform Resource Interface
WSDL	-	Web Service Definition Language
MVC	-	Model View Controller
WS	-	Web Service

Chapter 1:

1.0 Introduction

Registrar of political parties being the body charged with the responsibility of registering political parties before going for elections in Kenya, it is faced with a number of challenges which hinder it in determining the actual membership of each political party across the country as well as ensuring no multiple registration of members either in one or different parties.

To achieve its main objective of bringing sanity in an efficient manner, Registrar of political parties requires a technology that can collect data / information from political parties', stores data from different sources regardless format and transform it into a standard format to be used using web service applications that have improved performance, scalability through an interface that is easy and quick to maintain when need arises.

There are quite a number of technologies that can achieve these objectives including REST and SOAP,

REST stands for Representational State Transfer and marks a software architecture pattern (in contrast to SOAP which is a protocol). It relies on a stateless, client-server, cacheable communications protocol and in virtually all cases; the HyperText Transfer Protocol (HTTP) protocol is used.

REST being an architecture style for designing networked applications, its more preferred rather than using complex mechanisms such as CORBA, Remote Procedure Call (RPC) or SOAP to connect between machines and simple HTTP is used to make calls between machines (Singh, 2009).

It is a simple HTTP-based protocol that enables users to contact the message broker through a Web browser by navigating to appropriately formatted Uniform Resource Locator (URL) or by posting HyperText MarkUp Language (HTML) forms based on a subset of the HTTP protocol.

HTTP is the only supported transport and as result REST performs operations on resources through it, including the so called CRUD operations (Create, Read, Update, Delete) and query resources in an easy and natural way. REST uses the HTTP protocol to identify, query and manipulate resources in a computer network by using information provided by the HTTP protocol (Maven, 2011).

The HTTP URL is responsible for allocating a resource and can contain parameters in its query part. For instance the HTTP method is used for choosing the right action to be performed on the resource, Request headers provide Meta data for accessing a resource and the request body provides the resource input in case of POST and PUT requests.

SOAP stands for Simple Object Access Protocol and brings its own protocol and focuses on exposing pieces of application logic (not data) as services. It exposes operations and is focused on accessing named operations through different interfaces.
Though SOAP is commonly referred to as "web services" it has very little if anything to do with the Web but REST provides true "Web services" based on URIs and HTTP.

SOAP is not made for resource constrained mobile devices but it is for fixed network. The SOAP messages has heavy payload. In contrast, the REST messaging framework has lightweight payload which is suitable for mobile as well as cellular network. REST identifies the service resources by single URL only (Kishor Wagh, 2012).
SOAP request uses POST and require a complex XML request to be created which makes response-caching difficult While RESTful APIs can be consumed using simple GET requests, intermediate proxy servers / reverse-proxies can cache their response very easily.
SOAP consumes more bandwidth because its response could require more than 10 times as many bytes as compared to REST while REST consumes less bandwidth because its response is lightweight.

SOAP web services always return XML data While REST web services provide flexibility in regards to the type of data returned.
SOAP has heavy payload as compared to REST while REST is definitely lightweight as it is meant for lightweight data transfer over a most commonly known interface, - the URI
In SOAP, Client-Server interaction is tightly coupled while REST, Client-Server interaction is loosely coupled.

In the development of prototypes using REST and SOAP, a research is done to determine the best web service in terms of scalability, performance, extensibility in solving the main problem of the registrar of political parties.
Extensibility is the ability of the prototype system to allow and accept significant extension of its capabilities, functionality, enhancement for the purpose of meeting future needs and

significantly changing requirements without major rewriting of code or changes in its basic architecture. (Christoph Becker, 2009)

Scalability is the ability of the prototype architecture to grow to accommodate increasing number of users, applications and system. (Jon, 2007)

1.1 Problem Statement

Before the Registrar of political parties was introduced in by an act of parliament, the political parties used to provide their membership data using either printed documents or just soft copies which were not harmonized. This used to bring a lot of inconsistencies and false information to the former Electoral Commission of Kenya (ECK) as many members had multiple registrations across multiple parties which had a lot of redundant information. This used not to provide a true picture of parties popularity across the country and some of them got registered with only regional representation.

In this case the registrar of political parties, they had no way to analyze the data received and determine if it really represents a true picture of the situation.

This is really a great challenge as the membership data which the parties provide for them to be registered is not a true picture of some parties' representation, and some members end up being registered in more than one political party. Other members who end up being registered are either not aware or they are dead.

This usually brings a lot of difficulties and some parties end up being registered with no representation in some parts of the country.

So far no researches has been done to demonstrate on how web service applications can be used to solve the problem of data consolidation from different sources and present it in a common database format for use with emphasis on performance, scalability and extensibility.

1.2 Justification

This project is focusing on bringing sanity to the Registrar of political parties by ensuring they get a true representation of each individual political party in the country using web services. This will be achieved by having every political party establish a data store either using the normal standard databases, spreadsheets, comma separated values (CSV) or any other format that supports Open Database Connectivity.

The Registrar of political parties will be provide to the political parties a link and credentials to post data from their data stores which they will now be consolidated to a normal structured query language format for analysis and presenting the required data through a common browser using REST and SOAP technologies.

1.3 Objectives of Project

The objectives of this project are:

1. Design frameworks that Combine Registrar of political parties' data from different data sources and use REST and SOAP frameworks to present it through a browser.
2. Develop prototypes that use simulated political parties data based on HTTP web service on REST and SAOP principles that demonstrates the framework in objective one.
3. Evaluate and compare the prototypes based on REST and SOAP frameworks.
4. Recommend the most suitable framework for political parties' problem in terms of performance, scalability extensibility and maintenance.

1.4 Project outcomes and their significance to Registrar of Political parties

The system prototype is a web based interface which provides a facility for users (registrar of political parties) to upload their information and have it transformed to a standard format for storage in MYSQL database. It will also feature a representation of the parties' data in various data formats (HTML, XML and JSON) for consumption by other organization (NGO and international bodies and also the parties themselves).

Level of Complexity: Average

The project will ensure registrar of political parties have better maintained (Create Read Update and Delete) data that can be shared to the public and any other interested parties to add value and transparency in their operations.

1.5 Research Questions

To perfectly manage the current problems with the registrar of political parties, the research will be guided by the following questions

a. How does REST compare with SOAP in terms of scalability with the number of users (threads).

b. To what extent does REST increases performance compared to SOAP.
c. To what extend does REST interface and prototypes compare with SOAP in extensibility and maintainability.

1.6 Scope of the project

This research intends to design frameworks and come up with prototypes that combines and works with data using REST and SOAP web services. This will enable and guide the registrar of political parties in analyzing and determine presentation of each political party in every county. The consolidation of data will be limited to SQL databases using TXT, CSV files and spreadsheet formats.

1.7 Assumptions of the research

a. All political parties have a data store for their data and are willing to present to the register of political parties on demand.
b. Political parties can trust a third party with their data by providing credentials to their database.

Chapter 2: Literature Review

2.0 Introduction

Web Services is a modern and popular technology in managing institutional data and systems. To manage the Web service a list of protocols and technologies related to Web Services grows every day, but REST and SOAP are probably the most popular. SOAP being the first technology it was rapidly becoming the standard protocol for accessing Web Services using XML messages to exchange information across endpoints, and provides several advantages over other binary protocols (Singh, 2009).

With advancement of the technology, Web Services are the key point of Integration for different applications belonging to different Platforms, Languages, and systems. To achieve this, Representational State Transfer (REST) which basically means that each unique URL is a representation of some object was brought into action so as to outdo SOAP.

REST APIs haven't been around for long and their APIs are definitely modern for creating most of the web services (Singh, 2009).

The main advantages of REST web services are:

- Lightweight – not a lot of extra xml markup
- Human Readable Results
- Easy to build – no toolkits required

2.1 Differences between REST and SOAP web services

1. SOAP is a protocol based on XML message, while REST is an architectural style
2. SOAP uses WSDL for communication between consumer and provider, whereas REST just uses XML or JSON to send and receive data
3. SOAP invokes services by calling RPC method, REST just simply calls services via URL path
4. SOAP doesn't return human readable result, whilst REST result is readable with just plain XML, JSON or HTML
5. SOAP is not just over HTTP, it also uses other protocols such as SMTP, FTP and REST is over only HTTP.

2.2 How REST compares with SOAP

1. **Maintainability and extensibility -** Compared to SOAP, Restful applications are easy to maintain and extend as you only need to change the configuration files and the REST files for any changes you need to make, and also REST is able to process requests and responses in a very short time. SOAP requires any changes to be done in the WDSL which is complex compared to REST and it requires client and server files to be created and modified to match with the WSDL.

2. **Data Consolidation** – Both REST and SOAP will in the end be seen as only channels through which we post data to the consolidative database where we will store the data. SOAP and REST will use the HTTP POST method to pull data from clients and store in the database. As for SOAP, a WDSL document with this post method will have to be created and will require subsequent updates whenever there is a slight change in structure as it always will be referenced by the service to respond to the client, compared to REST as all the responses will come from the service.

3. **Performance and scalability -** REST uses web's semantics instead of trying to channel its requests via XML, so RESTful web services are generally designed to use cache headers, so they work well with the web's standard infrastructure like caching proxies and even local browser caches thus increased performance. Compared to REST, SOAP uses HTTP, it does not take advantage HTTP's supporting infrastructure as SOAP-based reads can't be cached.

2.3 Challenges facing Registrar of Political parties

As a result of personal and community interests registrar of political parties faces great challenges during parties' registrations:-

a. **Political parties' memberships** – unprincipled party officials register people / citizens to their parties without their consent when they get their personal information from unknown sources. This particularly happens to people who are public figures and the parties need to show they have representation across the country. At times an individual is found to be registered in more than one political party

To resolve this challenge the Registrar of political parties introduced a link in their website which the public can enter the national identity number for them to confirm if their name has been used unknowingly. http://www.iebc.or.ke/rpp/

In this project REST and SOAP web service technologies will comparatively demonstrate how data from all political parties can be consolidated into common structured database format for analysis and reporting for the registered members. This will drastically reduce double registrations and the culprits found can be black listed from holding political and public offices.

b. **Hate Speech** – with invent of social media (face book, twitter etc) people across the world are using them and some post political sentiments, that can cause violence in the country as they are fast circulated across the world.
Also politician during political rallies use inciting statements that can cause fights in the country; to resolve this, government have introduced zoom recorders assigned to police so that they can record content for later analysis to determine if it’s really hate speech. Watchdog personnel have also been put in place to check on the content from the media i.e. digital (face book, twitter etc) and voice (radio stations and TV).
With the RESTful interface because of its less bandwidth consumption all the audio clips will be uploaded and the registrar of political parties’ personnel will be able to listen and audit them online.

2.4 Contemporary Issues of Kenyan Voting Systems

In the past, registration of political parties and its members has been manual system; this stretched all through voter’s registration until the final voting process. There has been no a clean way to determine legitimate members of the registered political parties, this resulted to problems which brought serious conflicts in the country.
It was not easy to determine the true party membership and representation across the country as each party used to provide their list of members and there was no genuine way to determine the truth from the list as it was not easy to go through all the records provided.
Incase Kenyans find their names in any political party through: http://www.iebc.or.ke/rpp/, they can report to the registrar of political parties through reg.pol.party@gmail.com for action. With the help of web service technologies, REST and SOAP registrar of political

parties can confirm the membership status for different political parties by consolidating the data in a common standard database format.

To achieve this each political party will be given a link to the web service system to post their list of members online, so that the issue of double registrations is avoided and they can confirm the list from each political party and also compare across multiple lists.

2.5 Sources of political parties membership Information

Previously political parties used to provide membership list to registrar of parties in any format (hand written papers, word processed formats or spreadsheets), which in most cases was not easy and efficient to determine the truth of the data.

Present the registrar of political parties have given a link to all political parties which they can use to post their list of members to their database online which is very cumbersome to parties as they have to type name by name and not upload automatically.

With the web service technologies (REST and SOAP), all the political parties' will present their data in specified formats for consolidation to a standard format that can be used and made available to members who have accounts with registrar of political parties. This will enable Kenyans to register with political parties which are genuine and honest as they have information about the parties.

2.6 Mobile Use

The IEBC has been very vibrant in ensuring Kenyans get real time information on request through either the mobile technology or the electronic media online through their site.

At the moment Kenyans can check their registration details by sending their national identity card number to 15872 through SMS (Short Message Service)

With the web service technologies based on REST and SOAP, Kenyans will be in position to know their membership status with the political parties as all the data will be readily be available and consolidated in a common pool. In that case a mobile technology can be put into use as almost 90% of Kenyans now have access to mobile phones.

2.7 Case Study: Yahoo Web Services based on REST approach

Yahoo provides a variety of Web services at http://developer.yahoo.net/, from searching the web to interfacing with Flickr (Yahoo's photo-sharing community). Using REST, you can easily add integration for them within a Web page or larger application. Although several services are available, this case study demonstrates how to perform a Web search (http://developer.yahoo.net/search/web/) and how to perform a product search (http://developer.yahoo.net/shopping/V1/productSearch.html). Using the ideas and techniques presented in the examples, it is quite easy to apply them to access other offered services.

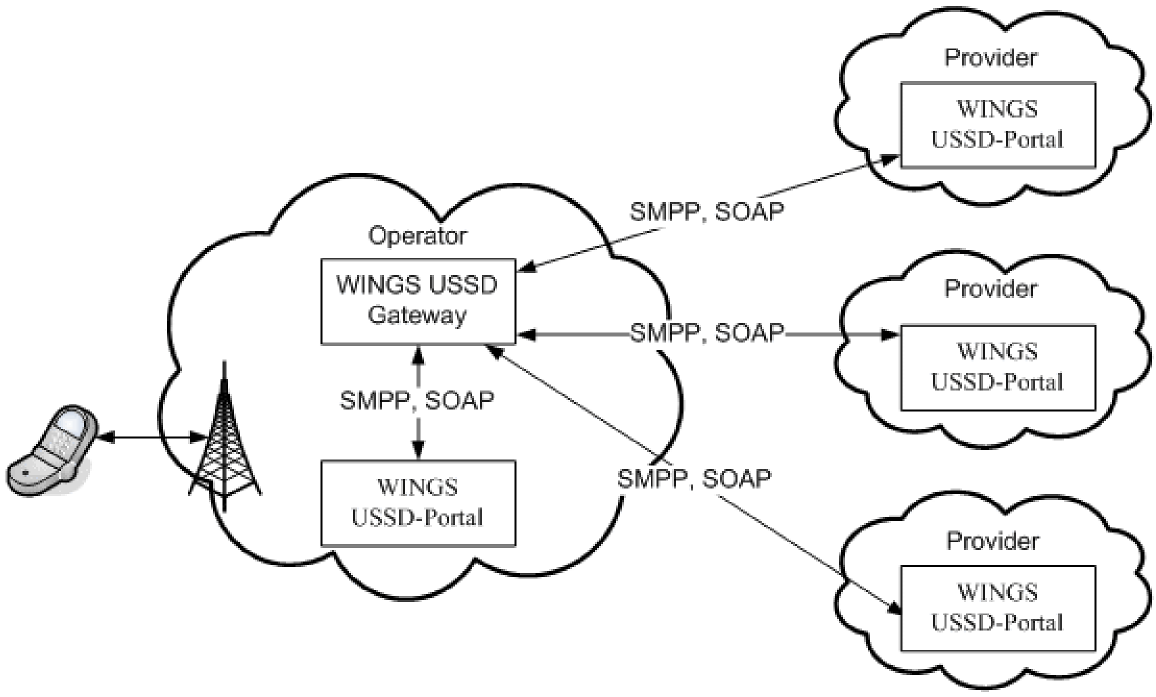
The Flickr service requires its own API key. Although a lot of functionality does not require authentication, to upload photos you must also register for an API secret key (at http://www.flickr.com/services/api/registered_keys.gne) once you have your API key you are now in position to register and share your profile for pictures.

2.8 Use of SOAP in the Mobile Technology

SOAP has been used in Unstructured Supplementary Services Data (USSD) service, which allows high speed interactive communication between mobile subscribers and applications across Interfaces to SOAP, XML, LDAP services

Basically the way USSD works is that you send some cryptic number like *333# to your mobile servers who in turn forward the request via various protocols mostly HTTP or SOAP but not limited to these.

