# Analysis Design and Outcome for Mobile Cloud Computing On Various Platforms

Mr. Rushi Raval 1st
Research Scholar, Department of Computer Sci.
Saurashtra University,
Rajkot, India
*rushionline@ymail.com*

Dr. Atul Gonsai 2nd
Associate Professor, Department of Computer Sci.
Saurashtra University,
Rajkot, India
*atul.gosai@gmail.com*

***Abstract***— Now-a-days web services (WS) are essential part to interact communication between internet and mobile clients. To build WS, researcher use SOAP based WS and REST based WS Architecture as well. Here, researcher carries out work and developed both WS with different parameters on to different cloud platforms. We were design cloud web services model with SOAP and RESTful applying with JSON based Cloud Web Services respectively in both of the CWS. The CWS are used by entirely different CWS application-based model to interact data context using diverse cloud servers/platforms. We have conducted different test cases on both and tested on web servers and on cloud servers such as Apache-Server, Windows Phones, Heroku and Google App Engine (GAE). Further, researcher also developed two models for mobile clients such as Native app model and Web work app model. Result of research suggestions that the REST based CWS is better in performance than SOAP based CWS and web work app model is to do better with cross compatibility features.

***Keywords***- *Mobile Cloud Computing; Cloud Web Services; Cloud Platforms; CWS Communication; Android; Windows Phone;*

*****

## I. Introduction

Cloud platforms provide wide range of features for computer related terms. Cloud computing usage will enhance the robustness, reliability, and scalability. Web-Services (WS) is a utility written as an application or module and established with consumption of differ technologies such as, JSON, CSV, RESTful CWS placed on cloud-server which can re-claimed via difference protocols / methods such as, HTTP; GET; PUT; POST; HEAD in another application such as a client-server or in distributed architecture manner. As CWSs are platform-unbiased and in general data context represent as text-based this can proposal, run as responsive resource and access on numerous platforms with dissimilar technologies.

With this paper, researcher explores a reasonably new methodology for REST and SOAP based API to communicate and transfer data context from “r-restful-client” as an app model to “r-restful-ws” another app model using cloud servers such as, G.A.E. and Heroku. The REST, SOAP, CSV and JSON (Java-Script Object Notation) are the standards which are used to join up with each-other over cloud computing platforms. These types of cloud web services are identically dynamic and economical as well. The CWSs can be functional in various different differ architecture methods and styles and frame as per mobile clients characteristic. For example, REST is lightweight as associated to SOAP standards and is frequently based on URL; numerous IT companies have used REST based WS for their architecture such as, Amazon-(AWS-S3), e-Bay, Flickr, and Yahoo pipes.

The leading concentration of this paper is to Analysis & Design of a REST and SOAP CWS application-based model for apps grouping on cloud-server/platforms. This paper is projected to design a cloud web services for mobile client with the implementation of several app based model such as, r-restful-client / r-soap-client apps. The CWS app models are designed and coding into PHP and C#.NET. For data storage point of view researcher also use data content format for ex, CSV and JSON into cloud platforms. Furthermore, as growing number of mobile clients and obtainability of CWS also drives the essentials of adapting and personalizing service-based mash ups.

## II. Contextual of Cloud Web Services

Many IT enterprises and commercial industries are being implemented cloud-services as their part of the organizations, since cloud computing offers several services to above kind of organisations such as, IaaS, DaaS, SaaS, MaaS and PaaS. Cloud platforms offer to host resources such as, WS, Web API, and other applications model as well.

WS is presently the foremost technology for delivering services to the end-users. In a mobile environment, most of the challenges are interrelated to platform and resource / infrastructure restrains. Because RESTful WS only requires HTTP protocol, it uniforms the mobile environment in better way. Caching and enhancing / compressing are two approaches to deal with bandwidth restrain. In our approach, the middle ware provides RESTful interfaces for mobile users. It also caches and optimizes service results from Cloud Service for e.g., Google App Engine, Microsoft Windows Azure, and Heroku.

Web Service is a broadly implemented methodology for given that services, but most prevailing web services in the Cloud are not attentive of mobile clients [3]. REST-WS is specifically designed for light-weight and elastic interfaces, such as, mobile web service communication. In this paper researcher has implemented cloud web services with proposed app model UI into cloud environment.

## III. Objective and Proposed Design for CWS App Model

The objective of this paper is to do Analysis and Design of a cloud web services model for mobile clients based on SOAP and REST CWS structures and provide service mash up concept as well, hence design several application based models for e.g., “r-restful-ws” CWS app-model with added mash up concept and for mobile client design r-restful-client and r-soap-client CWS app models. To achieve objective, researchers have intended above said CWS application models with SOAP and RESTful based CWSs and also use JSON and CSV data format for storing and parsing data, communicate with CWSs uses JSON format as light-in-weight and support for thin clients such as mobile users, smart-phone clients, as well.

Researcher has select Native app model for windows based clients and Web work app model for android based clients and also for implement cross compatibility for mobile users. Following is the CWSs modules shown with their functions.

Native CWS App Models:

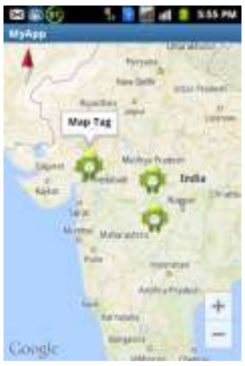
- Built upon specific platform or device dependent
- Rich GUI features
- Use C#, Windows Phone SDK, and Silverlight
- Here use a Windows platform 7.1 as Native CWS app model

Web-work CWS App Models:

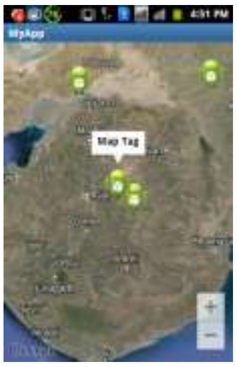
- Built upon cross-platform or device independent
- Additional method to implement the mobile CWS
- The user app runs on a Web browser
- Use PHP, JS, HTML, and CSS
- Here use an Android platform 4.4.4 as Web-work CWS app model.

