Implementation of enhanced generic REST API for cloud audio station

R.Prashanth*, V.N.Santhosh Ram, V.Srikrishna, E.Fenil
Department of Information Technology
Jeppiaar Engineering College
Chennai-600119, India

Abstract- **There has been a surging demand for the use of cloud based applications in this recent era. Cloud Computing is an emerging technology where a business can be started without upfront investments, storage becomes easy and more importantly cloud services are available on the fly elastically. Deploying applications on the cloud becomes cheaper. Hence, we propose a REST API which works as a middleware so that any device (Desktop / Laptop/ I pad / Mobile) can use this API and share files instantaneously. During our problem study we have noticed that some of the existing cloud providers do provide API to access their cloud but it works only with that particular cloud provider. To overcome this issue we intended to create a generic configurable REST API that will communicate with configured cloud without letting to know the client about this implementation details.**

Keywords: - REST API; Middleware; Cloud; Ruby on Rails

I. INTRODUCTION

Cloud computing is becoming part and parcel of every human life. Its power is more effective for humans to handle every tasks easily. It's purely based on services and its on-demand feature is handier for developers. Usage of the cloud is extremely increasing day by day. Application hosting has become cheaper and effective in cloud. Cloud providers have reduced per hour usage rates to per minute usage rates which is an encouraging factors for business people to reduce their costs. Cloud computing provides three different types of services. Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (Iaas). Among the three, Platform as a service (PAAS) is the eminent service provided by cloud. It provides the basic infrastructure and services that can be customized according to our needs. Moreover, it supports the business people to build applications which provide solutions for their business problems on the cloud. The applications that are target to build on cloud are extremely scalable, highly available, supports multiple tenants. Social Media are using huge volume of data on the cloud and our music station provides the user an ability to store music or any audio file on the cloud. The Music station cloud is aimed to serve for news agencies, private detective agencies and any other groups to share their audition / music / recordings on the cloud.

We utilize the PaaS in a more efficient manner by building an Application Program Interface between the cloud and the user/consumer. We follow the REST style in order to ensure light weight and high speed data transfer. Our aim is to introduce the layering approach of consuming cloud services with help of REST API as the bridge. Thus, we plan to build an adaptive API for the Sound Cloud which provides PAAS for the support of developers. This ensures maintenance of application for an audio station across different platforms in a customized way. We aim to provide much powerful functionality in our API by utilizing the merits of Ruby programming language on top of Rails frame work. Our API is highly pluggable, dynamic and scalable because of the Ruby on Rails framework. This API also ensures cross functional capability for the Sound Cloud and can be easily seated with any customized cloud built for Audio. We target the news agency, detective agencies and other people who want their audio files to be secretly held with private clouds.

The configuration of multiple cloud providers and client development is left for future enhancements. We have chosen Ruby on Rails as a technology to build this API as it is a open source, platform neutral and true rapid application development features. We have used sound cloud as our cloud provider who allows users to share music / audio files online to demonstrate our REST API concepts. We also wanted to make use of the sound cloud ruby sdk which is a built in support from the sound cloud provider. The sound cloud is the provider that maintains its cloud services in an enhanced manner. It provides PaaS to store audio files much efficiently. It is based on AMES framework and hence any API built on it ensures cross platform functionality. We integrate it with our API to provide effective audio service that can be maintained by any sort of client. Our API service

will ensure flexibility, cross platform capable and efficiency. Even an embedded system can easily communicate with our API to access cloud service. Thus we created a wide support and ensures a revolution in this field.

A. Existing System

Cloud computing which forms the integral part of every business is not much dynamic. The invincible features of cloud is not much discovered. The existing cloud services are only available as SaaS, thus making the cloud services more confined. It can be easily accessed with the help of middle wares. But the middle wares are not hybrid and light weight. The currently prevailing system is completely platform dependent and hence demands unique application for each platform. They are also not much scalable because of conventional technology.

While exposing the cloud services, there is greater chance of security breach. Some of the cloud are dependent on powerful authentication service which is not available in their conventional technology. The cloud where multimedia data are hosted provide only UI that is constant and unmodifiable.