The diagram below shows a USSD Gateway that provides the connection to the IP-based network which enables connection over the internet. The gateway contains application modules that enable development of applications based on an API or Soap interface.



http://www.wings-solutions.com/solutions/operator/ussd_center.php

2.9 Building a web service for political parties on SOAP

This will allow other applications to easily access the same data, also separates the data extraction from the data source and the application itself. For instance if you were storing the data in a MySQL database but later decided to move it to other database format. In this scenario your application wouldn't know the difference. Its calls to the Web Service remain unchanged.

To provide a political parties' resource service you will have to present the political party name and symbols to be stored in a database. This is not going to concentrate on the storage mechanism or how to obtain the political parties data.

To create the web service first I will create the SOAP server using script that will fetch the data from the database and then deliver it to the Client. Using the NuSOAP library, same Server script will also create a WSDL document for us.

To achieve this, I will create a function that will fetch the data we want. The name given to the function will be the name that is used when the Client contacts the Server.

```
<?
```

```
function getPartyMembers($symbol) {

   mysql_connect('server','user','pass');
   mysql_select_db(rp);
   $query = "SELECT party_members  FROM political_parties "
          . "WHERE party_symbol = '$symbol'";
   $result = mysql_query($query);

   $row = mysql_fetch_assoc($result);
   return $row[‘party_members ‘];
}
?>
```

The code below turns the function getPartyMembers into a Web Service. We have to include the NuSOAP library, instantiate the soap_server class and then register the function with the server.

1. The first thing necessary is to simply include the NuSOAP library.require('nusoap.php');
2. Next, instantiate an instance of the soap_server class.
   $server = new soap_server();
3. The next line is used to tell NuSOAP information for the WSDL document it is going to create. Specifically we specify the name of the server and the namespace, in that order.

```
$server->configureWSDL('partyserver', 'urn:partymembers');
```

Next, we register the function we created with the SOAP server. We pass several different parameters to the register method.

First is the name of the function we are registering.

The next parameter specifies the input parameters to the function we are registering. The implementation is in form of an array. The keys of the array represent the names of the input parameters, while the value specifies the type of the input parameter. This specify the types of input and return parameters with the designations of xsd:string and xsd:decimal.

The third parameter to the register method specifies the return type of the registered function. As shown below, it is fashioned in the same way as the last parameter, as an array.

The next two parameters specify the namespace we are operating in, and the SOAPAction.

```
$server->register("getPartyMembers",
          array('symbol' => 'xsd:string'),
          array('return' => 'xsd:decimal'),
          'urn: partymembers',
          'urn: partymembers# getPartyMembers');
```

Now, we finally finish it off with two more lines of code. The first simply checks if $HTTP_RAW_POST_DATA is initialized. If it is not, it initializes it with an empty string. The next line actually calls the service. The web request is passed to the service from the $HTTP_RAW_POST_DATA variable and all the magic behind the scenes takes place.

```
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA)
             ? $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
```

Creating a SOAP Client to access our Server does not necessarily need to be a PHP Client. The SOAP Server we just created can be connected to by any type of Client, whether that be Java, C#, C++, etc.

To create the SOAP Client, needs to do are three things.

1. First, include the NuSOAP library. This is done just as it was for the Server.
   require_once('nusoap.php');

2. Secondly, we need to instantiate the soapclient class. We pass in the URL of the SOAP Server we are dealing with.

```
$c = new soapclient('http://localhost/partiesserver.php');
```

3. Last make a call to the Web Service. The one caveat is that the parameters to the Web Service must be encapsulated in an array in which the keys are the names defined for the service.
   ```
   $members = $c->call(' getPartyMembers ', array('symbol' => 'image'));
   ```

Now, here is the completed Client script, which I have saved in a file named partiesclient.php.

```
<?php
require_once('nusoap.php');

$c = new soapclient('http://localhost/patriesserver.php');

$members = $c->call(' getPartyMembers ',
            array('symbol' => 'image'));

echo "The Members for 'Ford Kenya' is $members.";

?>
```

Chapter 3: Methodology

3.0.0 Introduction

This chapter presents the analysis, design, implementation and testing of the project prototypes for the registrar of political parties using both the REST and SOAP frameworks. It outlines the database structure in MYSQL in form of tables and fields where the political parties data is uploaded to using a feature that import data from CSV / XLS and TXT into the database tables.

The chapter outlines the algorithms and frameworks to be developed towards the achievement of the project objectives as well as addressing the research questions.

Methodology describes guidelines for solving a problem, with specific components e.g. phases, tasks, methods, techniques and tools to be used. It can be defined also as follows:-

1. The analysis of the principles of methods, rules, and postulates employed by a discipline.
2. The systematic study of methods that are, can be, or have been applied within a discipline.
3. The study or description of methods.

A methodology can be considered to include multiple methods, each as applied to various facets of the whole scope of the methodology.

To develop and implement the Web service prototypes for the registrar of political parties; the following phases are considered suitable for the Web Service implementation lifecycle: requirements, analysis, design, implementation, test, and deployment.

3.1.0 Requirements Phase

This brings an understanding on the issues affecting the registrar of political parties then translating them into REST and SOAP web service requirements in terms of the features, the functional and non-functional requirements, and the constraint that can help in solving the registrar of political parties' problem. Again this phase provides an opportunity for identifying the Web Services that can be used to solving the problems.

At the moment the registrar of political parties perform parties membership registrations manual, there is no way to determine how many times a member is registered across different political parties as the officials just submit their documents in non standard formats and the registrar of political parties has no mechanisms to know how many times a member is registered.

In the past the registrar of political parties introduced a link and gave it to the political parties for them to type all their data to the system, this is not only a tedious process but also an hard way to determine how many times a member is registered and to how many parties as the link was specific to individual parties.

3.2.0 Analysis Phases

This phase refines the requirements further and translates the requirements into conceptual models. Architecting analysis is done to define high-level structure and identify the Web Services interfaces contracts.

In this phase the following will be achieved: - Analyzing the granularity of Web Services interface contracts, selecting the web service (REST and SOAP) technology platform for implementation framework, Defining Web Services candidate architecture and finally Identify architectural components to be exposed as WSs and specify major information exchanged with client.

To eliminate the problems Registrar of political parties' faces when doing follow-ups on political parties' membership and representation across the country, a proper analysis is required on how they present their data to Registrar of political parties and how it analyses it. This was done using the following techniques:-

a. **Observation** – this involved gaining access to some information one of the political was to submit to the Registrar of political parties. The information was available in files (manual and electronic) and was to be typed one by one to through a link that was provided by the registrar office.
b. **Documentation** – the registrar of political parties had provided a link through the IEBC website for the public to confirm, to which political parties they are registered as members. A lot of inconsistencies were found as quite a number of people found their names appearing as legitimate members without their consent.

The proposed system will be based on an API and has a feature to allow the political parties to upload their membership data to the registrar of political parties' database. The documents to be uploaded need to be in the form of CSV / XLS, TXT format which the registrar will specify to the parties when collecting information from the members so that they can conform to the acceptable format for the prototype to be developed.

The uploaded information will be represented in the form of tables which can now be fetched and displayed using any of the following formats JSON, XML, HTML for analysis or references by the registrar of political parties.

3.2.1 Data sources and preparation

The prototype is based on political parties' simulated data, which is presented in CSV, TXT and XLS formats.

The following tables show the simulated political parties data, which forms just a percentage of the data kept by the registrar of political parties.

a. Parties table

The table below shows the design view of the simulated political parties' data

Attribute Name	**Data Type**
Id	int(5) (PK)
party_name	varchar(40)
party_patron	varchar(40)
party_secretary	varchar(40)
party_logo	varchar(40)
party_office	varchar(40)
Dcreated	Timestamp

TABLE 3.1: DESIGN VIEW OF THE SIMULATED POLITICAL PARTIES' DATA

The table below shows the datasheet view of the simulated political parties' data

ID	Party_name	Party_Patron	Party_Secretary	Party_Logo	Party_Office	DCreated
1	Wiper	Chris	Kiama	Umbrella	Machakos	12/2/2009
2	Sisi kwa Sisi	Kiama	Chris	Hand shake	Nairobi	2/12/2009

TABLE 3.2: DATASHEET VIEW OF THE SIMULATED POLITICAL PARTIES' DATA

b. Members table

The table below shows the design view of the simulated political party members' data

Attribute Name	**Data Type**
Id	int(11) (PK)
National_id	varchar(40)
Name	varchar(40)
Position	varchar(40)
date_joined	Timestamp
party_id	varchar(40)

TABLE 3.3: DESIGN VIEW OF THE SIMULATED POLITICAL PARTY MEMBERS' DATA

The table below shows the datasheet view of the simulated political party members' data

ID	National ID	Member Name	Position	Date Joined	Party ID
1	14730531	Chris Kiama	Member	12/3/2016	3
2	20086546	Sally Ngina	Secretary	3/12/2009	4

TABLE 3.4: DATASHEET VIEW OF THE SIMULATED POLITICAL PARTY MEMBERS' DATA

3.3.0 Design framework Phase

This phase deals with detail design of REST and SOAP Web Service prototypes. This is where the web service prototypes interfaces are refined further. The interactions between the web services (REST and SOAP) and the client, e.g. asynchronous/synchronous or RPC / document are considered.

The research being a comparative study of rest and soap, two prototype designs are designed that will be used to implement them and comparatively compare the results of the experiments' done.

3.3.1 SOAP Design framework

After getting the actual information relevant to the registrar of political parties and carefully analyzing the content, what follow is the design frameworks of how the project will be implemented.

SOAP services are typically defined using the Web Services Description Language (WSDL) as they require a library called NuSOAP from where to access most of its resources and features.

This can readily be downloaded from http://sourceforge.net/projects/nusoap/ or developed and modified every time depending on the modifications that you need to make on it the WSDL.

The diagram above shows the model framework of SOAP

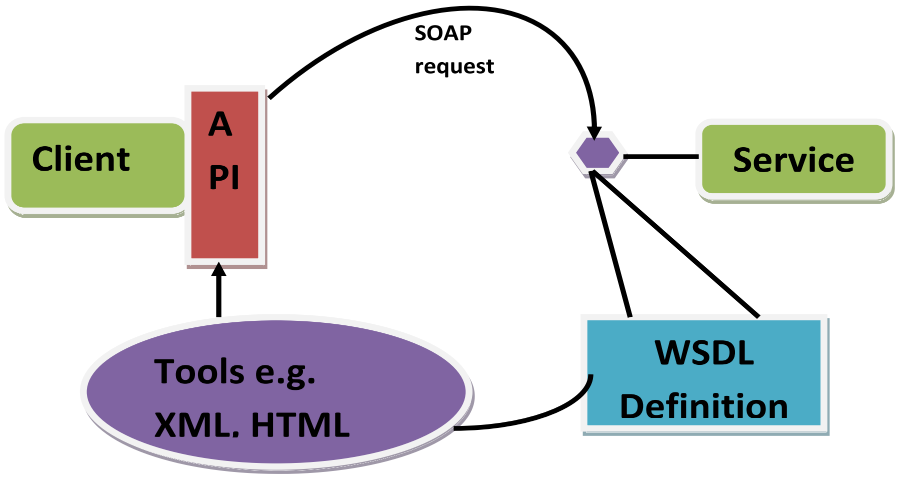
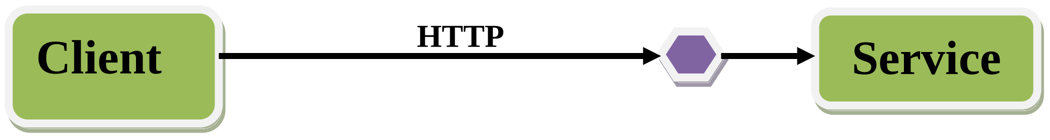


FIGURE 3.1: MODEL FRAMEWORK OF SOAP

3.3.2 REST Design framework

When designing the REST framework, there is no standard definition language for defining RESTful interfaces (Chappell, 2009). RESTful service can use XML, JavaScript Object Notation (JSON), and other formats to match different performance requirements as any HTTP client library may be used to interact with the Portal REST server. Any of the options below can be used as they provide a model framework of how:-

Option 1: Clients write raw HTTP calls



Option 2: A RESTful service provides a client library – Clients see methods with parameters

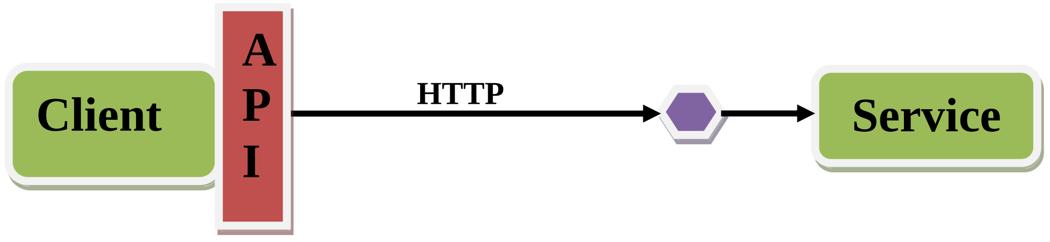


FIGURE 3.2: MODEL FRAMEWORK OF REST

3.4.0 Prototype Design

The project being a comparative study of REST and SOAP, two prototypes will be developed that compares and answers the research questions.

The prototype will involve the input (upload) of political parties data through a standard browser to a MYSQL database, then try to access the data using REST and SOAP frameworks and present it through the browser in HTML, XML and JSON.

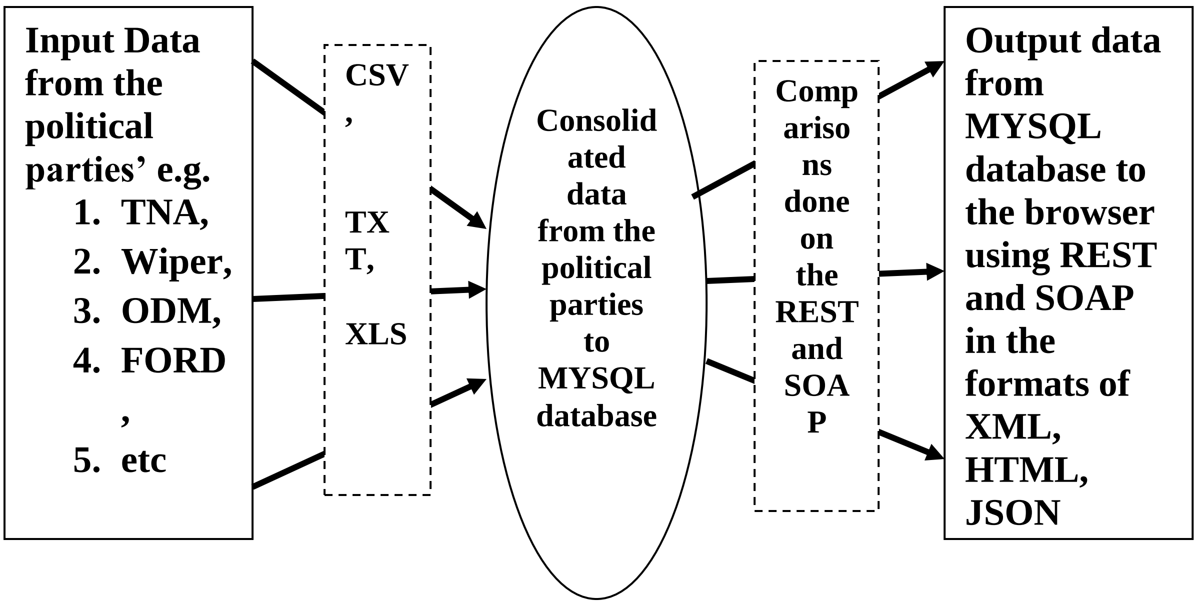

Input Data from the political parties' e.g.
1. TNA,
2. Wiper,
3. ODM,
4. FORD ,
5. etc
CSV , TXT, XLS
Consolidated data from the political parties to MYSQL database
Comparisons done on the REST and SOAP
Output data from MYSQL database to the browser using REST and SOAP in the formats of XML, HTML, JSON

FIGURE 3.3: MODEL FRAMEWORK OF THE TWO PROTOTYPES

The proposed system on REST will be developed using the Code Igniter, which is an open source rapid development web application framework for building dynamic applications with PHP. It very well supports REST by providing a rich set of libraries for the needed tasks, as well as a simple interface and logical structure to access libraries.

The prototype will be based on Model View Controller (MVC) which is a design paradigm that breaks application's interface, into three parts: the model, the view, and the controller which maps the traditional input, processing, output roles into the GUI system (Marston, 2004).