## IV. IMPLEMENTATION OF CWS APP MODELS

The aim of this paper is to design CWS app models for Mobile-clients, for validate the mobile users design; Researcher involved the proposal with a various types of apps are developed. Starting from initial level we have created an app for providing private service mash up platform in which mobile users have facility to see live map, mark the locations into app, and also tag that locations with user defined remarks using android based app. Figure 1 is the screenshot which displays that app screens [1]:



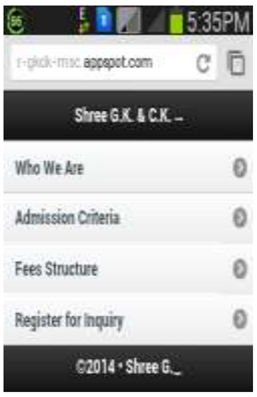
Live map (Normal view)



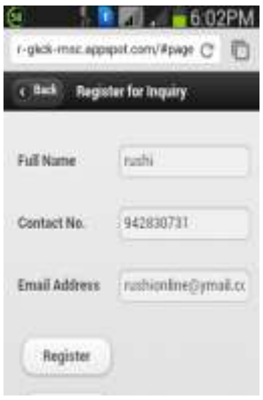
Live map (Hybrid view)

Figure 1: Live Map on Android based Smart-phone

Researcher has designed another app for described the CWS interrelated work with keep records of the students, and students can see the updates related to admission criteria, fees structure as day-to-day operation. Admin side also developed to download the students’ data as information in JSON or CSV format as well; figure 2 is illustrating the mobile client design for app named “r-gkck-msc” [2]:



Main Screen (Student as User)



Full Name rushi
Contact No. 942830731
Email Address rushionline@ymail.cc

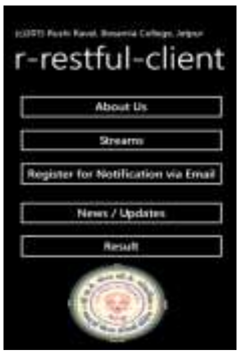
Inquiry Screen (Student as User)

Figure 2: Displays student inquiry procedure at college campus using Mobile App

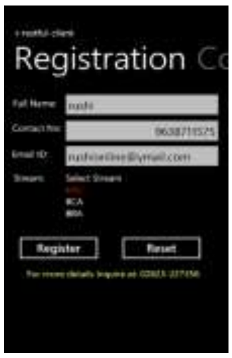
Furthermore, researcher implemented current Native app model for Windows-phone and also project for android as web work app-model to support added mobile clients as well, consider mobile user perception as college-students, defined as r-restful-client. The app is re-functional with the mobile user design on Windows Phones platform. By the usage of this app model, student as mobile users can do following things:

- Check their class information, course and college information
- Check the updated news information course wise
- Check their results from particular departments and also able to getting news of their department via an email only for those students who are registered.

*Figure 3* is some screenshots of the r-restful-client on Windows platform and *Figure 4* is some screenshots of the r-restful-client on GAE platform with Android mobile client [4]:



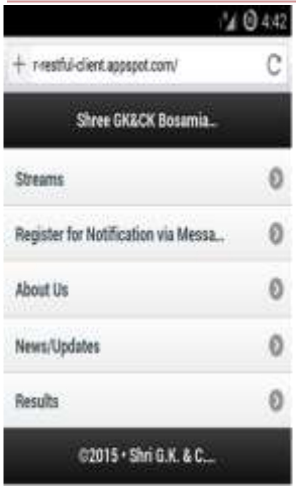
Home Screen (Student as User)



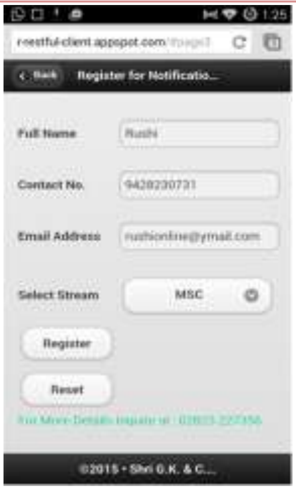
Full Name rushi
Email ID rushionline@ymail.com

Registration Frame (Student as User)

Figure 3: Mobile client UI design of r-restful-client on Windows-Phone Platform



Home Screen (Student as User)

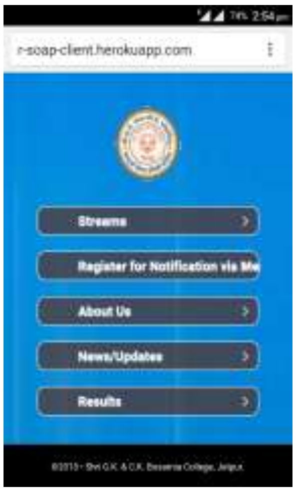


Full Name: Rushi
Contact No.: 9428230731
Email Address: rushionline@ymail.com

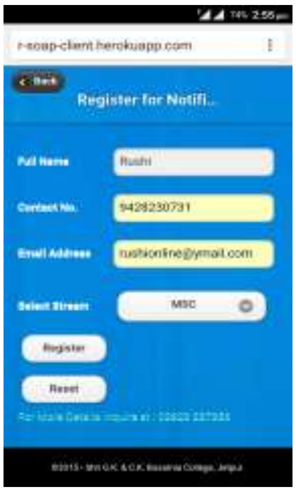
Registration Screen (Student as User)

Figure 4: Mobile client UI design of r-restful-client on GAE Platform

Next researcher to do comparison with SOAP CWS with REST CWS app model for that another one web work model created for that purpose and deploy on to cloud platform named Heroku. *Figure 5* is some screenshots for illustrated the procedure of SOAP CWS app model.


r-soap-client.herokuapp.com
Streams
Register for Notification via Me
About Us
News/Updates
Results
©2015 - Shri G.K. & C.K. Bosamia College, Jetpur

Home Screen (Student as User)


r-soap-client.herokuapp.com
Back
Register for Notifi...
Select Stream
MSC
Register
Reset
©2015 - Shri G.K. & C.K. Bosamia College, Jetpur