The existing cloud services are only available as SaaS. SaaS is completely unalterable and it can't be customized according to our needs. For instance, there is no point in providing genre classification of audio files for news agency. Middleware we built ensures high level abstraction which is completely lacking in this conventional SaaS. The SaaS provided can't be consumed by any API or middleware. It requires idiotic techniques like web scribing which is extremely vulnerable. SaaS is a software with no alteration and one must consume it without any changes. This hinders wide support and functionality across different devices. SaaS is more platform centric than device centric. This approach is not layered and is more difficult to protect it from security breaches. Even a small flaw can cause a complete exposure of cloud to vulnerabilities. There is only static form of technique that stores data in cloud. Lack of efficient API is the reason for such kind of defects. Most of the API are not scalable because of conventional languages like JAVA. In these languages everything exist as classes and demands object for each functionality that hinders certain functionalities from reusability. Thus, those middleware are far from scalability. We aim at removing these defects through our proposed work.

B. Field Study

There is a huge demand for API based approach in the field of multimedia as it devoid many platform approach. We chose the audio from multimedia as it has both entertainment and social cause aspect. In the field journalism, there is a huge demand for audio application with cloud support. They can be effectively used for providing news at a faster rate than video. They can also effectively used to provide social problem to our perspective with a valid proof. They can also expose social and political criminals with good and effective proof. They can act as an eye opener for the public. Audio is easily consumed by news agency and it can be captured easily under any circumstances as less bandwidth is required. Our direction towards this REST API when the demand for dedicated embedded device with security features aroused. The REST API can be contacted by XML and responds with an XML that can be effectively used from small to large device without any platform dependency. Moreover our API is aimed at multi API consumption feature with security that can promote on the fly upload of recorded sound to cloud. Thus the evidence is easily uploaded to cloud with anonymity support and can't be destroyed as it happens today with devices. Thus our product provides much pronounced features for detective agency, news agency and multimedia lovers.

C. Proposed Work

Based on our above field study and analysis existing system, we propose an API approach using REST architecture style for effective communication with cloud to handle audio data. Our proposal is a lucid approach that will create more enhancement and flexibility in the field of multimedia with cloud computing. In our approach we borrow the cloud platform service from the leading sound cloud which has a much adaptive and bandwidth saving architecture for audio management. The platform can be effectively accessed through its SDK. By consuming its SDK, we build a more effective REST API using the powerful RAILS framework. Hence we utilize the ruby SDK of Sound Cloud which is the perfect language to run RAILS on top of it. Our API uses HTTP verbs to communicate the Sound Cloud platform. The ruby SDK provided by Sound Cloud provides the end points of sound cloud which can be consumed for effective communication. Thus a layered approach with enhanced security is achieved. This API can in turn be consumed by any platform and is more device centric. It uses XML/JSON format for communication. This API poses no threats and can't be spoofed

because of powerful security provided by RAILS framework. The authentication support is much effectively provided by RAILS framework. This API has a great durability and is completely agile. It can be clubbed with two or more APIs to widen its functionality. It can be consumed by another API. It is completely pluggable so that we can change the cloud provider at any time without the consumer interaction. Since the communication is XML/JSON based, it is completely cross platform consumable one. Thus, we aim at introducing more refined layering approach in the field of cloud computing by unleashing its PaaS powers.

As in Fig.1, our high level architecture has three important layers as shown below