The diagram below shows a simple structure of MVC:-

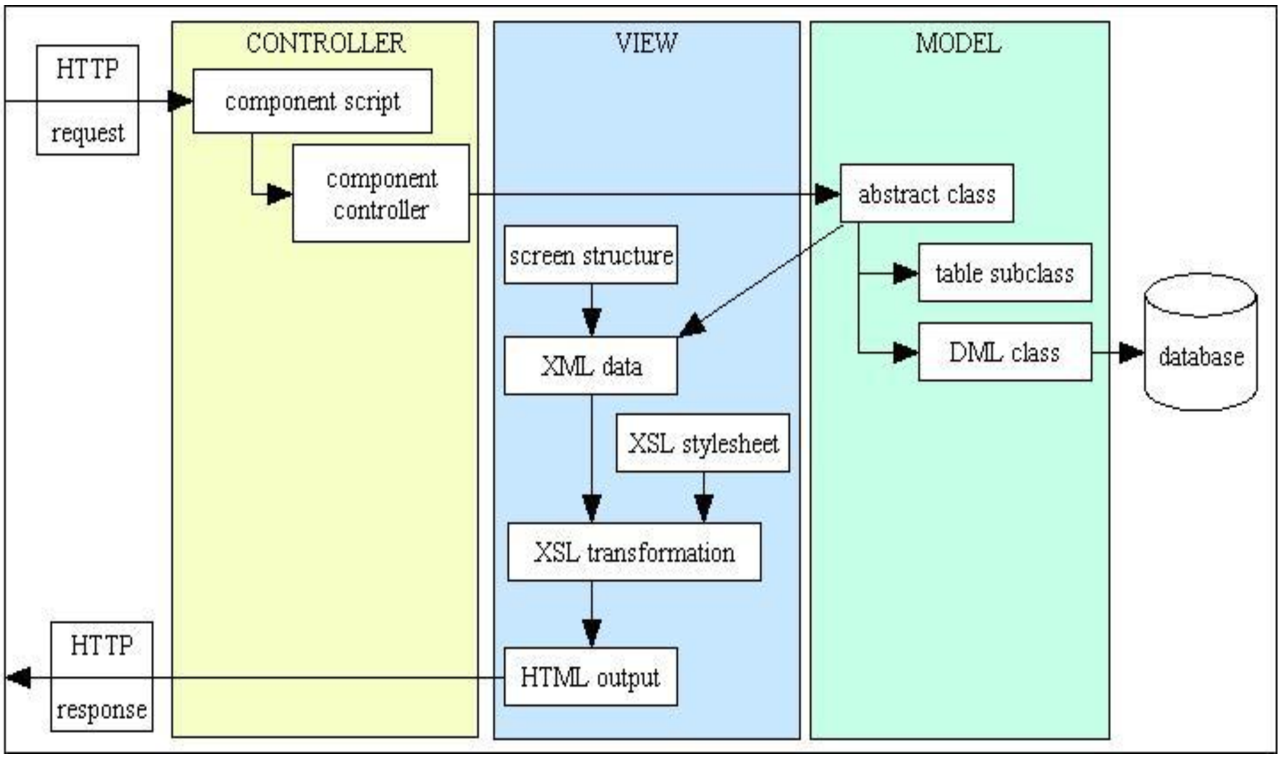


FIGURE 3.4: MVC DESIGN PARADIGM FOR THE TWO PROTOTYPES

i. Model

The model represents data in the form of MYSQL contained in database called RP. It manages the activities and data of the registrar of political parties on membership, responds to requests for information about its state and responds to instructions to change state depending on the need.

The model represents and dictates registrar of political parties’ data and the rules that govern access to and updates of the data.

ii. View

The view is a form of visualization of the state of the model i.e. it queries the database and decides on the form of visualization / display.

The view also manages the graphical and textual output to the portion of the database to be displayed which is allocated to its application. The view will generate HTML / XML / JSON as the standard formats for displaying the queried data from the database.
The view will render the contents of a model (registrar of political parties data) and specifies the format that data should be presented.

iii. Controller

The controller interprets the mouse and keyboard inputs from the user either through typing or uploading the contents from other file formats, making the model or the view to change as appropriate depending on the request.
A controller forms the means by which the user interacts with the application by accepting input from the user and instructing the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response.
The controller translates interactions with the view into actions to be performed by the model. This prototype (system) being a Web application the interactions appear as HTTP GET and POST requests.

3.5.0 Screen Design

The prototypes have the common browsers as the client application where the user enters the URL for the prototypes, then he can present the username and password to access the data / information about the parties data.

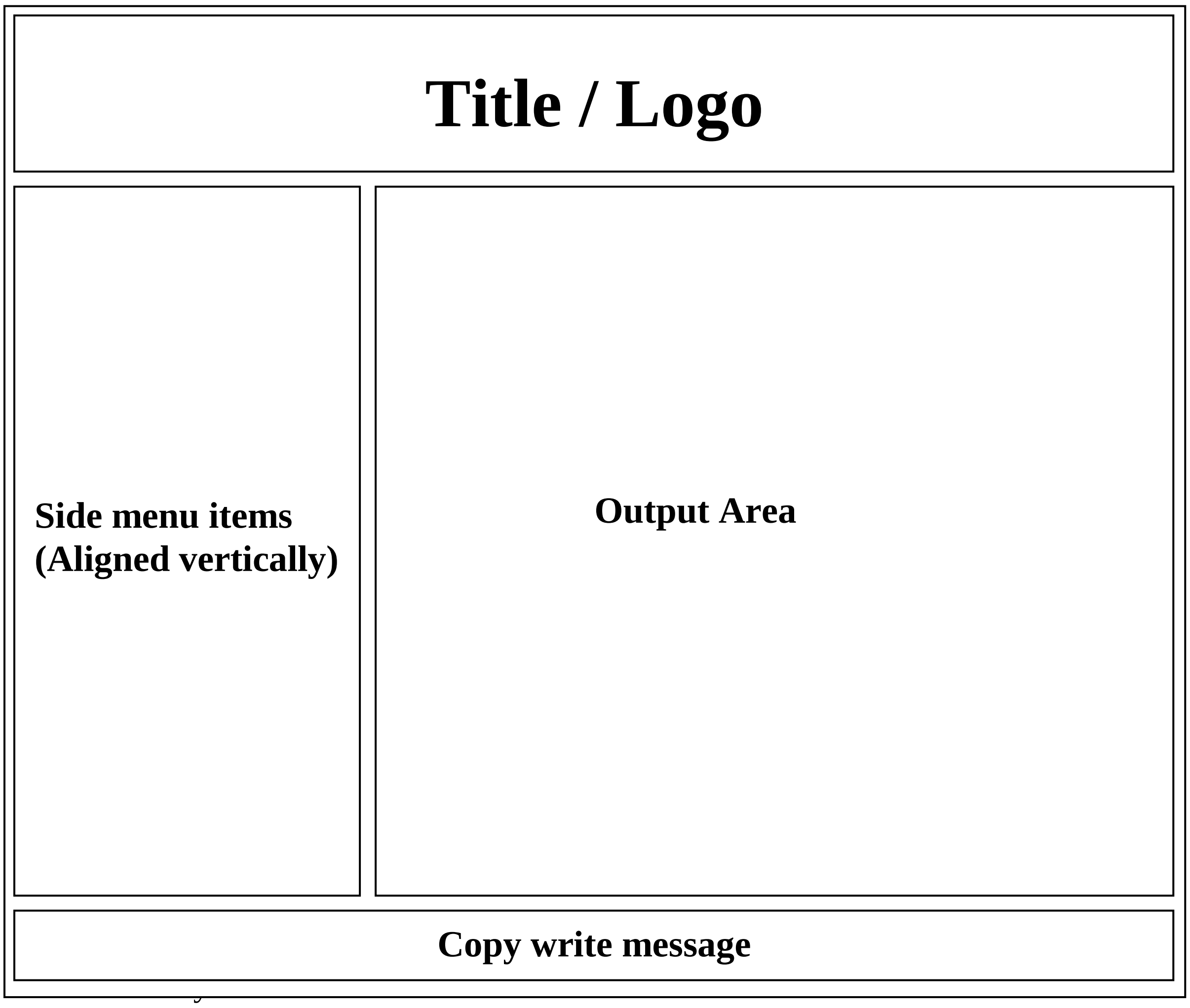


FIGURE 3.5: OUTPUT DISPLAY FORM FOR THE PROTOTYPES

3.6.0 Database Design

The prototypes are developed using a MYSQL database named “RP” with several tables as outlined below:-

a. Parties Table

This table contains a list of all political parties registered with the registrar of the political parties in Kenya

Before a party appears in the database, the party has to be registered first with the registrar of political parties by so doing; the secretaries to political parties will be in position to upload their members to the members table in the RP database.

Field Name	Type	Null	Default
id	int(5)	No	PRIMARY KEY
party_name	varchar(40)	No	
party_patron	varchar(40)	No	
party_secretary	varchar(40)	No	
party_logo	varchar(40)	No	
party_office	varchar(40)	No	
dcreated	timestamp	No	CURRENT_TIMESTAMP

TABLE 3.5: POLITICAL PARTY'S IMPLEMENTATION DESIGN VIEW

b. Members table

The members table holds information regarding the members registered to different political parties after being uploaded by the secretaries of respective parties.

Field Name	Type	Null	Default
id	int(11)	No	PRIMARY KEY
national_id	varchar(40)	No	
name	varchar(40)	No	
position	varchar(40)	No	
date_joined	timestamp	No	CURRENT_TIMESTAMP

TABLE 3.6: PARTY MEMBERS' IMPLEMENTATION DESIGN VIEW

3.7.0 Implementation Phase

This is where the actual coding of Web Services prototypes (REST and SOAP) is done. The wrapping of components APIs to Web Services interface is done. The generations of WSDL for SOAP and WS test client are produced. The WS will be deployed to the target application server.

3.7.1 Prototype Implementation

The prototypes are implemented purely on open source software's using REST and SOAP frameworks.

a. Server side tools

1. PHP sockets.

2. Apache Web server.

3. MYSQL database

b. Client side languages

1. HTML - HyperText Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser.

2. XML - Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

3. JSON- JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays

c. Installation and Configuration of the Application Prototype and other Installations

XAMMP

The system uses the XAMPP server and requires installation of Apache Server, MYSQL, and PHP in the computer that will serve as the web server. XAMPP is open source and freeware package that bundles Apache, MYSQL, and PHP into one executable application.
After downloading XAMPP, double click on the file then begin the installation process as follows:

i) Click Next to begin: After agreeing to the XAMPP license, select the destination location. Leave the default location as "c:/xampp" and click Next.
ii) Leave the default Start Menu shortcut as "XamppServer" and click Next.
iii) Select automatically launch XAMPP on startup. This will allow your machine to act as a server whenever it is started. Select the check box and click Next.
iv) XAMPP will summarize your selections. Click Install and XAMPP will begin the install process; XAMPP extracting and installing itself. The process should only take

a few seconds, and XAMPP will prompt you to choose a folder for your "Document Root." Leave the default folder as "www" and click Ok.

v) XAMPP will prompt you to enter the SMTP server to be used by PHP to send emails. Leave the default value as "localhost" and click Next.

vi) XAMPP will then prompt you to enter the default email address to be used by PHP to send emails. Put your email address in this field and click Next.

vii) If you have Firefox installed, XAMPP will ask you if you would like to use Firefox as the default browser with XAMPP. This is a personal preference, so feel free to choose "Yes" or "No." I will choose "Yes" and then click Next.

viii) Congratulations, the installation process is complete, click Finish and Launch XAMPP now.

3.7.2 SOAP Implementation

SOAP-based architecture revolves around the transmission of XML-encoded messages over HTTP. Specific SOAP service sets are defined in web service definition language (WSDL) files which are essentially XML files (Gavin Mulligan, 2009).

The WSDL file defines the port which is later mapped to the overarching Portal SOAP service set. The port itself is composed of multiple operations, each one representing a single service to be implemented in the set. In turn, each operation maps previously-defined SOAP messages as its input and output types.

The content of this WSDL file represents a language- and platform-neutral method of remotely communicating the SOAP service interface. Once the WSDL file for the SOAP implementation of the service-oriented architecture is written, it needs to be made publicly available to all Portal clients capable of reaching a particular Portal server. This is done through the use of a SOAP-enabled HTTP web server; and may be enabled in a variety of different ways.

Regardless of how the HTTP server to be used is configured, server-side code needs to be written that is responsible for handling incoming service requests and formulating appropriate responses.

3.7.3 REST Implementation

While SOAP adheres very closely to the RPC model, REST revolves around the concept of resources and focuses on using the inherent power of HTTP to retrieve representations of these resources in varying states. In REST style, every resource is signified by a unique URL which may be operated on by a subset of the core set of HTTP commands: Get, Post, Put, and Delete.

Compared to SOAP, instead of a specialized SOAP client, any HTTP client library may be used to interact with the Portal REST server. Every contemporary language comes equipped with a built-in HTTP library as it is a universal protocol for communicating over the Internet (Gavin Mulligan, 2009).

3.8.0 Experiments Phase

This being a comparative study is where complete test / experiments on both prototypes for the Web Services including functional and non-functional requirements are done. The experiment will be done to answer the research questions and also see if the research objectives are met.

3. 8.1 REST and SOAP Experiments

After the two prototypes were completed, experiments were done to determine how the two compares:-

a. The two are tested on their usage of consolidated data from different sources.
b. How the two compares in terms performance and scalability

As previously mentioned, the main objective of this project is to investigate how REST and SOAP compares and to examine the performance, scalability, maintainability and extensibility of the Web Services in application development.

The experimental procedure I followed evaluates the best web service to be used. The following is description of the experiments taken:-

i) Performance measure

The two prototypes are used to carry out the experiments needed to check on the performance of the Web Services during data access from the database server.

The two prototypes are used to access the same data for parties and members data from the registrar of political parties database and the results are accessed and viewed from a web application

The evaluation for both services is carried out using two different scenarios. In the first set of experiments the level of internal resource consumption is examined including memory resources. In the second set of experiments the overall performance is evaluated by measuring total elapsed response time for execution of each request. After that, evaluation for the two web service is done to determine the best performer. (Feda AlShahwan, 2010)

ii) Scalability measure

The two prototypes REST and SOAP are experimented on how well they grow or expand with the increased users; this was done using apache-jmeter which tests throughput with the increase in the number of users accessing the same data.

For the two prototypes you create a test plan for threads (users), then set the thread properties, finally you add and set the requests by specifying the server name (IP of the server), the path of the HTTP request, where the listeners will get their results from.

The process is repeated as many times as possible depending on the number of users (samples) you intend to run.

3. 9.0 Deployment Phase

From the experiment done in the previous phase above now the registrar of political parties will make a choice on which web service to use in developing a system or modify one of the prototypes to a complete system that can solve its problems.
The selection will be base on research findings; performance in terms of time and memory required to access the same amount of data, scalability, extensibility and finally the maintainability of the prototypes.