Full Name: Rushi
Contact No.: 9428230731
Email Address: rushionline@ymail.com

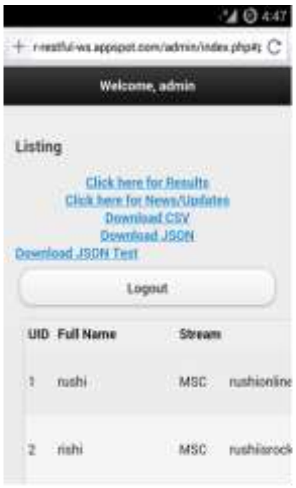
Registration Screen (Student as User)

Figure 5: Mobile client UI design of r-soap-client on Heroku Platform

Researcher also include a mash up concept with administrative purpose, for that we have developed one more CWS app model with motive to store student registration data, course information, news context, result information named as r-restful-ws. In this app model researcher have interrelated SOAP to REST CWS communication and windows phone client (r-restful-client) plus android (r-restful-client) client communication to this mash up app model. By the use of this CWS app model administrator can do following things, for e.g.:

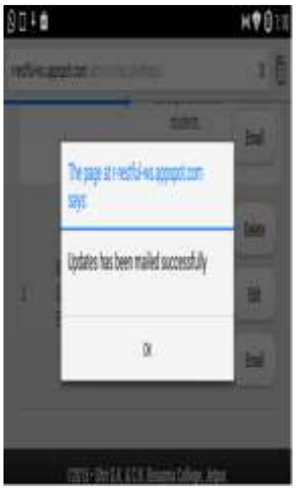
- See the registered student data
- Download that data in JSON or in CSV data context format
- Add, Edit, Delete the news or updates stream wise
- Send an e-mail to particular group of students (stream wise)
- Add, Edit, Delete the result information stream wise

Following *figure 6* is demonstrates the mash up CWS app model with the administrative level purpose



| UID | Full Name | Stream | |
|---|---|---|---|
| 1 | rushi | MSC | rushionline |
| 2 | rishi | MSC | rushiisrock |

Student information Screen (Admin as User)


r-restful-ws.appspot.com
The page at r-restful-ws.appspot.com says:
Updates has been mailed sucessfully
Ok

Send E-mail Screen (Admin as User)

Figure 6: Mobile client UI design of r-restful-ws on GAE platform

Researcher has implemented above mentioned Web-work CWS app models on GAE and Heroku cloud-platform as well to achieve optimum results for the mobile users. Furthermore, researcher procedures JSON and CSV data-context formats for data storage and parsing comparison analysis as well.

## V. Test And Outcomes Of CWSs App Model

Researcher has tested CWS app models with different experimentations. Aimed at, researcher implements RESTful and SOAP based CWS and set up them on different servers for e.g., the Apache web server, cloud servers such as G.A.E. and Heroku. Apache-WAMP server runs on DELL-I-5110x companionable Workstation with 2.30-GHz Intel® Core™ i3-2350-M processor with 4GB-RAM, where OS operates as Windows 7 Ultimates with SP1, and for smartphone apps (mobile-user) functional with Windows Phone 7.0, MS-Silverlight for Windows Phone Toolkit (MSS-WPT), and running on Windows-Phone OS 7.1 Emulator. Heroku and GAE service platform was occupied on free-quota origin to deploy and test our CWS app model.

**Consuming REST CWS App Model for Native UI:**
To dimensions RESTful CWS for mobile clients, Researcher has used tools and editors for developed Native UI, VS-2010 IDE. RESTful based CWSs are designed by Windows Phone SDK 7.0 version and RESTful API with C# as native language.

**Consuming REST CWS App Model for Web-work UI:**
To dimensions REST CWS, Researcher has used Notepad++ with v6.8.6 (Freeware) IDE. RESTful based CWSs are designed by RESTful API with PHP server-side language.

For consuming RESTful CWS in Android smart phone platform and several other mobile clients for e.g., laptop, Researcher has as well advanced REST CWS with Web-Work UI design support. This CWS app models created and hosted from a cloud platforms that makes data to and forward from CWSs. For storing data context point-of-view, Researcher has used JSON plus CSV data format, for data parsing and communication used JSON format. Moreover, it uses REST CWS that is receiving and sending data to and forward using http-request (URI). It usages POST method as request for acquiescing students' particulars, to get informed by e-mail info / data shown to mobile-users is in point of fact, JSON data, which has been make out using PHP scripts.

For consuming SOAP CWS with Web-Work design support, researcher has also established SOAP CWS app model, named as r-soap-client. It uses SOAP CWS with nuSOAP API that requests SOAP showed method by calling soap-client object's call process that agrees arguments as an array. This CWS app models usage JSON as request-response, data contexts show to the mobile clients.

Consuming RESTful and SOAP based CWSs over the middle ware, in this experimentations associate the above accompanying with app considered r-restful-client and r-soap-client CWS app model interfaces as a client. r-restful-ws offer both REST and SOAP CWS interfaces for educational-service. The "r-restful-ws" CWS reoccurrence outcome in either CSV or JSON format for registered students' facts and for other procedures such as News / Updates, Results return outcome in JSON format. The established CWS is as following:

- About us: This will returns CWS app information.
- Stream: returns a list of contents match with the keywords BBA / BCA / MSC.
- Register: registered student info, which will stores at CWS app model named, r-restful-ws and precedes a reactive communication at mobile user side.
- Updates/ News: returns a list of news/updates associated to contest with the keywords BBA / BCA / MSC.
- Results: returns a list of marks occupied by the appropriate stream to match with the keywords MSC / BCA / BBA.

For Administrative standpoint:

- Form registration records and similarly capable to take those students data who have recorded via CWS app models such as, Native / Web-work in JSON / CSV design.
- Add, Edit, and Delete Updates/ News.
- E-mail the most recent Updates/ News to the students relevant to specific stream.
- Add, Edit, and Delete Results.