1. Sound Cloud with RUBY SDK(providing PaaS)
2. Our REST API
3. A Test UI (any platform UI or any device)

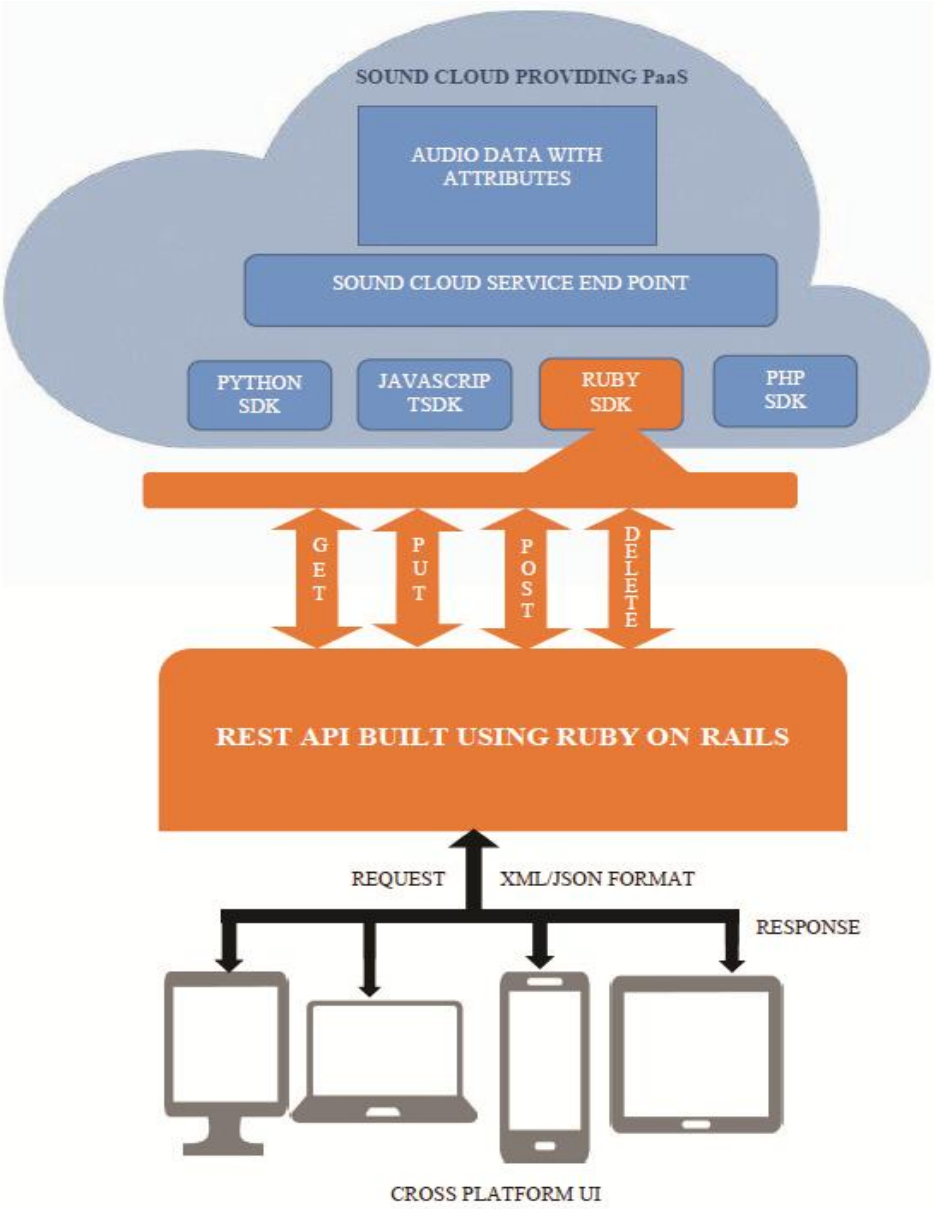

Figure 1. High level Architecture

II. ARCHITECTURE

A. REST API:

In our paper, the REST API is the adaptive music cloud which is used to communicate with the sound cloud. The music cloud API is a middleware, a service that communicates with Sound Cloud provider and facilitates the basic CRUD operations to make the communication. This service follows a REST (Representational State Transfer) architecture style so that it provides a unique interface for any type of client. Music Cloud API can be consumed by any type of client applications such as mobile, web & console application and also another API or service that can make use of the Sound Cloud services. This API can be easily consumed by the client and can share audio. Using REST helps us to make point to point communication very easily.