The figure below shows Web Service development lifecycle

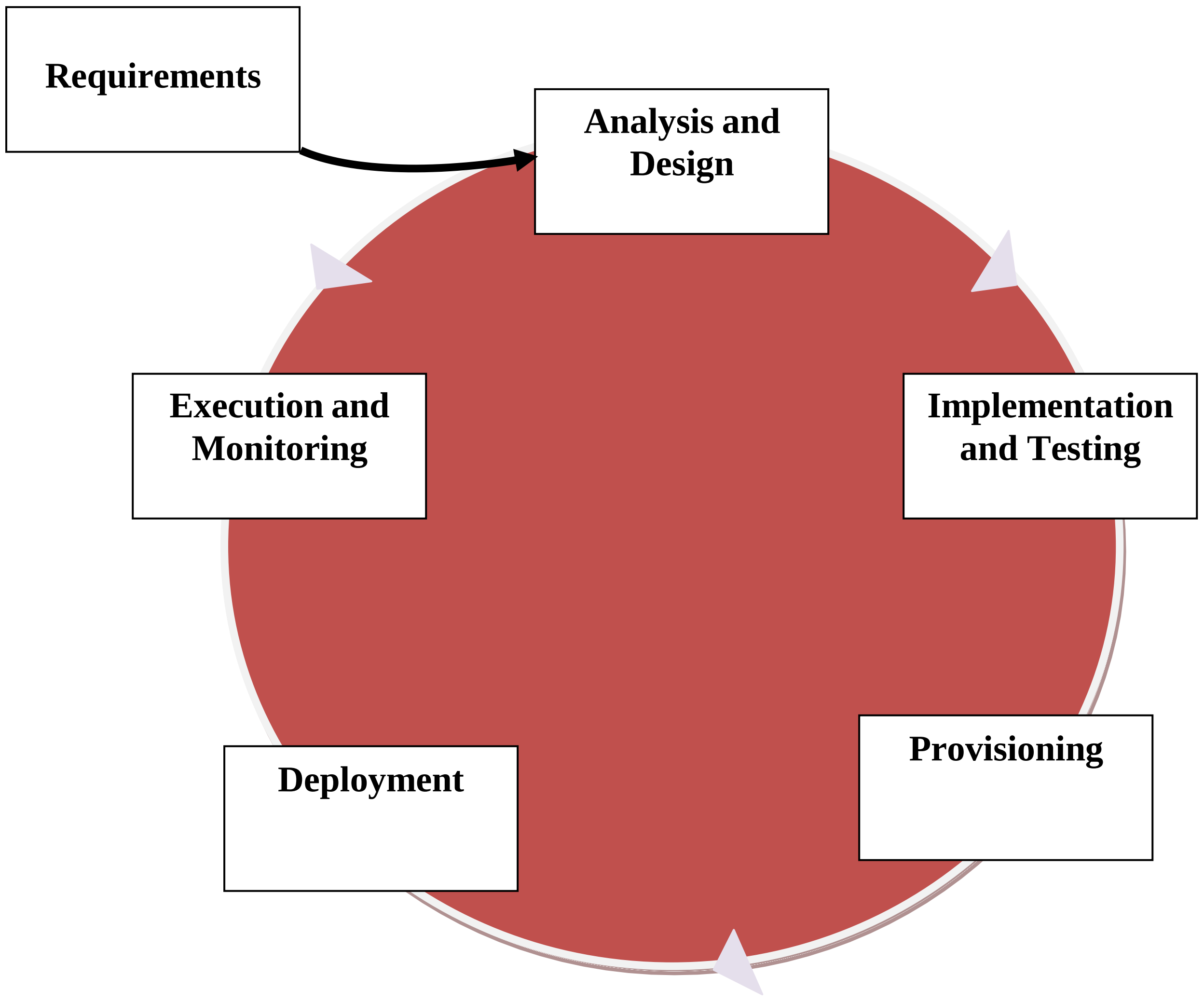


FIGURE 3.6: WEB SERVICE DEVELOPMENT LIFECYCLE

Chapter 4: Results and Discussion

4.0.0 Introduction

This chapter addresses the research findings, performance and scalability, maintainability and discussion on the results obtained from the SOAP and REST frameworks. It has graphical output inform of screen shots and gives analysis for each framework based on the research questions.

4.1.0 Findings and Results

The two frameworks SOAP and REST are able to work with consolidated data in a common MYSQL database. The two are able to fetch data from the MYSQL tables and presenting it through a common browser (internet explorer, Mozilla, Chrome, Opera etc.)

The REST framework is able to fetch the data directly from the database and present it through a browser unlike the SOAP which the developer has to develop the soap server for fetching the data from the MYSQL database and a soap client which now presents the data from the soap server to the browser through WSDL.

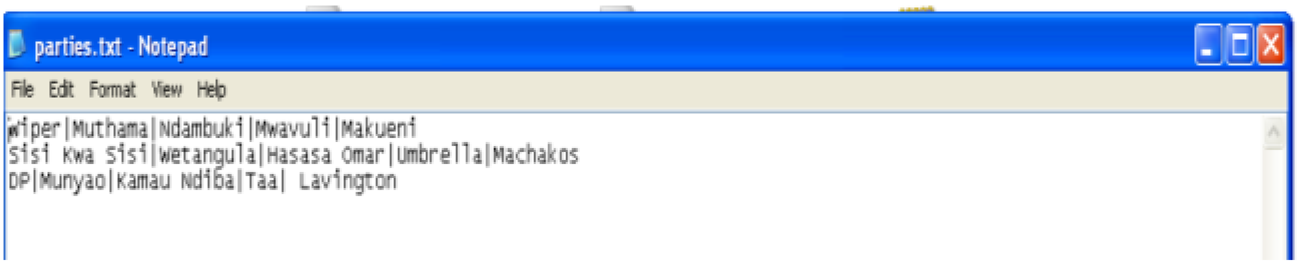
REST has the potential to present the data from the database in the formats of HTML, JSON and XML while soap is only able to present its data in the format of JSON.

4.1.1 Data consolidation (CSV/XLS and TXT file)

The two frameworks REST and SOAP are able support the uploading data to the MYSQL database using either CSV or TXT without any problem.

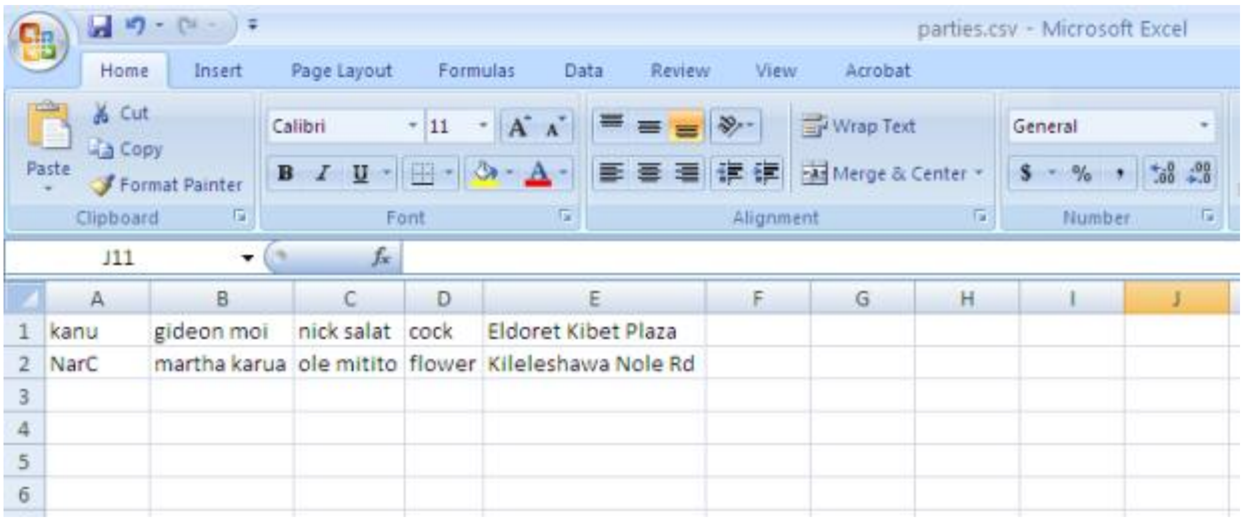
The data to be upload has to be pre-processed so that it meets a standard format with all the columns / attributes as the exactly appear in the database tables.

The screen shots below shows the sample CSV and TXT screen shots to be used for the purpose of uploading the data.



```
wiper|Muthama|Ndambuki|Mwavuli|Makueni
Sisi Kwa Sisi|Wetangula|Hasasa Omar|Umbrella|Machakos
DP|Munyao|Kamau Ndiba|Taa| Lavington
```

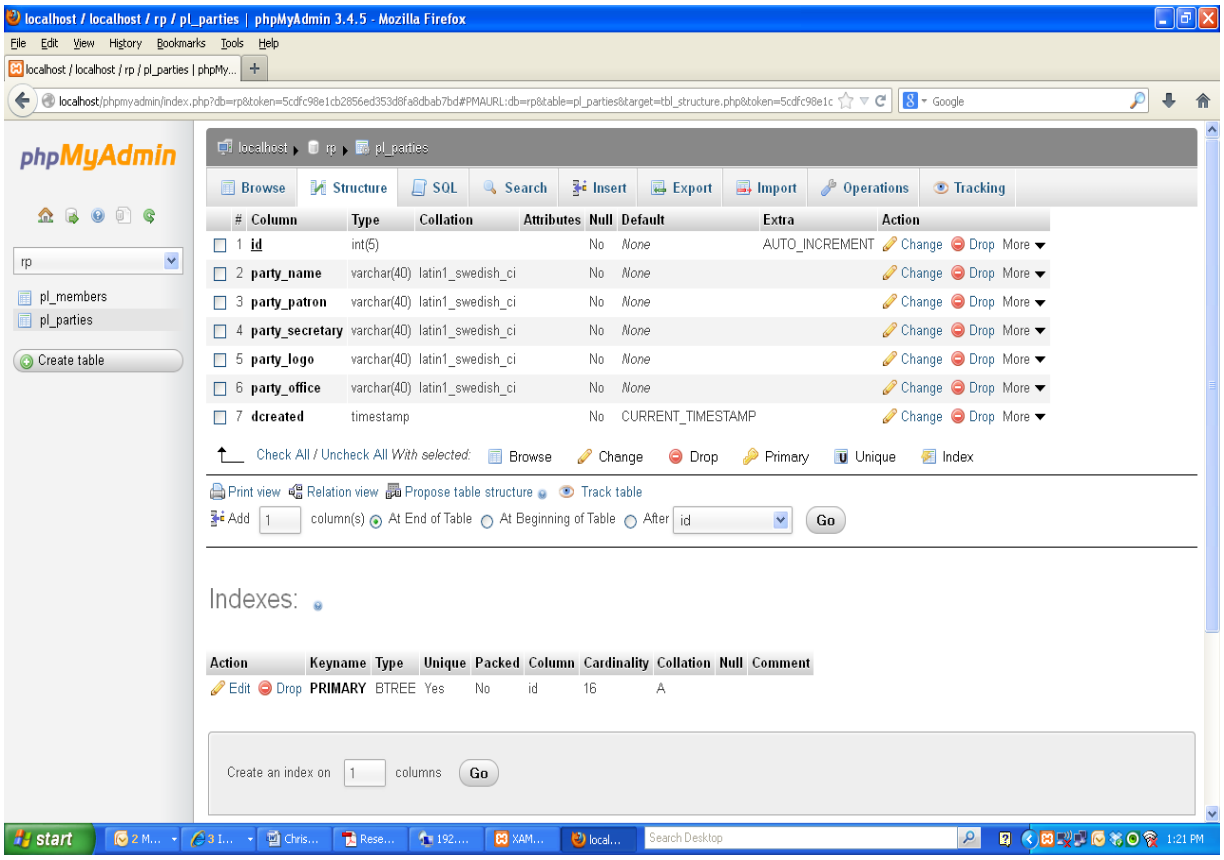
FIGURE 4.1: TEXT FILE


parties.csv - Microsoft Excel
Home Insert Page Layout Formulas Data Review View Acrobat
Cut Copy Format Painter Paste Clipboard
Calibri 11 Font
Wrap Text Merge & Center Alignment
General Number
J11

	A	B	C	D	E	F	G	H	I	J
1	kanu	gideon moi	nick salat	cock	Eldoret Kibet Plaza					
2	NarC	martha karua	ole mitito	flower	Kileleshawa Nole Rd					
3										
4										
5										
6										

FIGURE 4.2: CSV / XLS FILE

Once the data is uploaded in the databases using the formats specified above, it will appear in the MYSQL as shown in the screen shot below:-


localhost / localhost / rp / pl_parties | phpMyAdmin 3.4.5 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost/phpmyadmin/index.php?db=rp&token=5cdfc98e1cb2856ed353d8fa8dbab7bd#PMAURL:db=rp&table=pl_parties&target=tbl_structure.php&token=5cdfc98e1c
phpMyAdmin
rp
pl_members
pl_parties
Create table
localhost ▸ rp ▸ pl_parties
Browse Structure SQL Search Insert Export Import Operations Tracking
Check All / Uncheck All With selected: Browse Change Drop Primary Unique Index
Print view Relation view Propose table structure Track table
Add 1 column(s) At End of Table At Beginning of Table After id Go
Indexes:
Create an index on 1 columns Go
start
1:21 PM

	#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	id	int(5)			No	None	AUTO_INCREMENT	Change Drop More
☐	2	party_name	varchar(40)	latin1_swedish_ci		No	None		Change Drop More
☐	3	party_patron	varchar(40)	latin1_swedish_ci		No	None		Change Drop More
☐	4	party_secretary	varchar(40)	latin1_swedish_ci		No	None		Change Drop More
☐	5	party_logo	varchar(40)	latin1_swedish_ci		No	None		Change Drop More
☐	6	party_office	varchar(40)	latin1_swedish_ci		No	None		Change Drop More
☐	7	dcreated	timestamp			No	CURRENT_TIMESTAMP		Change Drop More

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	16	A		

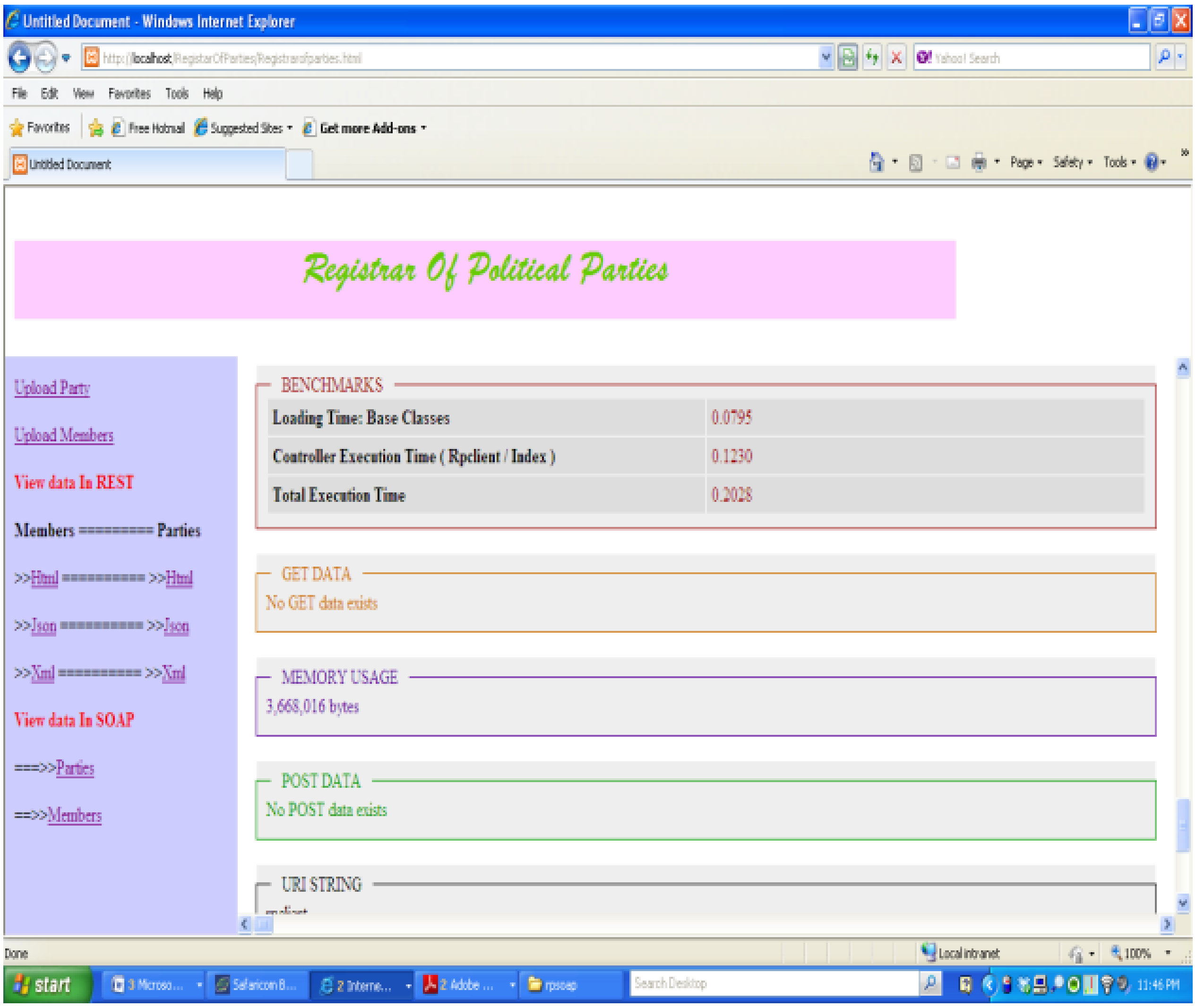
FIGURE 4.3: PHP MYADMIN SCREEN FOR MYSQL

4.1.2 Performance measure in REST and SOAP

When it comes to performance REST takes less time and computer resources e.g. memory to fetch and load the same amount of data from the database as compared to SOAP which consumes a lot of memory and time to load.

REST has a bit better performance because it bears minimal overhead on top of HTTP compared to SOAP which brings with it a stack of different (generated) handlers and parsers. SOAP requires a lot of linkage with the soap server, WSDL, nusoap and soap client to be able to load data from the MYSQL server to the browser.