To test comparison between REST and SOAP CWS app model also middle ware and local web server as well. For that different parameters are set. The 6000 (MS) time set as maximum page load time and 3000 (MS) time is maximum time to first byte. The middle ware is runs on the GAE, Heroku, and local web server [4].

The load initiator gives directions HTTP-request based on defined virtual user profiles with testing tool LoadUIWeb. Researcher has defined detailed experimentation details in previous work [4]. Figure 7 and 8 indications a column graph associating the outcome times of dissimilar experiments. There is in the clouds related with the middle ware. Still, outcome optimization provocatively declines the bandwidth phase.

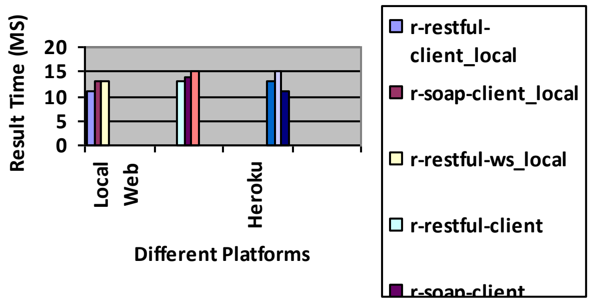


Figure 7: Column graph for Outcome time to first byte receive (MS)

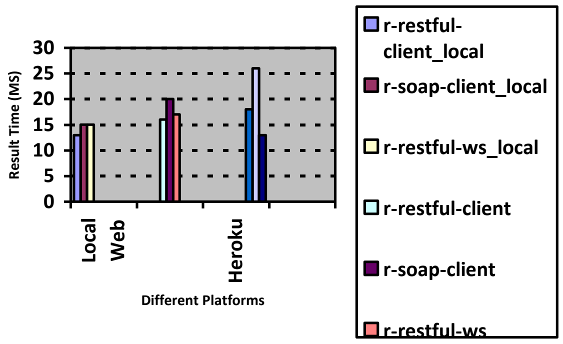


Figure 8: Column graph for Outcome time to Page Load Time (MS)

- Middle ware v/s. Local: Associate the experiments 1 to 9 [4] for equally views such as outcome time to first byte receive and outcome time to Page Load Time, whether the r-restful-ws_local and r-restful-ws CWSs return JSON data format, the middleware addition a certain extent of overhead (on an middling 1.050s to 3.700s) on the outcome time for dissimilar middle ware platform. By way of the middle ware, it doesn't see to any kind of processing of the CWS outcomes; specific extent of in the clouds is habitually originated by network prospect amongst the middleware and mobile clients.

- REST CWS v/s. SOAP CWS: By way of the experiments 7, 8, and 9 presented [4], SOAP CWS has great extent of outcome times than the rest of the experiments 4, 5, and 6 [4] with RESTful CWS. SOAP-CWS is longwinded procedure, which means there are certain additional content requests to be communicated. In addition, processing time pre-mandatory for the middle ware producing nuSOAP object from the SOAP CWS message / communication. The lead of SOAP CWS is result access is easy, however RESTful CWS is also produce outcome in an enhance way.

For next experiments researcher has studied the different data format to storing a message for a request-response parsing through the app UI models and determines to concentrate only on JSON and CSV formats. Researcher also include test case for bandwidth and parsing time association of JSON and CSV. Furthermore, Researcher defines that CSV and JSON are two extensively used formats for transmitting CWS messages but as CSV is a slightest easy-going data formats for transferred message over CWS app UI model. As mobile users have inadequate processing power and limited bandwidth, Researcher has studied JSON format for transmitting CWS messages and for storing data studied JSON and CSV format as CSV uses fixed format for ex., Tabular view to store data context and consequently CSV consume less bandwidth. This experiment calculates the use of CSV and JSON. As JSON is a light-weight context for parsing message as well as for storing a data through the CWS app model and CSV similarly do same thoughtful but only for storing a data. To added define that, Researcher use an “r-restful-ws” CWS app model which returns the utmost current students registration updates in both JSON and CSV format. The mobile client describes the CSV result and JSON result with PHP DOM parser. Besides, Researcher test some experiments related to JSON and CSV parsing from Apache J-Meter v2.13 [5].

| Mobile platform | Usage | Format | Data context size (KB) | Avg. Parsing time (MS) |
|---|---|---|---|---|
| Android Phone | Data storage | CSV | 2.71 | 2955 |
| | | JSON | 5.05 | 1074 |
| Windows Phone | Data storage | CSV | 2.71 | 2948 |
| | | JSON | 5.05 | 1776 |

Table 1: Parsing time and size of CSV and JSON data context on different mobile clients

Table 1 demonstration the data context size and avg. parsing times for the JSON and CSV messages (data context range about 55 to 60) determined 50 autonomous trials on an Android and a Windows Phone. 1st associating the size, the size of CSV outcome is 2.71KB and 5.05KB for JSON. To indicate the same kind of data, the CSV format requires less bandwidth. 2nd subsequently the parsing time, parsing CSV message is additional resource over-whelming than parsing JSON message on both Android Phone and Windows Phone. So, for data storing consequence CSV format consume less space as compare to JSON format. Although for data context parsing consequence CSV format consume more space as compare to JSON format. At this point slowness is due to complication of parsing and also CSV format is not fit for transferred data context as request-response over CWS app UI model. To conclude, JSON format has identical flexible for transferred data context as request-response and correspondingly stable parsing time. And yet, it is very hard to signify compound data structure in JSON component format.