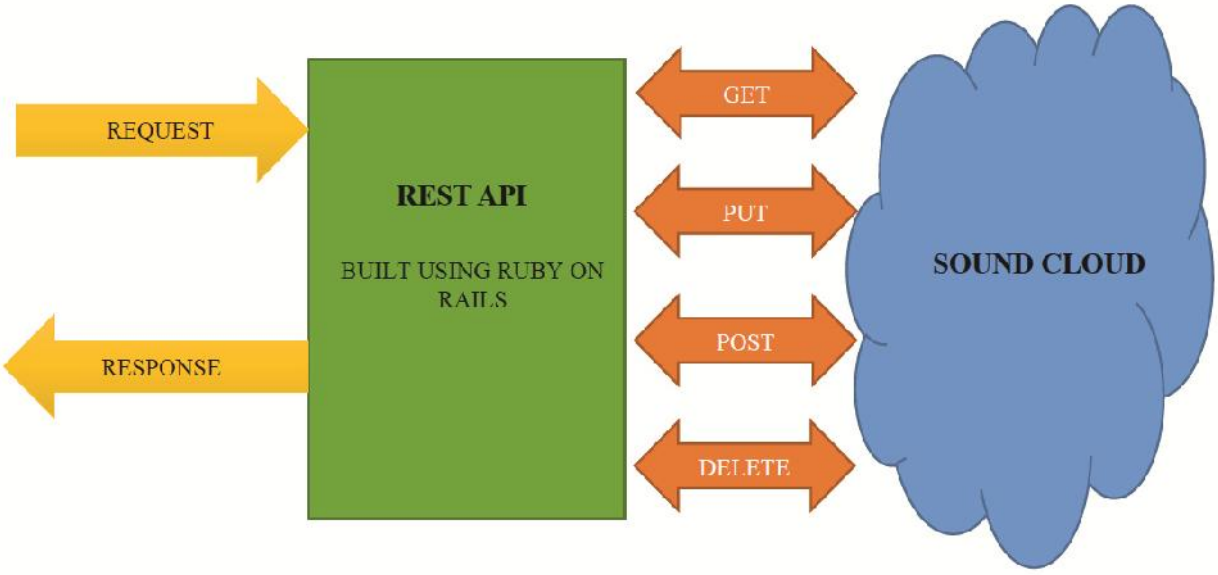

Figure 2. REST API

One of the main primacy of the REST is that it is light weight when compared to the SOAP services. By using REST, we can send and receive data as JSON, XML or even plain text. The REST API mainly constitutes the HTTP verbs which forms a considerable portion of our “uniform interface” constraint and provide us the action counterpart to the cloud-based resource. As in Fig.2, the most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilised very rarely. Of those rarely used methods, OPTIONS and HEAD are used more often than others.

Web Service APIs that adhere to the REST constraints or conditions are called Restful. Restful APIs are defined with these specifications: base URI, an Internet media type such as JSON, standard HTTP methods and hypertext links to reference state resources. We have built our REST API using Ruby on Rails which follows Model View Controller pattern. The model comprises of the data pertaining to a business logic and they can fetch the data from the database. Rails derive its model classes from ActiveRecord::Base. The view comprises of the User Interface that are usually coded with HTML. The view should be simple and pleasant in order to attract the users. The controller plays the vital role here. It gets the incoming Http requests and gives the corresponding http response. When the user gives some requests, the request is directed to the controller. The controller then fetches the data from the model and it then displays the output in the view.

The Rails API is a part of the Rails application which is used extensively in API applications and it is one of the gems provided by rails. It provide only the functionalities needed for the API and thus it is little light weight compared to the normal Rails application. The Rails API primarily uses JSON because this format can be easily decoded by rails. Rails API provides good security against spoofing attacks and timing attacks. Many third party plugins are supported by the Rails which comes in handy in most of the situations. An important

advantage of using Rails API is that it makes use of Active Record which provides both entity framework and Object Relational Mapping (ORM) framework in Rails. The ORM framework is very important as it maps the tables in the database with the objects in the application.

B. SOUND CLOUD:

Sound Cloud is one of the leading cloud provider that provides SaaS and PaaS for audio files. They paved the path for other developers who can utilize its PaaS for designing some sophisticated applications for audio file. In our approach, we utilize it as a repository for audio data and other supporting data.