The screen shots below shows the performance and memory benchmarks for SOAP framework.


Untitled Document - Windows Internet Explorer
http://localhost/RegistarOfParties/Registrarofparties.html
File Edit View Favorites Tools Help
Favorites
Free Hotmail
Suggested Sites
Get more Add-ons
Untitled Document
Page Safety Tools
Done
Local intranet
100%
start
Search Desktop
11:46 PM

Registrar Of Political Parties

Upload Party

Upload Members

View data In REST

Members ========= Parties

>>Html ========== >>Html

>>Json ========== >>Json

>>Xml ========== >>Xml

View data In SOAP

===>>Parties

==>>Members

BENCHMARKS

Loading Time: Base Classes	0.0795
Controller Execution Time (Rpclient / Index)	0.1230
Total Execution Time	0.2028

GET DATA

No GET data exists

MEMORY USAGE

3,668,016 bytes

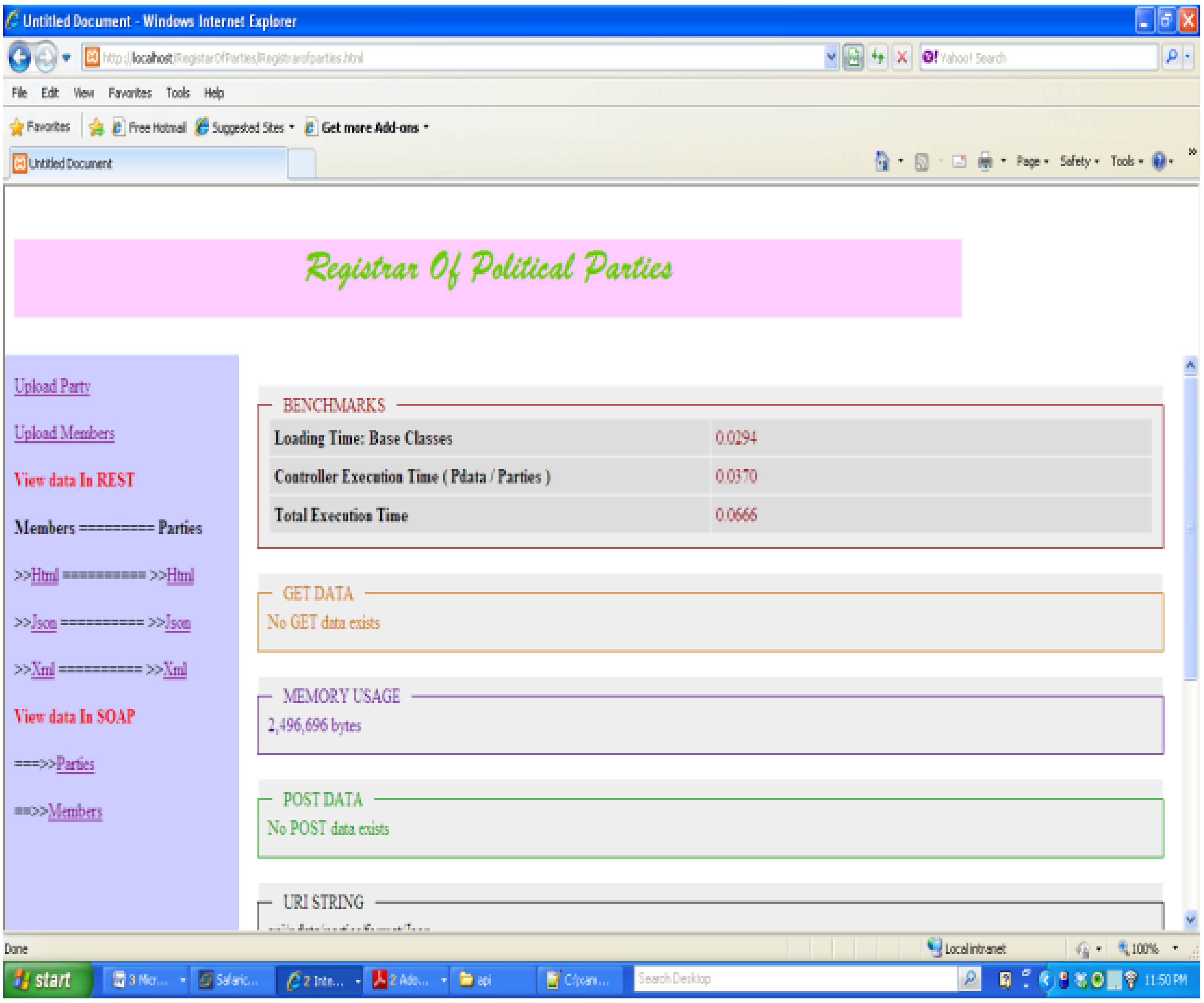
POST DATA

No POST data exists

URI STRING

FIGURE 4.4: SCREEN SHOT FOR PERFORMANCE MEASURE IN SOAP

The screen shot below shows a demonstration of how REST framework fetches and loads same amount of information from the database within a short time and memory usage compared to SOAP



Registrar Of Political Parties

Upload Party

Upload Members

View data In REST

Members ========= Parties

>>Html ========== >>Html

>>Json ========== >>Json

>>Xml ========== >>Xml

View data In SOAP

===>>Parties

==>>Members

BENCHMARKS

Loading Time: Base Classes	0.0294
Controller Execution Time (Pdata / Parties)	0.0370
Total Execution Time	0.0666

GET DATA

No GET data exists

MEMORY USAGE

2,496,696 bytes

POST DATA

No POST data exists

URI STRING

FIGURE 4.5: SCREEN SHOT FOR PERFORMANCE MEASURE IN REST

4.1.3 Graphical Analysis of REST and SOAP on Performance

The data below forms a sample data of the analysis that I did on performance of both REST and SOAP

Transaction ID	Data Records	REST Memory	REST Time	SOAP Memory	SOAP Time
1	10	2.4	0.06	3.504	0.1842
2	40	3.0	0.8	4.38	2.456
3	70	4.0	0.1	5.84	0.307
4	110	4.4	0.26	6.424	0.7982
5	140	5.2	0.3	7.592	0.921

TABLE 4.1: REST AND SOAP DATA ON PERFORMANCE

Memory Usage Graph

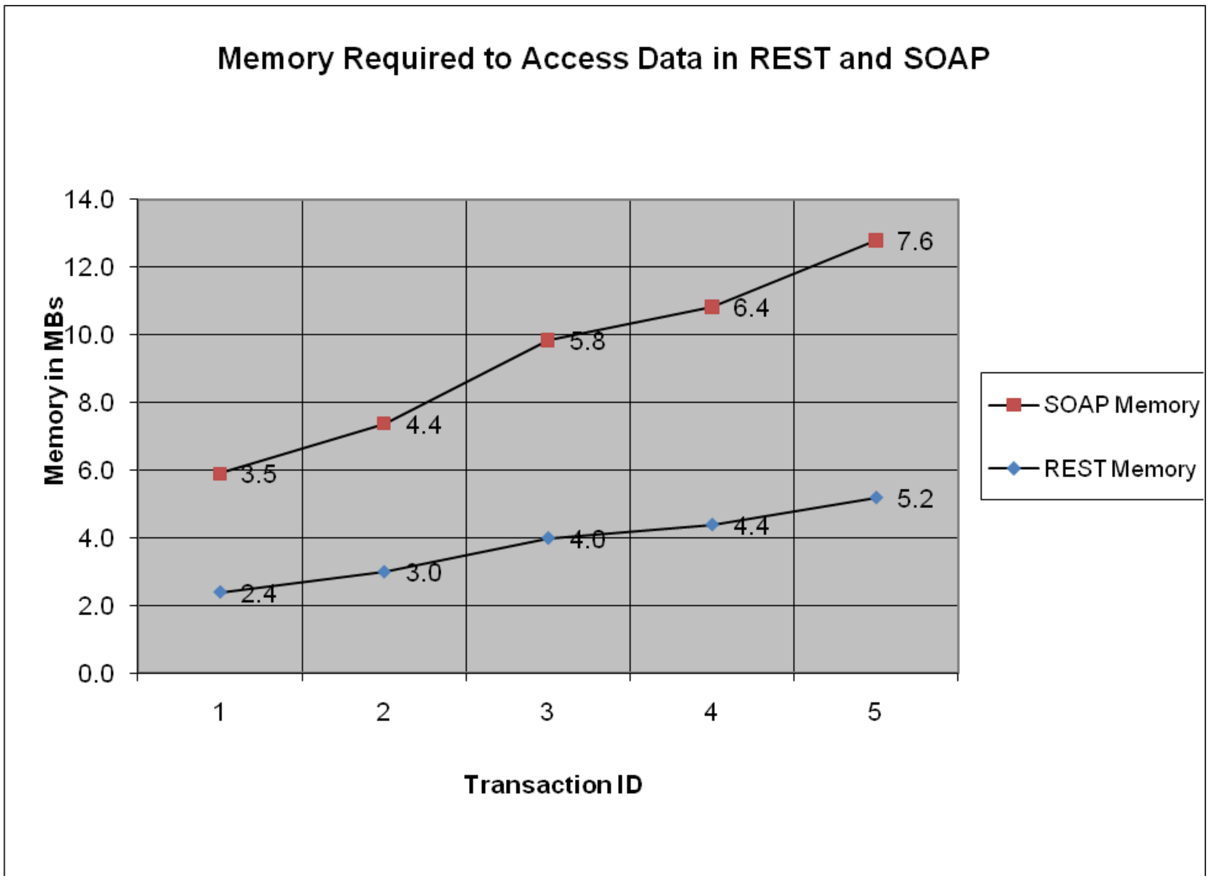


FIGURE 4.6: MEMORY REQUIRED FOR DATA ACCESS IN REST AND SOAP

Time Usage Graph

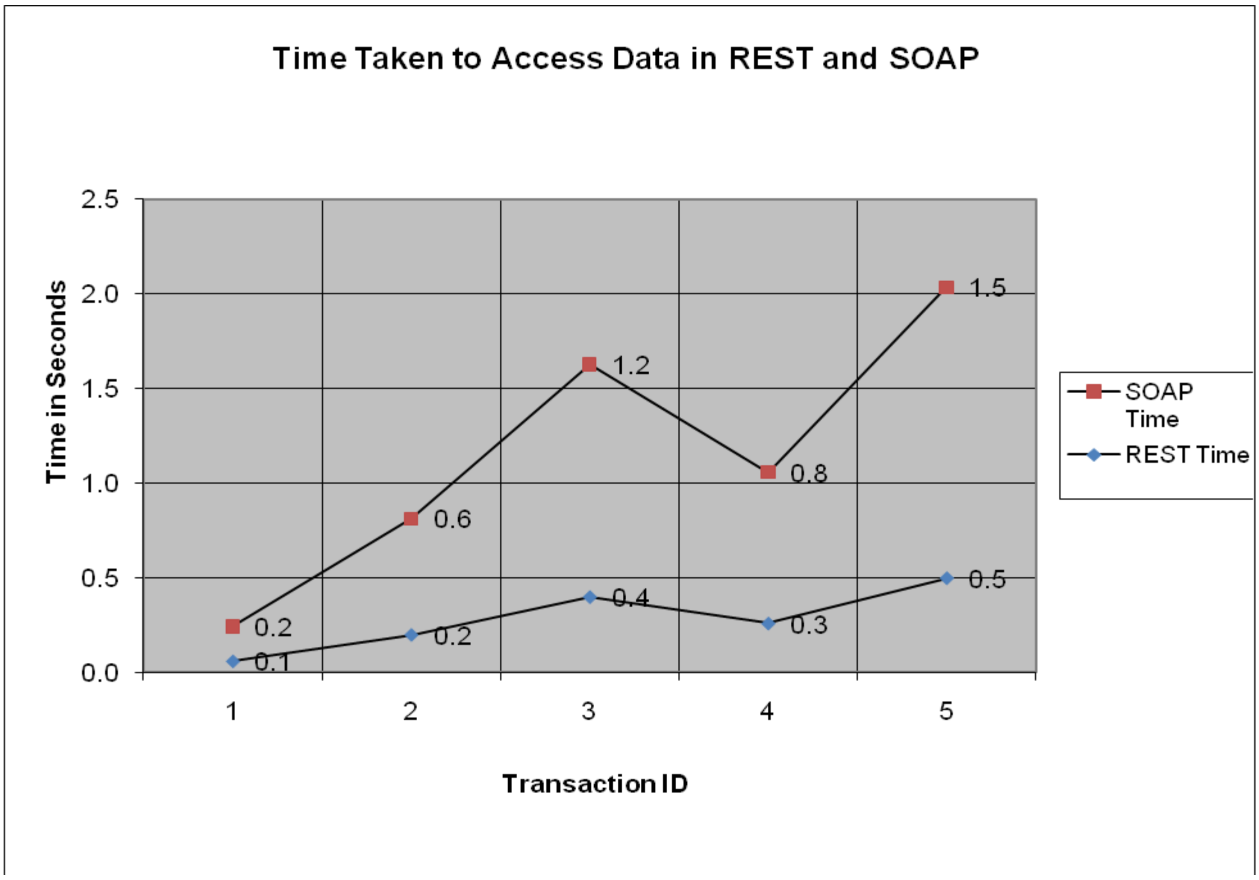


FIGURE 4.7: TIME REQUIRED FOR DATA ACCESS IN REST AND SOAP

From the results above, REST has proved to have the best performance compared to SOAP in terms of Time and Memory required to access to access the same amount of information. This is because REST is lighter than SOAP. SOAP requires an XML wrapper around every request and response while in REST not a lot of extra xml markup is needed. SOAP response could require more than 10 times as many bytes as would the same response in REST (Rozlog, 2010).

4.1.4 Graphical Analysis of REST and SOAP on Scalability

Scalability being the ability of the prototype architecture to grow to accommodate increasing number of users, applications and system; this will be measured based on the throughput in bits per second as the number of users or threads increase.

In this research scalability is measured using throughput depending on the number of users connecting to the server. Throughput or network throughput is the average rate of successful message delivery over a communication channel. The data is delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot.

The throughput can be analyzed mathematically by means of queuing theory, where the load in packets per time unit is denoted arrival rate, and the throughput in packets per time unit is denoted departure rate.

The screen shot below shows the process of capturing throughput for both SOAP and REST using apache-jmeter which is simulation software that measures scalability.