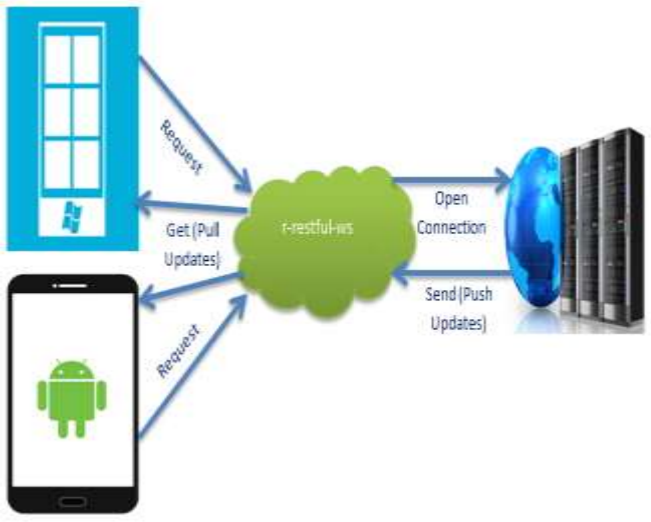


Figure 9: Push and Pull using CWS app UI

Further test case implementations update the mobile clients with Notification such as, Incoming e-mail features. Researcher has been analyse that short of updates sending to the mobile clients CWS app UI model is not deliver full-fledge functionality as well as for CWS app model require a mash up feature and with the additional this researcher are able to send the latest news / updates to the registered mobile clients (here, consider the registered students). Here *(See figure 9)*, two approaches to request data framework from a cloud server with the CWS app model, pull and push. Pulling means that mobile users irregularly connects to the cloud server with the CWS app model, checks for and pulls (gets) newest updates and then later wind-ups the current connection and disconnects from the cloud server. The mobile client reappearances this entire process to catch updated about new info / events. In this methodology, the mobile clients intermittently PULLs the new updates / events from the cloud server. Pushing means the mobile users open a current connection to the cloud server and preserves it continually active. The cloud server will push (send) all news / updates to the mobile client consuming that CWS app model. In other words, the cloud-server PUSHES the new updates / news to the mobile clients.

Researcher show e-mail pathway for incoming e-mail *(see figure 10)*, the server drives the alteration via the email account setup on the mobile client when the news / updates add, edit or update.

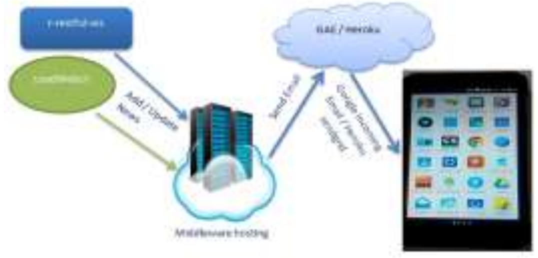


Figure 10: Incoming email pathway for mobile client

To compute and assessment pull and push, Researcher extent the subsequent values in his experimentation test case also using testing tool named Fiddler [6]. Fiddler is a HTTP-based debugging and proxy-server application, it apprehensions several protocols such as http and https movement and generate logs.

a) Bandwidth usage: The total data transferred for the mobile client through the test case consist of upload & download for dissimilar data format. Larger bandwidth means that more extent of the file is being relocated at any given time.
b) RTT used: The time needed for a network packets or messages to transportable from the source to the destination and back as of the destination to the source. Round-Trip-Time is used by assured routing algorithms to backing in scheming optimal routes.
c) Energy consumption: Network edges for ex., HTTPS, HTTP and SMTP consumes energy. The additional amounts of network interfaces are involved; as a result the added energy is used up.
d) Response-bytes by Content Type: The bytes are receiving as a response and considered as content such as HTML, CSV and JSON with its headers info.

| Cloud Service | Data Format | Bandwidth Used (Bytes) | | Overall Elapsed (MS) | Energy Consumption (No. of send req. and get No. response) |
|---|---|---|---|---|---|
| | | Upload | Download | | |
| **GAE** | **JSON** | 451 | 1579 | 01.240 | 1 (Response Count: 60) |
| | **CSV** | 456 | 1444 | 01.831 | 1 (Response Count: 60) |
| **Heroku** | **JSON** | 451 | 558 | 00.321 | 1 (Response Count: 03) |
| | **CSV** | 456 | 430 | 00.325 | 1 (Response Count: 03) |

Table 2: Experiment case for Pull investigation

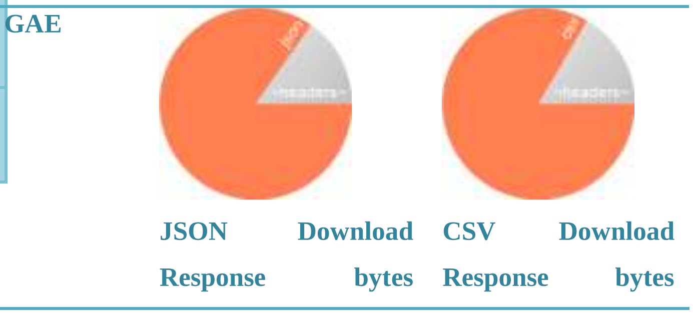
Table 2 indications the bandwidth used (bytes) and energy consumption of the mobile user during the online activity such as 15 to 20 minutes. Figure 11 indicate the Response bytes of total 56-60 update for GAE and total 4-5 update for Heroku and also recognize Push investigation.

a) Bandwidth usage: To assessment pulling investigation on GAE for JSON, the user sends 451Bytes as headers information and receives 1,579Bytes data with 241Bytes as headers information and 1,338Bytes as body in total. Aimed at mobile users the bandwidth difference is affected by the message headers of be different protocols such as SMTP, HTTP, and HTTPS.

b) Energy consumption: Aimed at the pulling investigation, the user sends 1 HTTP GET request and receives 60 responses for GAE and 03 responses for Heroku in total. The pull investigation consumes further energy. At this point, keep in mind that the energy consumption can be reduced or compact by cumulative the pull intervals. And yet, fewer recurring pulling increases the overall elapsed time.

Figure 11: Estimated World-Wide RTT Performance chart using Fiddler [6]

c) RTT used (see Figure 11): For the pulling investigation, the RTT time is nearly constant for above stated regions with various devices, as the middle ware (CWS App Model) receives update at continual rate, the time differ amongst each thin client pulls and fresh updates in the middle ware is also recurrent. Still, it is exact improbable the update occurs next to a constant rate in perceptible circumstances. The rate of fraction for pull essentials is aware permitting to the time circulation of updates, if any. For the push investigation, the RTT and elapsed time fluctuates a lot, as quite a lot of e-mails are batched into one pushing message.