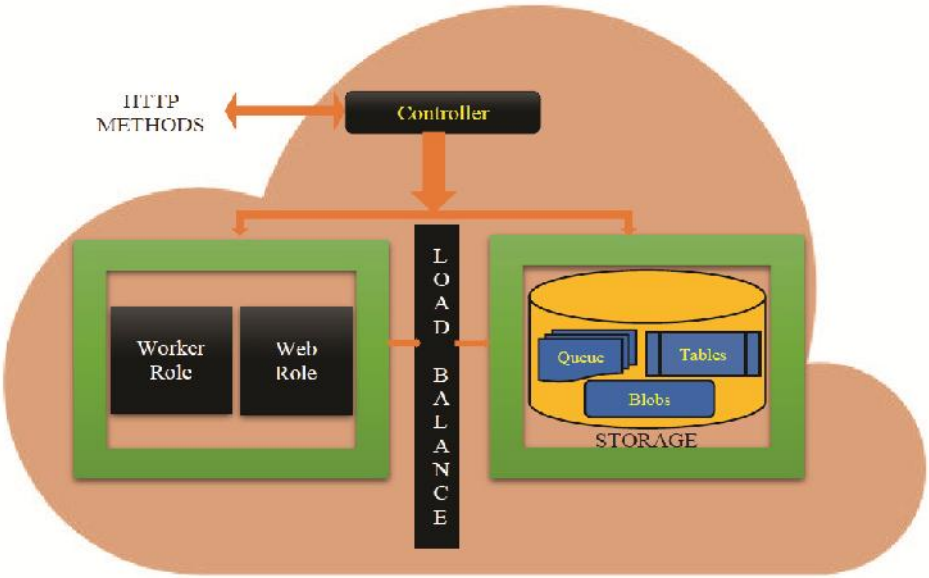

Figure 3. Sound Cloud Architecture

Sound Cloud is much more adaptive and is following AMES Framework (Adaptive Mobile video streaming and Efficient Sharing). Thus its architecture is more enhanced to suit different devices. Sound Cloud's SaaS is more famous for entertainment. But it has much more features that can be more useful for entertainment as well as social cause. Hence, we utilize its PaaS to communicate with our REST API. Sound Cloud gives free support for developers and just require a free subscription for utilizing its free platform.

The cloud has a strange, efficient and enhanced architecture. The audio data are stored in blobs with its attributes mapped in corresponding database. The cloud has a load balancer that balances and manages everything according to the load. The cloud has a parent controller that adjusts components according to needs. It schedules the main processors, allocates right caches and efficiently manages data flow. The cloud has a worker role that configures the storage area. It also has web role which is meant for network communication. Thus the cloud is shaped in a better manner than conventional server for effective data handling.

Sound Cloud provides SDKs supporting different languages. It has support for Ruby, Python, JavaScript and PHP. It allows only two languages for effective communication since both of the languages have pronounced security support. They are Ruby and Python. We chose Ruby for building our API since it integrates best with Rails framework for API development. As we described earlier, Rails-API is much more flexible and support the lightweight REST architecture using JSON/XML. Thus the Ruby SDK comes handy for our development.

We utilize the Sound Cloud as a model according to our MVC approach for our API development. As we discussed, we utilize the platform of the cloud. Since it provides PaaS, the required software are installed. Its rules are already designated that are exposed in those SDKs. The SDKs act as a gateway for communication. They are readymade end points of Sound Cloud. Our communication begins with registration to access those end-points. We use the network component of the cloud for communication.

The Sound Cloud's service end point can be much easily accessed by HTTP methods combined with security. The four important methods namely PUT, POST, DELETE and GET are useful for communicating with Sound Cloud. Moreover the Ruby SDK provides better authentication feature using "OAUTH" technology. Using the APP registration feature of Sound Cloud and this technology better security is provided which is described in our authentication module description. Thus the "OAUTH" mechanism act as a key to Sound Cloud services.

Sound Cloud exposes its features using HTTP endpoints. They involve:

1. Upload – End Point '/tracks' using post method
2. Download/Stream – End Point '/tracks/download_url' using get method
3. Search – End Point '/tracks' using get method
4. Follow – End Point '/users/followings'
5. Like – End Point '/users/favorites'
6. Comment – End Point '/comment'

It provides an elegant approach of accessing its repositories using HTTP verbs powered by OAUTH mechanism. Without this OAUTH mechanism nothing can be communicated with Sound Cloud. Thus it exposes its features with enhanced security mechanism.