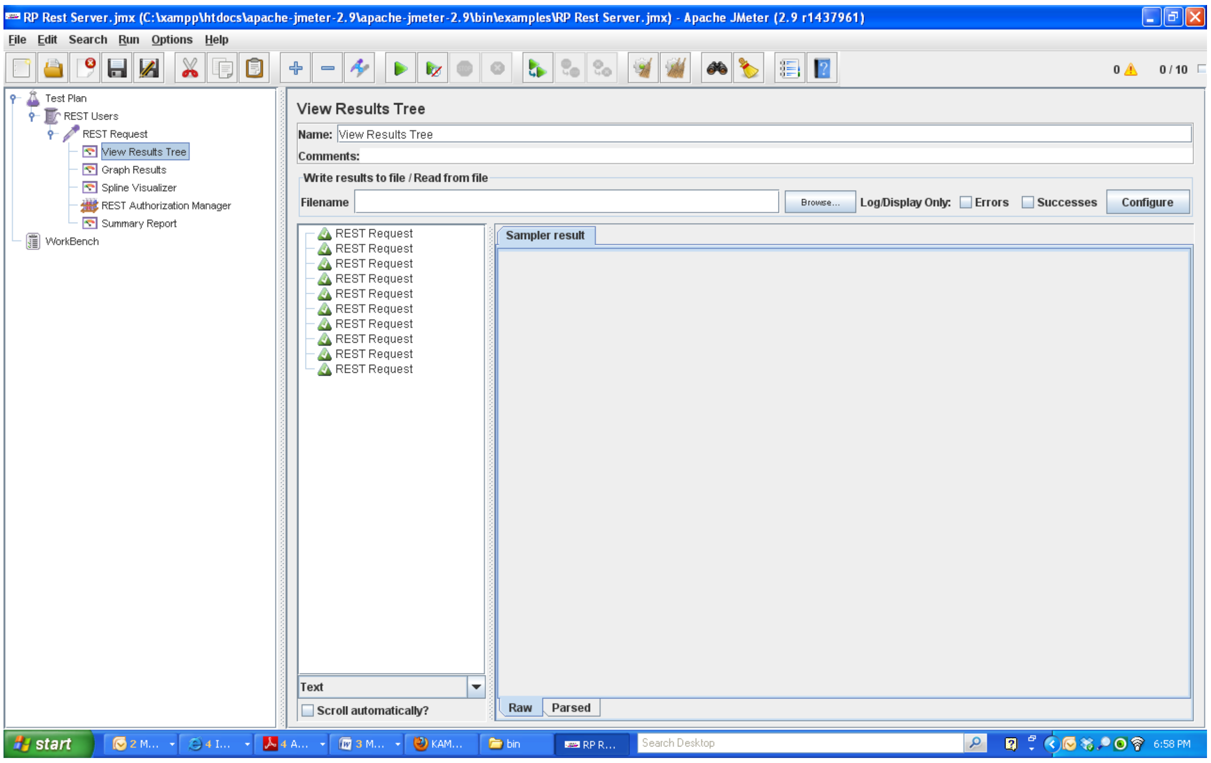
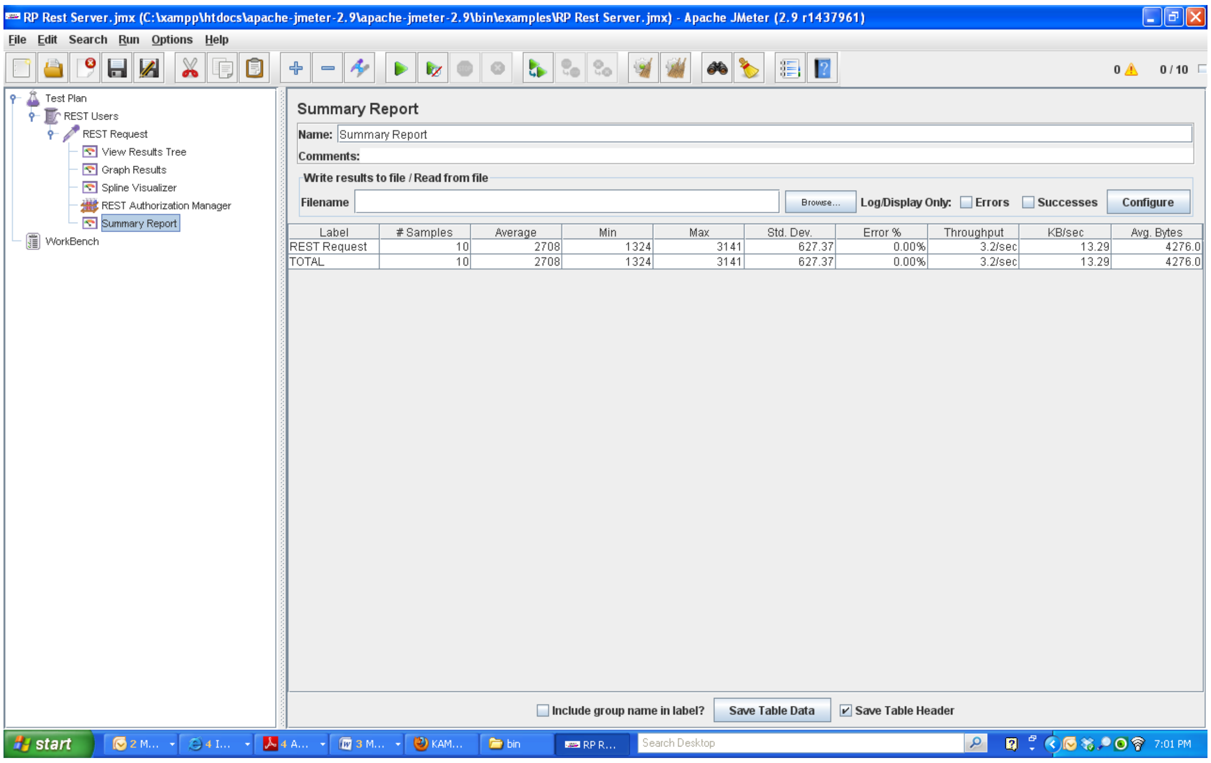


FIGURE 4.8: REST REQUESTS TO MEASURE SCALABILITY BASED ON NUMBER OF USERS


RP Rest Server.jmx (C:\xampp\htdocs\apache-jmeter-2.9\apache-jmeter-2.9\bin\examples\RP Rest Server.jmx) - Apache JMeter (2.9 r1437961)

Test Plan
REST Users
REST Request
View Results Tree
Graph Results
Spline Visualizer
REST Authorization Manager
Summary Report
WorkBench

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

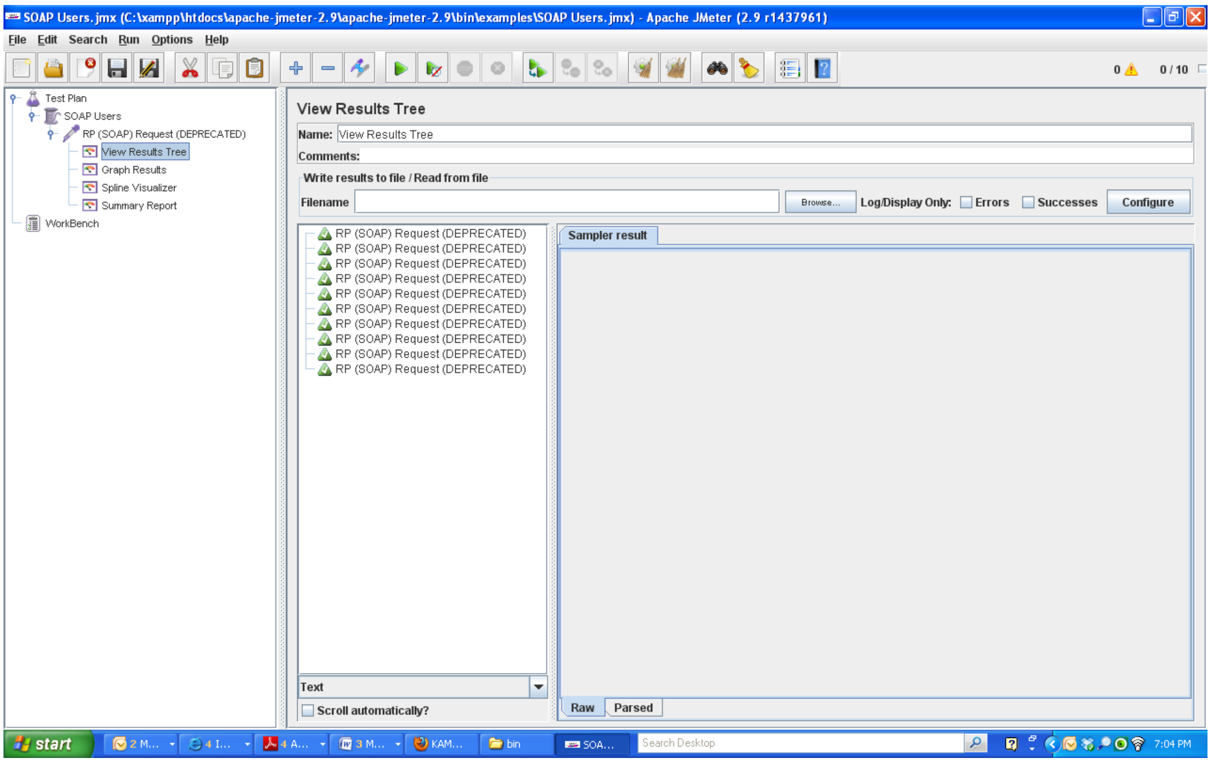
Filename

Log/Display Only: Errors Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
REST Request	10	2708	1324	3141	627.37	0.00%	3.2/sec	13.29	4276.0
TOTAL	10	2708	1324	3141	627.37	0.00%	3.2/sec	13.29	4276.0

Include group name in label? Save Table Data ✓ Save Table Header

FIGURE 4.9: REST THROUGHPUT MEASURING SCALABILITY BASED ON NUMBER OF USERS


SOAP Users.jmx (C:\xampp\htdocs\apache-jmeter-2.9\apache-jmeter-2.9\bin\examples\SOAP Users.jmx) - Apache JMeter (2.9 r1437961)

Test Plan
SOAP Users
RP (SOAP) Request (DEPRECATED)
View Results Tree
Graph Results
Spline Visualizer
Summary Report
WorkBench

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename

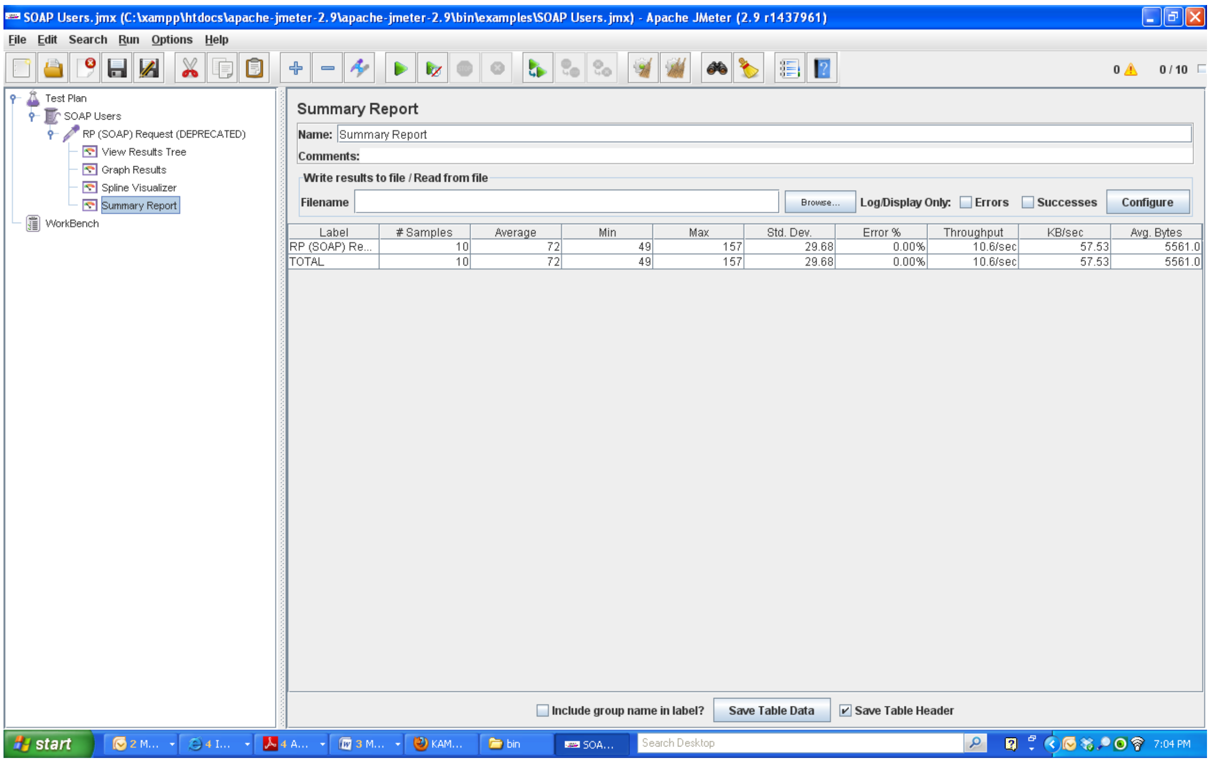
Log/Display Only: Errors Successes

RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)
RP (SOAP) Request (DEPRECATED)

Text

Scroll automatically?

FIGURE 4.10: SOAP REQUESTS TO MEASURE SCALABILITY BASED ON NUMBER OF USERS


SOAP Users.jmx (C:\xampp\htdocs\apache-jmeter-2.9\apache-jmeter-2.9\bin\examples\SOAP Users.jmx) - Apache JMeter (2.9 r1437961)

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
RP (SOAP) Re...	10	72	49	157	29.68	0.00%	10.6/sec	57.53	5561.0
TOTAL	10	72	49	157	29.68	0.00%	10.6/sec	57.53	5561.0

☐ Include group name in label? ☑ Save Table Header

FIGURE 4.11: SOAP THROUGHPUT MEASURING SCALABILITY BASED ON NUMBER OF USERS

Scalability being one of the main differences between REST and SOAP, SOAP services are much harder to scale than RESTful services, which is, of course, one of the reasons that REST is often chosen as the architecture for services that are exposed via the Internet (like Facebook, MySpace, Twitter, and so on) (Jon, 2007).

The data below forms a sample data of the analysis that I did on scalability for both REST and SOAP

USER SAMPLES										
Sample (Users)	10	20	30	40	50	60	70	80	90	100
REST Throughput	5.1	22.4	29	27	27	24	24	23	23.2	46
SOAP throughput	10	18.1	14	17	19	21	23	24.9	26.2	27

TABLE 4.2: REST AND SOAP DATA THROUGHPUT BASED ON USERS

Scalability of SOAP and REST

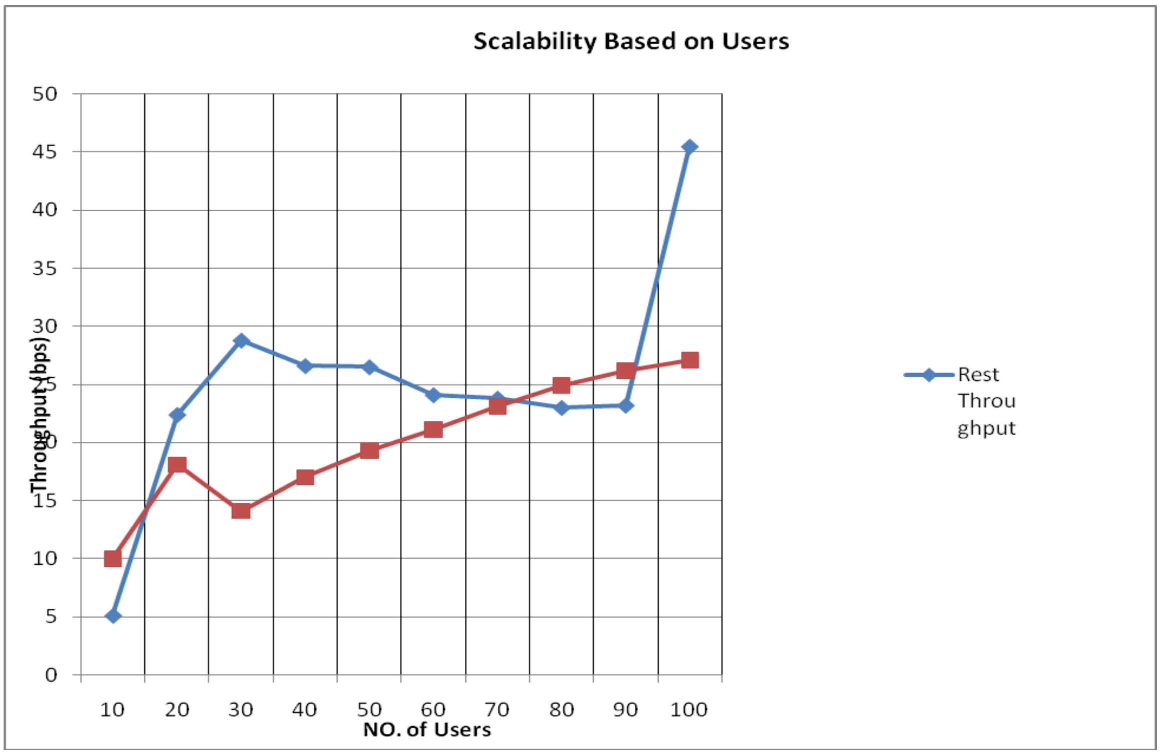


FIGURE 4.12: SCALABILITY OF REST AND SOAP MEASURED USING THROUGHPUT

Form the graph above its evident that REST scales very well with the increase in the number of users as its information can be cached by intermediate proxy servers as its request uses GET and also because it's totally stateless operations compared to SOAP. (Rozlog, 2010)