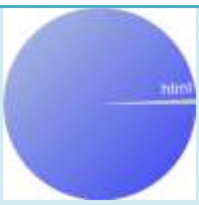
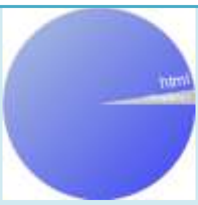
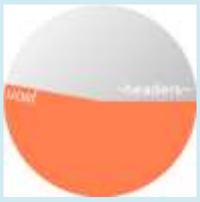
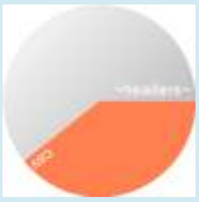
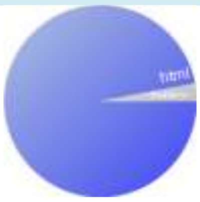
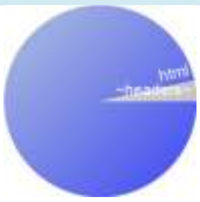
| | | |
|---|---|---|
| | **(Response Count: 60)** | **(Response Count: 60)** |
| |  |  |
| | **HTML Email Response bytes (Push)** | **HTML Email Response bytes (News/Updates-Pull)** |
| **Heroku** |  |  |
| | JSON Download Response bytes **(Response Count: 03)** | CSV Download Response bytes **(Response Count: 03)** |
| |  |  |
| | **HTML Email Response bytes (Push)** | **HTML Email Response bytes (News/Updates-Pull)** |

Figure 12: Response bytes by Content-Type and Push investigations

d) Response-bytes by Content Type (see Figure 12): For the GAE, JSON download investigation, the response bytes acknowledged as Application / JSON: 1338, Headers: 241 and CSV download experiment, the response bytes received as Application/CSV: 1205, Headers: 239, For the pushing testing (r-restful-ws.appspot.com/admin/), whole e-mail message is getting HTML Electronic mail response bytes acknowledged as Text/HTML: 31,871, with headers 266Bytes and for the pulling carrying out tests (r-restful-client.appspot.com/), each news/updates is getting response bytes acknowledged as Text/HTML: 6,192 with headers 152Bytes. For the Heroku, JSON download experiment, the response bytes acknowledged as Application/JSON: 295, Headers: 263 and CSV download experiment, the response bytes acknowledged as Application/CSV: 261, Headers: 169, Aimed at the pushing conducting tests (r-restful-ws.herokuapp.com/admin/), whole electronic message is getting HTML Email response bytes acknowledged as Text/HTML: 11,369, with headers 352Bytes and for the pulling testing (r-restful-client.herokuapp.com/), each news/updates is getting response bytes acknowledged as Text/HTML: 5,968 with headers 214Bytes.

Moreover investigation, Researcher has study related to performance, scalability and robustness concerns for the CWS app model design which were hosted on different cloud-services (Heroku and GAE). The key task of proposed middle ware has distinction and service-mash up which associates some goings-on such as, CPU share, multi-tenant, RAM and network I-O operations. When the middle ware gets service-outcomes from dissimilar cloud-services, it procedures out-bound network connections. After the middle ware receives the responses, it studies and pools them. In defined method, it demos the response-time of the middle ware for dealing out a service-mash up request, how the outcome time of a mash up req. variations when the capacity of the middle ware rises and at which request-rate the middle ware go wrong to response. The examination server is Heroku "railgun" Dynos instance and load designer is on Heroku 1x-4x with 512MB-RAM, 1x CPU segment with multitenant instance and for GAE server is standard 1 CPU with 3.75GB-RAM, 2.3GHz Intel Xeon E5 v3 with 2.75 GCEUs.

Figure 13 demos how the middle ware measures a mash up call response thru usage CWS app model. When the middle ware takings a HTTP(S) GET request, this one first achieves service-based outcomes from the Cloud-Services. For pretending CPU share totalling, the middle ware sending a GET request via "r-restful-client" app model with "J-Meter" [5] tool and create request up to 1 -to-5000 users as a user-group (Thread). J-Meter is a testing tool established in Java and considered for load test, ration performance as well as other test purposes such as Cloud web services, web dynamic lang. it can be similarly used for simulate a heavy load on to the server, objects (CWSs) or grids to analyse their strength and overall performance. Final, it precedes a response with the designed outcome to the mobile users.

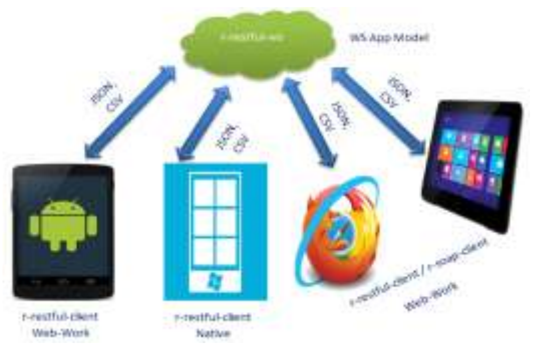
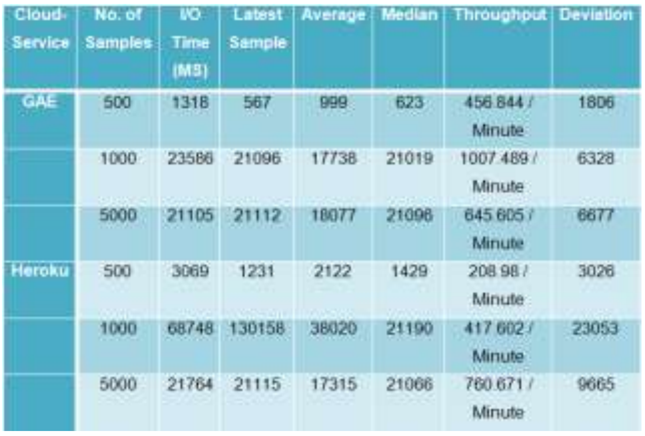