C. USER INTERFACE:

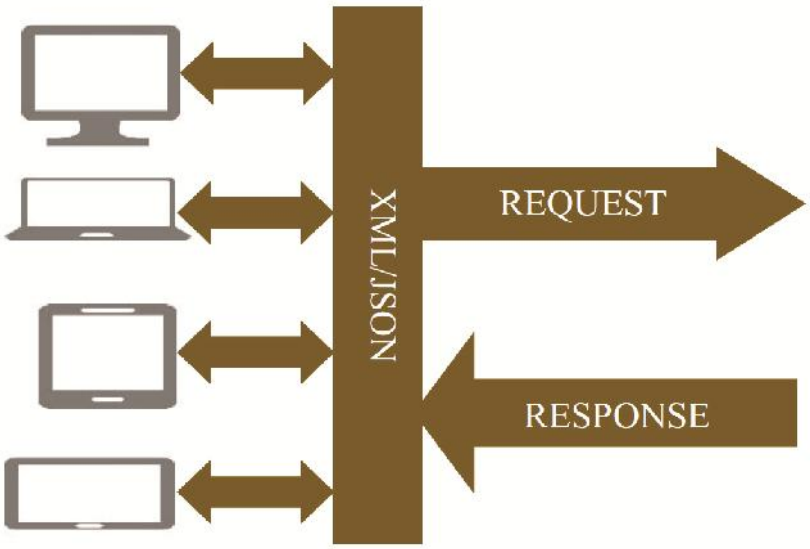

Figure 4. Cross Platform UI

The basic definition of user interface in computer language is the ability of the users to interact with the computer, tablet, laptop or a Smartphone in this high prolific world. The user interface comprises of list, menus, command-line argument, gesture recognition movement, command language and online help. Although user interface is made of fraught errors such as voice recognition and gesture recognition laughable actions to provide a user friendly environment. Our main aim is to provide such typical user interface which is capable of accessing all the devices at one go. This aims in providing much useful better interface facility and have an added advantage to provide a better friendly environment to the user from respective devices and get an improver response from the devices end which is automatically generated process. All these interfaces can be designed in different programming languages. They can be easily communicated using REST API with XML/JSON. All the UIs have consumption point where the address of the API is mentioned to consume them more effectively. We have just developed a test UI to verify our API integration. Our aim is to make different developers to develop more sophisticated and device centric UI for effective communication. Thus we feed for some thoughts to renowned UI developers for developing customized User Interface more suitable for physically challenged and mentally retard people. They can also enjoy this boon of music through our effective API.

III. MODULE DESCRIPTION

Our paper involves six important modules which are the sophisticated features of Sound Cloud. They are as follows:

A. Authentication
B. Upload
C. Download/Stream
D. Like and Comment
E. Follow
F. Search

A. Authentication

Authentication is the main module of our paper. It provides the much more security to clients and is much efficient using RAILS framework. Authentication is achieved much effectively through OAUTH technique of RAILS. It involves registration followed by token exchanges. In order to communicate with Sound Cloud platform we require application/API registration. On creating our application in Sound Cloud, we get two secret credentials namely,

1. CLIENT ID
2. CLIENT PASSWORD

These two along with a registered user id and password forms the OAUTH credentials. It is communicated to sound cloud to get our tokens. These tokens are randomly generated and it is indirectly connected with our two primary credentials. These tokens are short lived and thus prevents spoofing attack and masquerading.

Without this access token no service can be accessed from Sound Cloud End-Point. We use the "Sound Cloud gem" file in our rails API to create object for accessing sound cloud services. A separate controller is created to promote reusability. The access object is created using that controller. These security credentials are the ones required to create the access object that allows to access Sound Cloud services. This ensures high level security abstraction and provide more secured functionality.

```ruby
def self.soundcloud_client(options={})

  options = {
    :client_id     => SOUNDCLOUD_CLIENT_ID,
    :client_secret => SOUNDCLOUD_CLIENT_SECRET,
    :username => "#{t}" ,
    :password => "#{b}"
  }.merge(options)

  Soundcloud.new(options)
end
```

```ruby
def connect
  #redirect_to soundcloud_connected_path
end
def connected
  User.t = params[:user_name]
  User.b = params[:pass_word]
  if params[:error].nil?
    soundcloud_client.exchange_token(:code => params[:code])
    me = soundcloud_client.get("/me")

    login_as User.find_or_create_by_soundcloud_user_id({
      :soundcloud_user_id  => me.id,
      :soundcloud_username => me.username
    })
    current_user.update_attributes!({
      :soundcloud_access_token  => soundcloud_client.access_token,
      :soundcloud_refresh_token => soundcloud_client.refresh_token,
      :soundcloud_expires_at    => soundcloud_client.expires_at,
    })
    redirect_to root_path
  else
    render :layout => false
  end
end
```