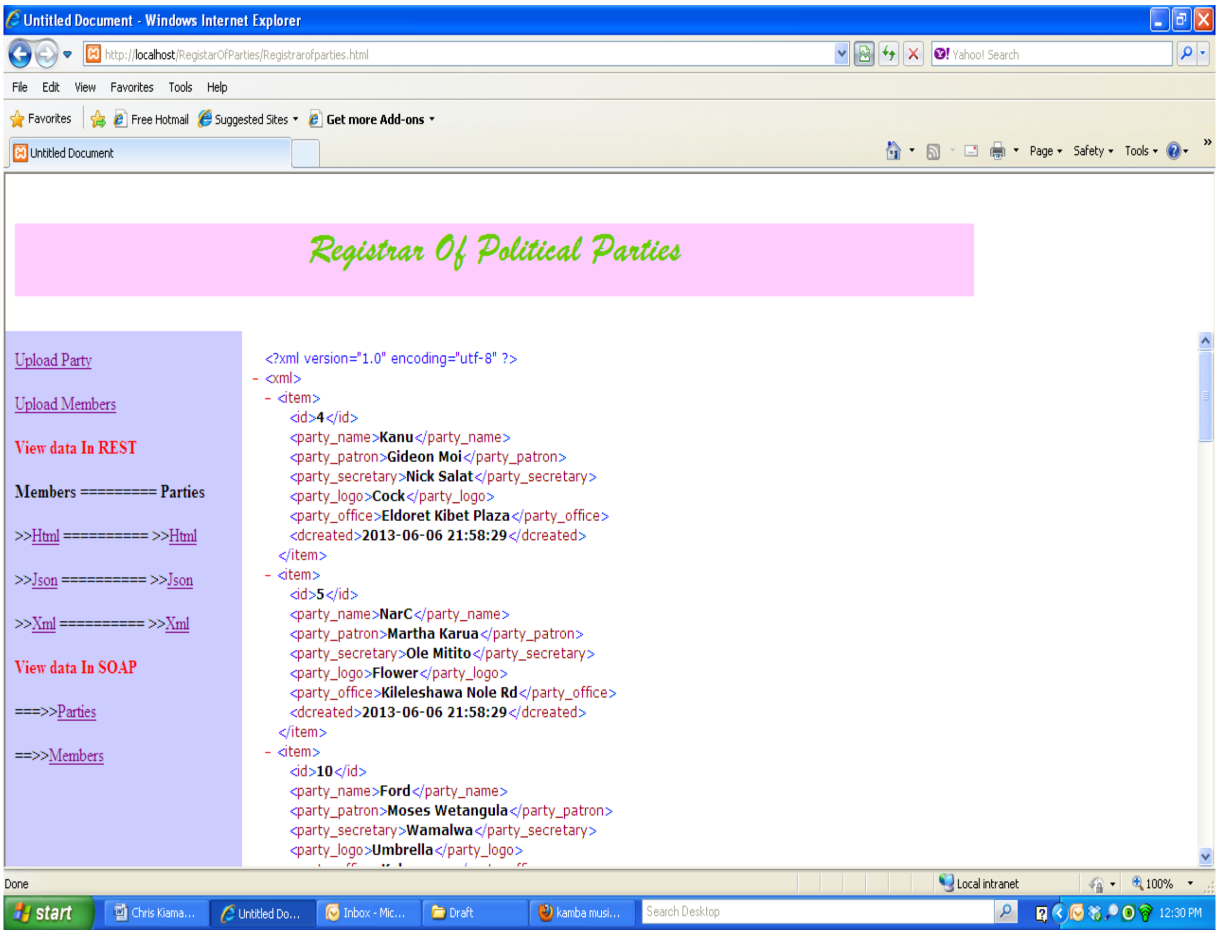
4.1.5 Extensibility of REST and SOAP

Extensibility being the ability of the prototype system to allow and accept significant extension of its capabilities, functionality, enhancement for the purpose of meeting future needs and significantly changing requirements without major rewriting of code or changes in its basic architecture.

Comparatively REST extents very well compared to SOAP and is able to have several options to display data from the server e.g. JSON, HTML, XML.

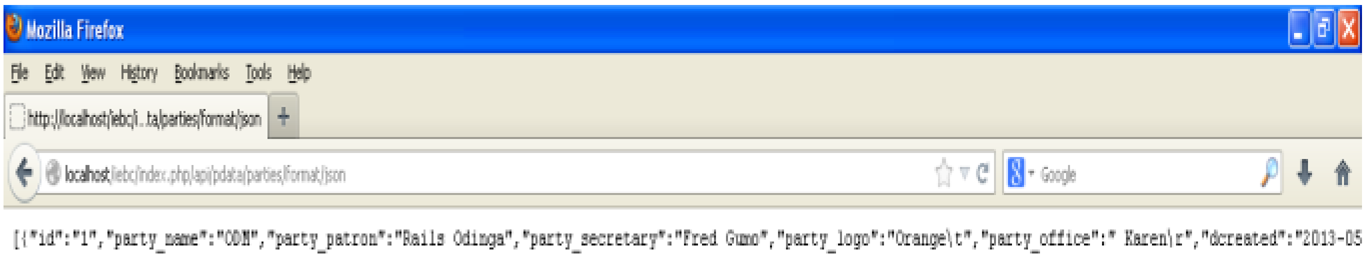
Again REST has better extensibility as it able to present the same data in different formats e.g. XML, HTML and JSON compared to SOAP which can only present data in format of JSON.

The screen shots below show the various ways in which the two are able to present the same data form a common database.


Untitled Document - Windows Internet Explorer
http://localhost/RegistarOfParties/Registrarofparties.html
Registrar Of Political Parties
Upload Party
Upload Members
View data In REST
Members ========= Parties
>>Html ========== >>Html
>>Json ========== >>Json
>>Xml ========== >>Xml
View data In SOAP
===>>Parties
==>>Members

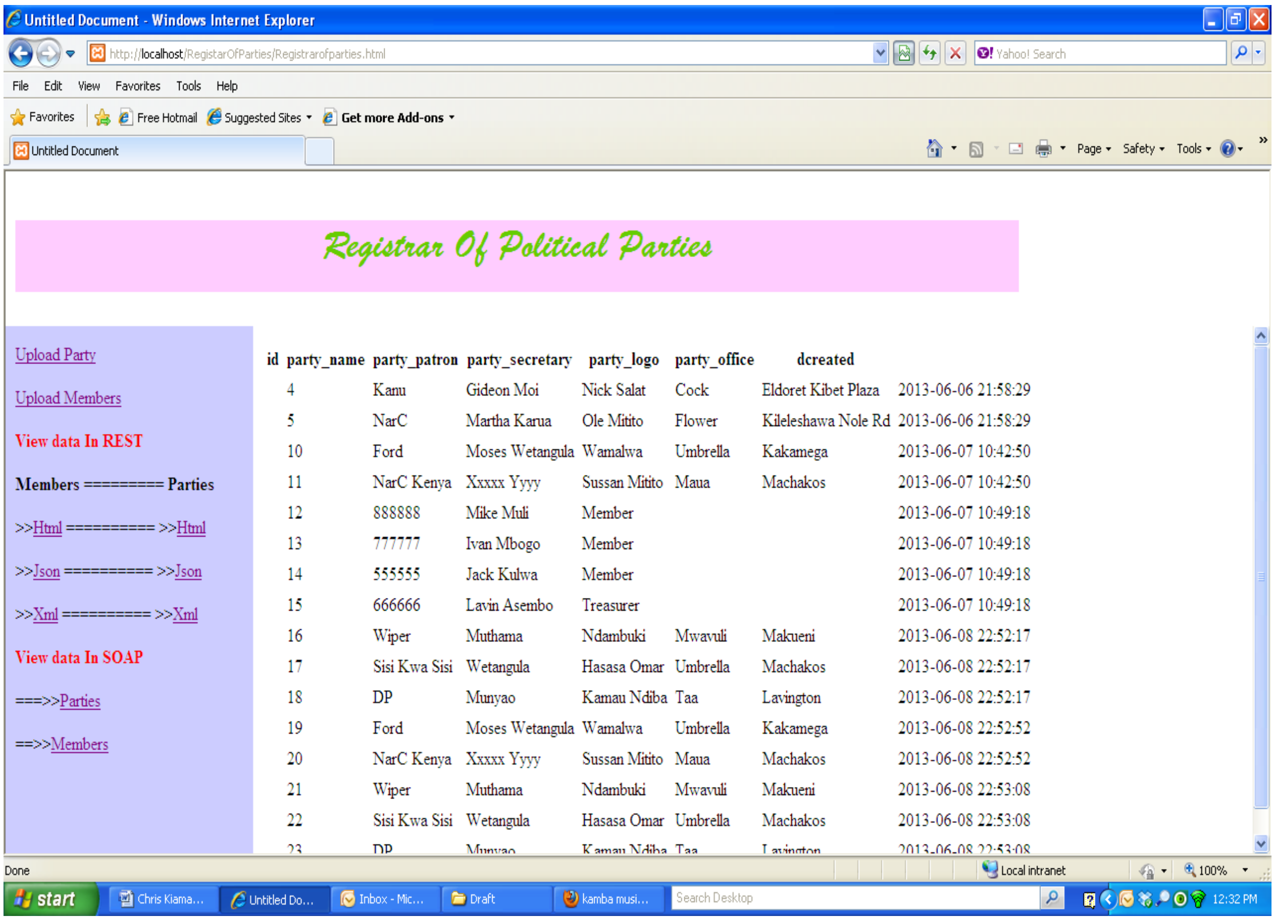
```
  <?xml version="1.0" encoding="utf-8" ?>
- <xml>
  - <item>
      <id>4</id>
      <party_name>Kanu</party_name>
      <party_patron>Gideon Moi</party_patron>
      <party_secretary>Nick Salat</party_secretary>
      <party_logo>Cock</party_logo>
      <party_office>Eldoret Kibet Plaza</party_office>
      <dcreated>2013-06-06 21:58:29</dcreated>
    </item>
  - <item>
      <id>5</id>
      <party_name>NarC</party_name>
      <party_patron>Martha Karua</party_patron>
      <party_secretary>Ole Mitito</party_secretary>
      <party_logo>Flower</party_logo>
      <party_office>Kileleshawa Nole Rd</party_office>
      <dcreated>2013-06-06 21:58:29</dcreated>
    </item>
  - <item>
      <id>10</id>
      <party_name>Ford</party_name>
      <party_patron>Moses Wetangula</party_patron>
      <party_secretary>Wamalwa</party_secretary>
      <party_logo>Umbrella</party_logo>
```

FIGURE 4.13: XML DATA USING REST


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/iebc/i...ta/parties/format/json
localhost/iebc/index.php/api/pdata/parties/format/json
Google

[{"id":"1","party_name":"ODM","party_patron":"Rails Odinga","party_secretary":"Fred Gumo","party_logo":"Orange\t","party_office":" Karen\r","dcreated":"2013-05

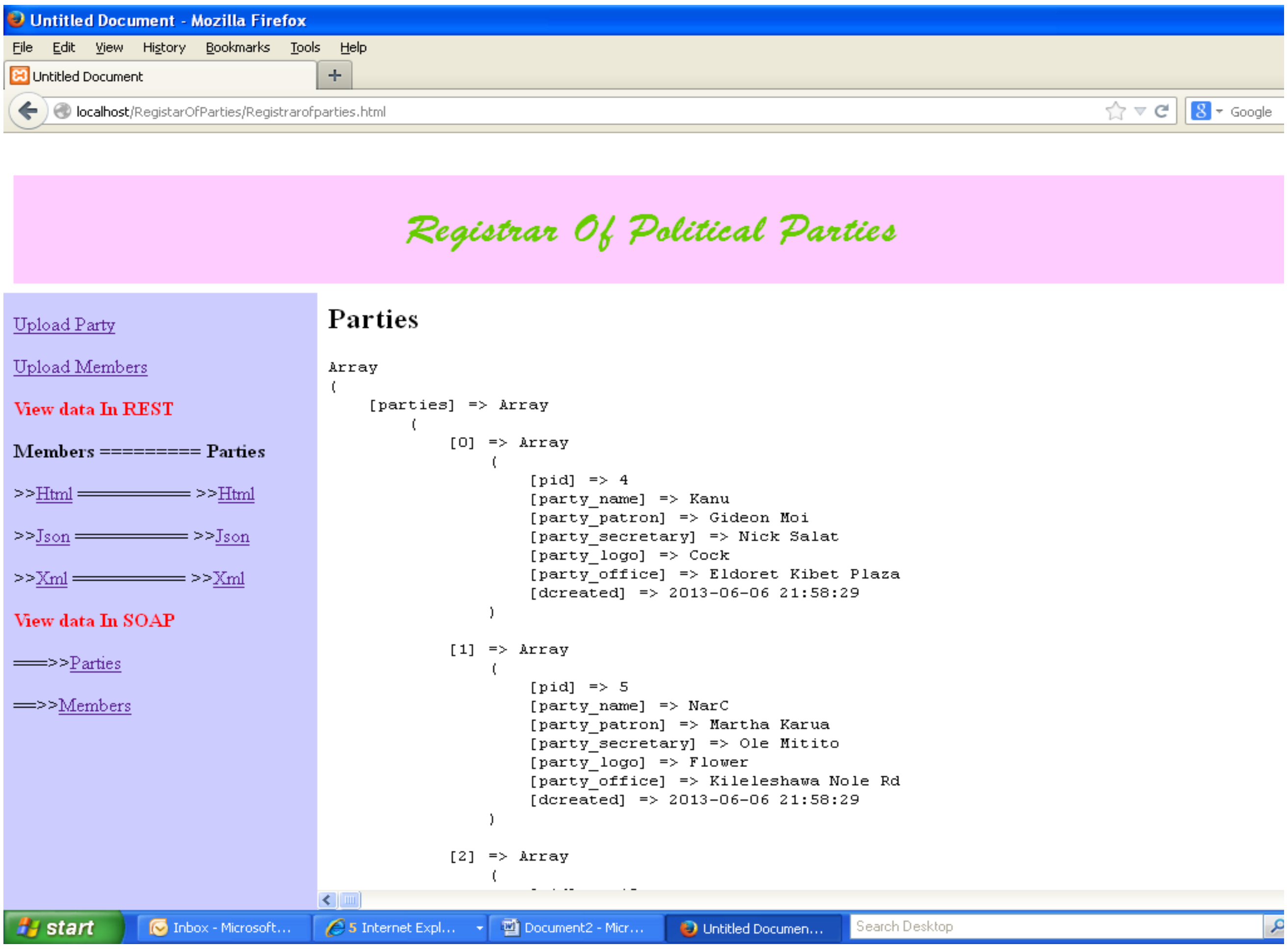
FIGURE 4.14: JSON DATA USING REST


Untitled Document - Windows Internet Explorer
http://localhost/RegistarOfParties/Registrarofparties.html
File Edit View Favorites Tools Help
Registrar Of Political Parties
Upload Party
Upload Members
View data In REST
Members ========= Parties
>>Html ========== >>Html
>>Json ========== >>Json
>>Xml ========== >>Xml
View data In SOAP
===>>Parties
==>>Members
Done
Local intranet

id	party_name	party_patron	party_secretary	party_logo	party_office	dcreated
4	Kanu	Gideon Moi	Nick Salat	Cock	Eldoret Kibet Plaza	2013-06-06 21:58:29
5	NarC	Martha Karua	Ole Mitito	Flower	Kileleshawa Nole Rd	2013-06-06 21:58:29
10	Ford	Moses Wetangula	Wamalwa	Umbrella	Kakamega	2013-06-07 10:42:50
11	NarC Kenya	Xxxxx Yyyy	Sussan Mitito	Maua	Machakos	2013-06-07 10:42:50
12	888888	Mike Muli	Member			2013-06-07 10:49:18
13	777777	Ivan Mbogo	Member			2013-06-07 10:49:18
14	555555	Jack Kulwa	Member			2013-06-07 10:49:18
15	666666	Lavin Asembo	Treasurer			2013-06-07 10:49:18
16	Wiper	Muthama	Ndambuki	Mwavuli	Makueni	2013-06-08 22:52:17
17	Sisi Kwa Sisi	Wetangula	Hasasa Omar	Umbrella	Machakos	2013-06-08 22:52:17
18	DP	Munyao	Kamau Ndiba	Taa	Lavington	2013-06-08 22:52:17
19	Ford	Moses Wetangula	Wamalwa	Umbrella	Kakamega	2013-06-08 22:52:52
20	NarC Kenya	Xxxxx Yyyy	Sussan Mitito	Maua	Machakos	2013-06-08 22:52:52
21	Wiper	Muthama	Ndambuki	Mwavuli	Makueni	2013-06-08 22:53:08
22	Sisi Kwa Sisi	Wetangula	Hasasa Omar	Umbrella	Machakos	2013-06-08 22:53:08
23	DP	Munyao	Kamau Ndiba	Taa	Lavington	2013-06-08 22:53:08

FIGURE 4.15: HTML DATA USING REST

Using SOAP the political parties data can only be viewed using JSON and data is presented in the form of arrays.