Figure 13: Process way of a mash up request

The interval of each load assessment case is 5, 10 to 20 minutes. Response time is planned every 1 sec. The working outcome is presented in figure 14 for GAE and figure 15 for

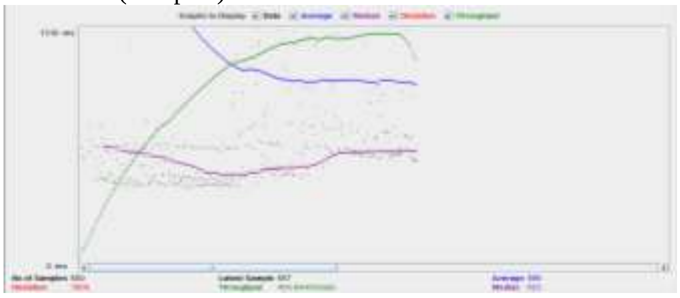
Heroku. Variety engaged for this experiment case is 1-500, 1-1000, and 1-5000 samples for each cloud services, See Table 3 for details of sample outlines.



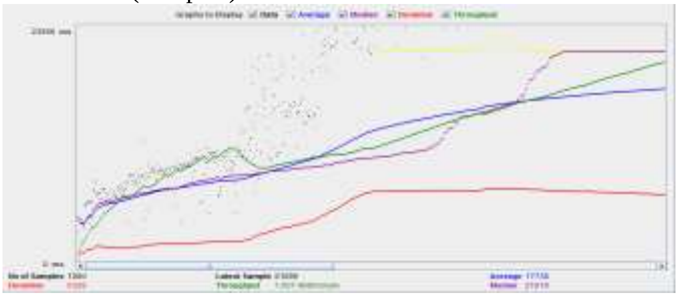
| Cloud-Service | No. of Samples | I/O Time (MS) | Latest Sample | Average | Median | Throughput | Deviation |
|---|---|---|---|---|---|---|---|
| GAE | 500 | 1318 | 567 | 999 | 623 | 456.844 / Minute | 1806 |
| | 1000 | 23586 | 21096 | 17738 | 21019 | 1007.489 / Minute | 6328 |
| | 5000 | 21105 | 21112 | 18077 | 21096 | 645.605 / Minute | 6677 |
| Heroku | 500 | 3069 | 1231 | 2122 | 1429 | 208.98 / Minute | 3026 |
| | 1000 | 68748 | 130158 | 38020 | 21190 | 417.602 / Minute | 23053 |
| | 5000 | 21764 | 21115 | 17315 | 21066 | 760.671 / Minute | 9665 |

Table 3: Study of sample outcome for GAE and Heroku CWS App Model

500 Users (Samples):



1000 Users (Samples):



5000 Users (Samples):

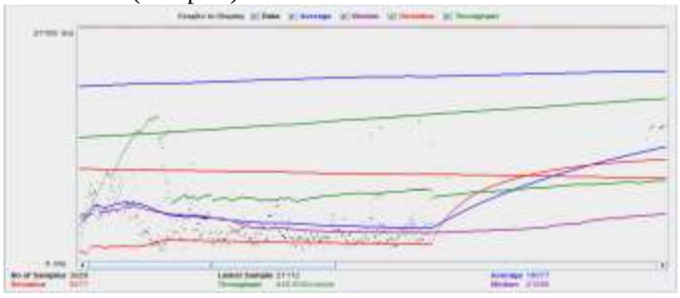
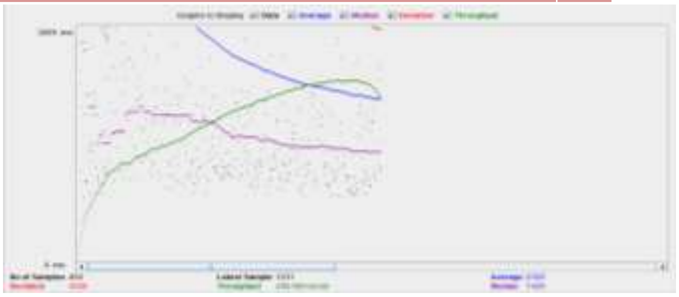
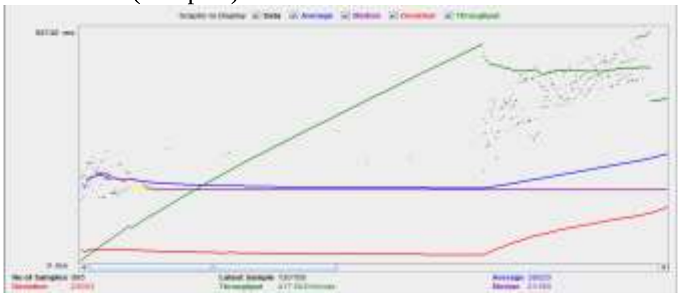


Figure 14: Response time for GAE

500 Users (Samples):



1000 Users (Samples):



5000 Users (Samples):

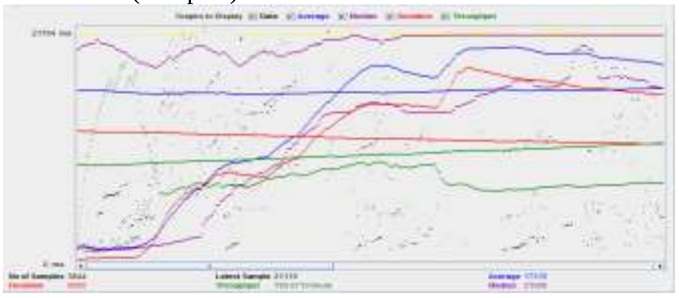


Figure 15: Response time for Heroku

The outcome of experiment cases shows that GAE balances. The avg. and median response time is lower for GAE. The throughput is 456.844 / min. for 500 request rates, 1007.489 / min. for 1000 request rates, and 645.605 / min. for 5000 all request rates displays high accessibility excluding some exceptions. Though, there are certain ideas where the maximum response is unrivalled. As resources use by App Engine is of Google's substructure, the span of resources for an app model is not determined at all the time. Such as, when Google understandings a high-volume of load origin, a GAE app model may acquire less resource, therefore response becomes slower.

Heroku with Web dynos, with 450 trials, the average response time is 2122 and nearby about half of the requests failed or dropped. The median time 1429 with throughput of 208.98 / min. and deviation is 3026, with other trial case, such as with 1000 samples, the average response time is 38020. The median time 21190 with throughput of 417.602 / min. and deviation is 23053, and for 5000 samples, the average response time is 17315. The median time 21066 with throughput of 760.671 / min. and deviation is 9665. This can be defined by the dynos for cloud mode of Heroku. A Heroku dynos free occurrence sleeps minimum 6HRS / Day. Heroku dynos doesn't share resource with each other, in added words free Heroku dyno type can only use with the free Heroku dyno type for both web and worker Heroku dynos, it can't be mix up with other Heroku dynos type. With the similar amount of resource, thus, it is feasible as the response time is increases as the load turn into increases.

## VI. Conclusions

Researcher has developed the App UI Models for RESTful CWS and for SOAP CWS. For succeed the purpose of performance, as student mass will more or less uses a smart-phone. Regularly a huge number of students are revised to work under the Mobile environment. Execution this point in mind researcher has developed a PHP based web work app model and also developed a native app model based on windows phone platform named r-restful-client for the front-end UI purpose. This proposed work more relating to the mobile users with estimated the cross platform capability, researcher has proposed directing CWSs request from the mobile users, this assessment case investigation done on Window Phone and Android Phone as well, which can able to send a request as RESTful through the r-restful-client app to the middle ware.

Researcher has focus on data formats for mobile users such as CSV and JSON, as they are the light-weight processing, straightforwardness to produce a request-response from the CWSs App Model to the mobile user. Table 1 shows the parsing time and size of JSON and CSV data context. In this investigational work researcher has compare the outcome size for the corresponding data formats as outcome CSV format consume less bandwidth, additional analyse parsing time for both of the data formats as outcome JSON format consume less space as compare with CSV data format.

Researcher has an objective to classify component based energy optimization and similarly recognize scalable platform for the mobile client's environment. Aimed at this researcher has applied a Pull and Push using CWS App Model. In the tentative work for Push and Pull, researcher usages an electronic-mail as the push method to send updates / news to the registered users and associates it to the HTTP pull method on Android phone / Windows-phone. Towards analyse and experiment pull and push, Researcher scope the specific parameters such as, Energy consumption, Bandwidth used, RTT used, and Response bytes by Content-Type. Table 2 demonstrations the Experiment case for Pull investigation and Figure 11 show the estimated RTT for worldwide.

Furthermore measure the response bytes by Content-Type such as for Application/CSV, Application/JSON, and Text/HTML. As an outcome of bandwidth experiments, inclusive JSON data format consume less bandwidth as compare to CSV data format on to the cloud environment, further JSON consume less total elapse time as compare to CSV over cloud platforms. Additionally, pull research consumes additional energy and it can be reduced by cumulative the pull intervals. At this point elapsed time and RTT are almost constant for stated regions with a number of devices established on different operation specifically push investigation implicates less network interfaces. Besides, response bytes for communication (GAE investigates) consume more bytes (31,871) as compare to regular pull investigates getting response bytes acknowledged as Text/HTML: 6,192 (updates/ news) and for communication (Heroku investigates) consume more bytes (11,369) as compare to consistent pull investigates getting response bytes acknowledged as Text/HTML: 5,968 (updates/ news), at this point noted that pull investigates are only one request, if user requests are cumulative then received bytes are multiply with that request numbers.

The research has made known the following design of the mobile user and CWSs App Model identify as middle ware. Such as, Mobile User is capable to consume SOAP based and RESTful CWS as well over the CWS App Model, The mobile user can be realistic on diverse mobile clients platforms (Windows Phone, Android), the Mobile User can be fulfilled as a Web work app model as well as Native app model, JSON format workings more proficiently than CSV format in mobile background, CWSs App Model pushes to saves energy and bandwidth in mobile surroundings, The mobile user can able to implement mash up services from the CWS App Model, It is more in effect to create mash up with merge of one or supplementary functionalities on the middle ware than the systematic client-side, The CWSs app model can be hosted on GAE and on Heroku. To conclude, Projected CWSs App Model is establish to be reasonable for interrelate with mobile users to the cloud platforms / services.

## Acknowledgment

Researcher would like to express sincere gratefulness to all those who provided the prospect to complete this research work. Besides researcher would also alike to recognize with much appreciation the crucial role of the staff of computer science department, who provided the authorization to use all essential tools and the required materials' to complete the job "CWSs App Models". Distinctive thanks a lot go to my research guide, Dr. Atul Gonsai, who helps me to give suggestion around the task "CWSs App Models".

## References

[1] Dr. Atul M. Gonsai, and Mr. Rushi R. Raval, Analysis and Development of a Service mash up application for mobile users, International Journal of Computer Engineering and Technology, Volume 4, Issue 6, November - December 2013, pp. 262-268.
[2] Dr. Atul M. Gonsai, and Mr. Rushi R. Raval, Enhance the Interaction between Mobile Users and Web Services using Cloud Computing, Oriental Journal of Computer Science & Technology, Vol: 7, No: 3, December 2014, pp. 416-424.
[3] Dr. Atul Gonsai, and Mr. Rushi Raval, Mobile Cloud Computing: A Tool for Future, International Journal of Computer Science & Engineering Technology, Vol: 4, No: 7, July 2013, pp. 1084-1090.
[4] Mr. Rushi Raval, and Dr. Atul Gonsai, Performance Analysis and Design of a Mobile Web Services on Cloud Servers, International Journal of Emerging Technology and Advanced Engineering, Volume 5, Issue 9, September 2015, pp. 104-113.
[5] Apache JMeter, last retrieved from http://jmeter.apache.org/ (2015)
[6] Fiddler web debugger, last retrieved from http://www.telerik.com/fiddler/ (2015)