Parties

```
Array
(
    [parties] => Array
        (
            [0] => Array
                (
                    [pid] => 4
                    [party_name] => Kanu
                    [party_patron] => Gideon Moi
                    [party_secretary] => Nick Salat
                    [party_logo] => Cock
                    [party_office] => Eldoret Kibet Plaza
                    [dcreated] => 2013-06-06 21:58:29
                )

            [1] => Array
                (
                    [pid] => 5
                    [party_name] => NarC
                    [party_patron] => Martha Karua
                    [party_secretary] => Ole Mitito
                    [party_logo] => Flower
                    [party_office] => Kileleshawa Nole Rd
                    [dcreated] => 2013-06-06 21:58:29
                )

            [2] => Array
                (
```

FIGURE 4.16: JSON DATA USING SOAP

4.1.6 Maintainability of REST and SOAP

REST is very lightweight, and relies upon the HTTP standard to do its work. It is great to get a useful web service up and running quickly without developing the WSDL as REST doesn't need such features. REST essentially requires HTTP, and is format-agnostic meaning you can use XML, JSON, and HTML.

SOAP is good though for the computers to understand the web service a WSDL is must as all the methods need to be defined in it. SOAP (using WSDL) is a heavy-weight standard that is centered on document passing. REST specifications are generally human-readable only (Kekoa, 2009).

The implementation of SOAP requires a client and server components that needs to be registered in the WSDL.

For more details on the SOAP implementation see **Appendix B**

4.2.0 Discussion

After the experiments were done on REST and SOAP for my research, it has proved that REST consumes less amount of memory and time to fetch the same amount of data / information from the same database source compared to SOAP.
In terms of scalability it has also proved that REST scales very well with addition / increase in number of users who access the data in the server at the same time.
Form the experiments done in this chapter above; it’s evident that REST scales very well with the increase in the number of users as its information can be cached by intermediate proxy servers as its request uses GET and also because it’s totally stateless operations compared to SOAP. (Rozlog, 2010)

With enterprise applications, think of speed and scalability—scalability being one of the main differences between REST and SOAP. SOAP services are much harder to scale than RESTful services, which is, of course, one of the reasons that REST is often chosen as the architecture for services that are exposed via the Internet like Facebook, MySpace, Twitter, and so on

Inside enterprises, applications also often need to scale as well. Using REST means that you can take advantage of HTTP caching and other features, like Conditional GET, that aid in scaling services. Many of these techniques can't be used with SOAP because SOAP uses POST only over HTTP (Jon, 2007)
With the load of the server, REST has a better performance because it bears minimal overhead on top of HTTP. Usually SOAP brings with it a stack of different (generated) handlers and parsers. Again RESTful service is easier to scale up since it doesn’t have any server side sessions (wuher, 2010).

Form the experiments done in this chapter above; REST has proved to have the best performance compared to SOAP in terms of Time and Memory required to access to access the same amount of information. This is because REST is lighter than SOAP. SOAP requires an XML wrapper around every request and response while in REST not a lot of extra xml markup is needed. SOAP response could require more than 10 times as many bytes as would the same response in REST (Rozlog, 2010).

Chapter 5: Conclusion and Recommendation

5.0 Conclusion

The usage of web service technologies will be of great assistance in providing sanity and a solution to the registrar of political parties' problems of registering genuine and legitimate members to various political parties. Both SOAP and REST has proved to be very efficient in data consolidation from different sources into a common database but data access from the common database REST has better performance, extensibility and scalability.

Comparatively REST has proved to be the best and easy to maintain as it is an architecture that needs very few lines of code to do any urgent changes required during maintenance or modifications to solve a specific problem. In terms of performance, REST takes shorter time to load similar amount of data as compared to SOAP and also it uses less resources e.g. Memory and processing power.

REST has also demonstrated high degree of scalability with increasing number of users accessing the system at the same time; this has been achieved by improved and better throughput.

In terms of complexity REST API accesses data directly from the database tables while SOAP requires a soap server, WSDL and a soap client to access the same data from the database table.

Use of REST framework will help the registrar of political parties manage party's membership with computing resources that have better performance and maintainability

5.1 Recommendations

The two approaches, SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) have advantages and disadvantages to interfacing to web services, but it is up to the web developer to make the decision of which approach may be best for each particular case (Rozlog, 2010).

From the research results, out of the two prototypes REST show considerable efficiency and effectiveness in managing the issues affecting the registrar of political parties e.g. political party's membership information. I therefore recommend REST as the preferred framework in the development of a system that will help the registrar of political parties in managing parties information and ensuring only legitimate members are registered to political parties once and not double registrations as the system will check on the national id being upload to the members table and ignore double registrations by rejecting them.

Most developers from the mainstream have at least, been exposed to the REST approach, which uses a standard URI (Uniform Resource Identifier) that makes a call to a web service very simple to understand and can be executed on really any client or server that has HTTP/HTTPS support. Developers that use this approach, cite the ease of development, use of the existing web infrastructure, and little learning overhead as key advantages to the style (Rozlog, 2010).

5.2 Suggestions for Further Improvement

Future research may address security and the analytical bit of the captured data, comparing REST and SOAP; including automatic alerts if the system encounters a member who is registered more than once across different political parties.

The registrar of political party's data being very sensitive and a concern in Kenya, a combination of several techniques need to be used to secure the Web service systems. The Security should encompass any of the following which may form part the overall security plan:-

i. Equipment deployment - this concerns where the web servers should be placed i.e. internally or off site. Web service for the registrar of political parties will be publicly accessed and will need to expose as little of its internal infrastructure as necessary, database machines should be behind a firewall within demilitarized zone (DMZ).
ii. Authenticating users – all the users of the registrar of political party's web service system will need to be authenticated with their identities to use the system. The identity information should be used to make sure a person have access to the Web service. The same should also be used to track the user's activities.
iii. Guarding data – the authentication mechanisms need to have Access Control Lists to guard files and SQL-based security to guard data in your database from un-trusted public members.
iv. Tracking user activity – as the public users are using those resources, you will want to be able to see what was done on the web service.

References

1. Chappell, D. (2009). *SOAP vs. REST:Complements or Competitors?* San Francisco, California: Chappell & Associates.
2. Elkstein, M. (n.d.). *Learn REST: A Tutorial*. Retrieved February 1, 2013, from Learn REST: A Tutorial: http://rest.elkstein.org/
3. Francia, S. (2010, January 15). *SOAP vs. REST*. Retrieved March 15, 2013, from SOAP vs. REST: http://spf13.com/post/soap-vs-rest/
4. Gavin, D. G. (2009). A Comparison of SOAP and REST implementations of a service based interaction independence middleware framework. *Proceedings of the 2009 Winter Simulation Conference* , 3-6.
5. IEBC. (n.d.). Retrieved February 5, 2013, from http://www.iebc.or.ke/rpp/?keyword=20846558&submit=&check=set
6. Kishor, R. T. (2012). A Comparative Study of SOAP Vs REST Web Services Provisioning. *Journal of Information Engineering and Applications* , 2-4.
7. Kothari, C. (2004). *Research Methodology Methods and Techniques (Second Revised Edition).* Jaipur (India): New Age International (P) Limited Publishers.
8. Marston, T. (2004, May 2nd). *The Model-View-Controller (MVC) Design Pattern for PHP*. Retrieved May 14, 2013, from Rapid Application Development toolkit for building Administrative Web Applications: http://www.tonymarston.net/php-mysql/model-view-controller.html
9. Maven. (2011). *Lightweight REST framework*. Retrieved June 4, 2013, from Lightweight REST framework: http://essentialsource.sourceforge.net/documentation/rest.html
10. Nahon, J. (2011). *A Comparative Analysis of REST and SOAP*. Leeds: School of Computing University of Leeds.
11. Pavan, S. A. (2012). Comparing Performance of Web Service Interaction Styles: SOAP vs. REST. *Proceedings of the Conference on Information Systems Applied Research* (pp. 2-5). New Orleans Louisiana, USA: University of North Florida.
12. Potti, P. K. (2011). *On The Design Of Web Services: SOAP vs. REST*. Florida: University Of North Florida School Of Computing.
13. Singh, T. (2009, August 24). *REST vs. SOAP – The Right WebService*. Retrieved January 28, 2013, from REST vs. SOAP – The Right WebService: http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/#ixzz2IcJgJ2yO
14. Jon, F. (2007, July). *Which is better, REST or SOAP?* Retrieved July 11, 2013, from SOAP, REST, and More: http://msdn.microsoft.com/en-us/magazine/dd942839.aspx
15. wuher. (2010, November 12). *Rest vs. Soap. Has REST a better performance?* Retrieved July 11, 2013, from web services - Rest vs. Soap. Has REST a better performance? - Stack Overflow: http://stackoverflow.com/questions/4163066/rest-vs-soap-has-rest-a-better-performance

16. Rozlog, M. (2010, April 01). *REST and SOAP: When Should I Use Each (or Both)?* . Retrieved August 06, 2013, from REST and SOAP: When Should I Use Each (or Both)? : http://www.infoq.com/articles/rest-soap-when-to-use-each

Appendix A: Systems Specifications

All the experiments in this research were done in the same environment having the specifications shown below:

a. Hardware Specifications

System Manufacturer: Dell Inc.

System Model: Latitude E6400

Memory: 4 GB RAM

Processor: Intel(R) Core(TM) 2 Duo CPU P8600 @ 2.40GHz (2 CPUs)

b. Software Specifications

Operating System: Windows XP Professional (5.1, Build 2600) Service Pack 3 (2600.xpsp_sp3_qfe.130704-0421)

c. Running Applications / service

At time the experiments were being done, there were no other applications running and the background process were at machine level (no user processes running or started).

Appendix B: SOAP Implementation code snippet

a. The code snippets below shows the SOAP server methods that needs to be registered in the WSDL

```
$server = new soap_server();
$server->configureWSDL("partylist", "urn:partylist");
$server->register("getMembers",
   array("category" => "xsd:string"),
   array("return" => "xsd:string"),
   "urn:partylist",
   "urn:partylist#getMembers",
   "rpc",
   "encoded",
   "Get a listing of all the political party members");

$server->register("getParties",
   array("category" => "xsd:string"),
   array("return" => "xsd:string"),
   "urn:partylist",
   "urn:partylist#getParties",
   "rpc",
   "encoded",
   "Get a listing of all the political parties");

$server->register("getParty",
   array("category" => "xsd:string", "pid" => "xsd:int"),
   array("return" => "xsd:string"),
   "urn:partylist",
   "urn:partylist#getParty",
   "rpc",
   "encoded",
   "Get a political party given the party ID");
```

b. The code below show the WSDL required by SOAP in the implementation of the prototype.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:partylist"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:partylist">
<types>
<xsd:schema targetNamespace="urn:partylist"
>
 <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="getMembersRequest">
  <part name="category" type="xsd:string" /></message>
<message name="getMembersResponse">
  <part name="return" type="xsd:string" /></message>
<message name="getPartiesRequest">
  <part name="category" type="xsd:string" /></message>
<message name="getPartiesResponse">
  <part name="return" type="xsd:string" /></message>
<message name="getPartyRequest">
  <part name="category" type="xsd:string" />
  <part name="pid" type="xsd:int" /></message>
<message name="getPartyResponse">
  <part name="return" type="xsd:string" /></message>
<portType name="partylistPortType">
  <operation name="getMembers">
    <documentation>Get a listing of all the political party members</documentation>
    <input message="tns:getMembersRequest"/>
    <output message="tns:getMembersResponse"/>
  </operation>
  <operation name="getParties">
    <documentation>Get a listing of all the political parties</documentation>
    <input message="tns:getPartiesRequest"/>
    <output message="tns:getPartiesResponse"/>
  </operation>
  <operation name="getParty">
    <documentation>Get a political party given the party ID</documentation>
    <input message="tns:getPartyRequest"/>
    <output message="tns:getPartyResponse"/>
  </operation>
</portType>
<binding name="partylistBinding" type="tns:partylistPortType">
```

```
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getMembers">
    <soap:operation soapAction="urn:partylist#getMembers" style="rpc"/>
    <input><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="getParties">
    <soap:operation soapAction="urn:partylist#getParties" style="rpc"/>
    <input><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="getParty">
    <soap:operation soapAction="urn:partylist#getParty" style="rpc"/>
    <input><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
    <output><soap:body use="encoded" namespace="urn:partylist"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
</binding>
<service name="partylist">
  <port name="partylistPort" binding="tns:partylistBinding">
    <soap:address location="http://localhost/rpsoap/rpserver.php"/>
  </port>
</service>
</definitions>
```

c. The code snippets below shows the SOAP client that uses the methods defined in the SOAP server. The code pulls data from the MYSQL database to the browser

```
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');
require_once "lib/nusoap.php";
class rpclient extends CI_Controller
{
		function __construct(){
		parent::__construct();

		// load model
```

```
                        $this->load->helper('url');
                        $this->output->enable_profiler(TRUE);
                        }
            function index()
{
                        // load the client
                        $client = new nusoap_client("http://localhost/rpsoap/rpserver.php");
                        //$client = new nusoap_client("rpsoap.wsdl", true); -- using the existing wdsl
location: /wdsl/rpsoap.wsdl
                        $error = $client->getError();
                        if ($error) {
                            echo "<h2>Constructor error</h2><pre>" . $error . "</pre>";
                        }
                        $result = $client->call("getParties", array("category" => "parties"));
                        //use the ?url to get the parties var and display
                        if ($client->fault) {
                            echo "<h2>Fault</h2><pre>";
                            print_r($result);
                            echo "</pre>";
                        }
                        else {
                            $error = $client->getError();
                            if ($error) {
                                echo "<h2>Error</h2><pre>" . $error . "</pre>";
                            }
                            else {
                                echo "<h2>Parties</h2>";
                                echo "<pre>";
                                //echo $result;
                                print_r(json_decode($result, true));
                                echo "</pre>";
                            }
                        }
                        echo "<h2>Request</h2>";
                        echo "<pre>" . htmlspecialchars($client->request, ENT_QUOTES) . "</pre>";
                        echo "<h2>Response</h2>";
                        echo "<pre>" . htmlspecialchars($client->response, ENT_QUOTES) .
"</pre>";
            }// end of index func
            function members(){
                        // load the client
                        $client = new nusoap_client("http://localhost/rpsoap/rpserver.php");
                        //$client = new nusoap_client("rpsoap.wsdl", true); -- using the existing wdsl
location: /wdsl/rpsoap.wsdl
                        $error = $client->getError();
                        if ($error) {
                            echo "<h2>Constructor error</h2><pre>" .$error. "</pre>";
                        }
                        $result = $client->call("getMembers", array("category" => "members"));
                        //use the ?url to get the members var and display
                        if ($client->fault) {
                            echo "<h2>Fault</h2><pre>";
                            print_r($result);
                            echo "</pre>";
                        }
                        else {
                            $error = $client->getError();
                            if ($error) {
                                echo "<h2>Error</h2><pre>" . $error . "</pre>";
                            }
                            else {
                                echo "<h2>Members</h2>";
                                echo "<pre>";
                                //echo $result;
                                print_r(json_decode($result, true));
                                echo "</pre>";
                            }
                        }
                        echo "<h2>Request</h2>";
                        echo "<pre>" . htmlspecialchars($client->request, ENT_QUOTES) . "</pre>";
                        echo "<h2>Response</h2>";
                        echo "<pre>" . htmlspecialchars($client->response, ENT_QUOTES) .
"</pre>";
            }// end of members func
            function party(){
                        //get the id
                        $ret_id = $this->uri->segment(3, 0);
                        // load the client
                        $client = new nusoap_client("http://localhost/rpsoap/rpserver.php");
                        $error = $client->getError();
                        if ($error) {
                            echo "<h2>Constructor error</h2><pre>" . $error . "</pre>";
                        }
                        $result = $client->call("getParty", array("category" => "party", "pid" =>
$ret_id));
                        //use the ?url to get the authors var and display
                        if ($client->fault) {
                            echo "<h2>Fault</h2><pre>";
                            print_r($result);
                            echo "</pre>";
                        }
                        else {
                            $error = $client->getError();
                            if ($error) {
                                echo "<h2>Error</h2><pre>" . $error . "</pre>";
                            }
                            else {
                                echo "<h2>Party</h2>";
                                echo "<pre>";
                                //echo $result;
                                print_r(json_decode($result, true));
```

```
                        echo "</pre>";
                    }
                }
                echo "<h2>Request</h2>";
                echo "<pre>" . htmlspecialchars($client->request, ENT_QUOTES) . "</pre>";
                echo "<h2>Response</h2>";
                echo "<pre>" . htmlspecialchars($client->response, ENT_QUOTES) .
"</pre>";
        }// end of index func


}//end of rpsoap class
?>
```