Figure 5. RUBY CODING IN RAILS FRAMEWORK:
Creation of access object and receiving tokens in connect action

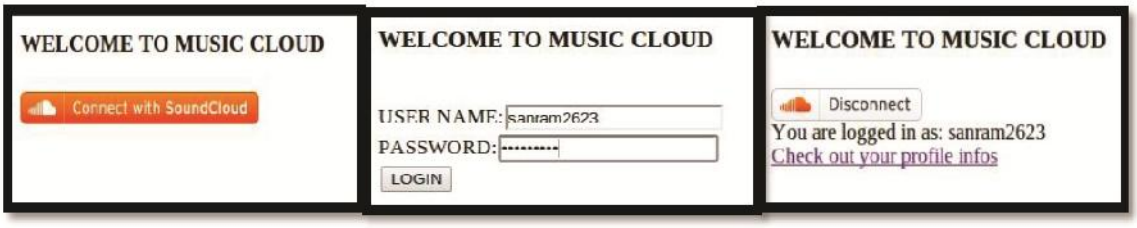

Figure 6. Sample output for authentication in browser by consuming our API

B. Upload

Uploading includes uploading audio files on the sound cloud using the REST API. Our API provides the option of upload and manage the audio files efficiently. Using this features, we can upload any audio files instantly and can make available on the sound cloud. We can upload the files with the Rails API by using the post method. We create a separate controller to control this activity in our API. Using the access object the post method is called with necessary end-point along with the path of our audio file obtained in UI as asset attribute.

Thus on successful upload our track can be accessed on the fly anywhere anytime. We can also provide on the fly recording by clubbing recording functionality with this controller.

C. Download/Stream

The terminology download is used to refer to getting the files from the cloud to our own space storage systems such as hard disk, memory card and other primitive devices used to store files. These files are said to be accessed from their personal devices without actually connecting over to internet every time. We provide the option download which is very flexible without affecting the actual process time on the cloud. Thus having enhanced way to show the path of downloading files to the system and helping them rejoicing the time spent on our cloud. It is completely authenticated by using access object created and can prevent personal data loss. This also involves separate controller with stream and download are separated as unique actions. Both uses separate get requests using the access object. Thus, user can get two features depends on his need and deed.

D. Follow

Sharing of audio files through internet was always a hectic task as there was always a leakage of sound files and other important files transferred over internet though these ages. As our project scope is to develop the unique methodology to prevent misuse of audio files being used by the interceptor. This provided us to build a private cloud where the audio files are said to be uploaded in the private cloud and unauthorized members cannot get access to it. Our aim is to provide a user to share any format of audio files and from anywhere on this planet in order to get the user to share audio files on the fly. The follow functionality comes handy for this feature to get successfully done within our scope. Follow feature is accessed by creating a separate controller to ensure loosely coupling. Using the access object, the user to be followed can be made as followings using the post method. This is more secure since the user id of user is required and the user can also cancel his followers if he found him anonymous. Thus through this simple and lucid task unwanted complexity is avoided.

E. Like and Comment

These are two features for multimedia lovers. These features are handier for users to keep personal favourites of him and his followers. This feature is more effective for music lovers. The comment involves timeline feature which involves direct player integration. Here user can give comments on his favourite second of his song. Both the feature are created as separate actions in a single controller as they don't need any further abstraction. They too involve access object by calling the post with '/favorites' and '/comment' as end points.

F. Search

This feature is directed towards multimedia lovers as well as social people. Here we have genre based and user based search. User search is directed towards social people and genre based search involves searching best tracks of 15 customized genres available in this Sound Cloud. The programming part of this feature involves '/tracks' end-point with two different get requests in two different actions namely, 'search by user' and 'search by genre' inside the search controller. You can limit the number of tracks to be displayed and can get tracks in a customized way more easily and securely.

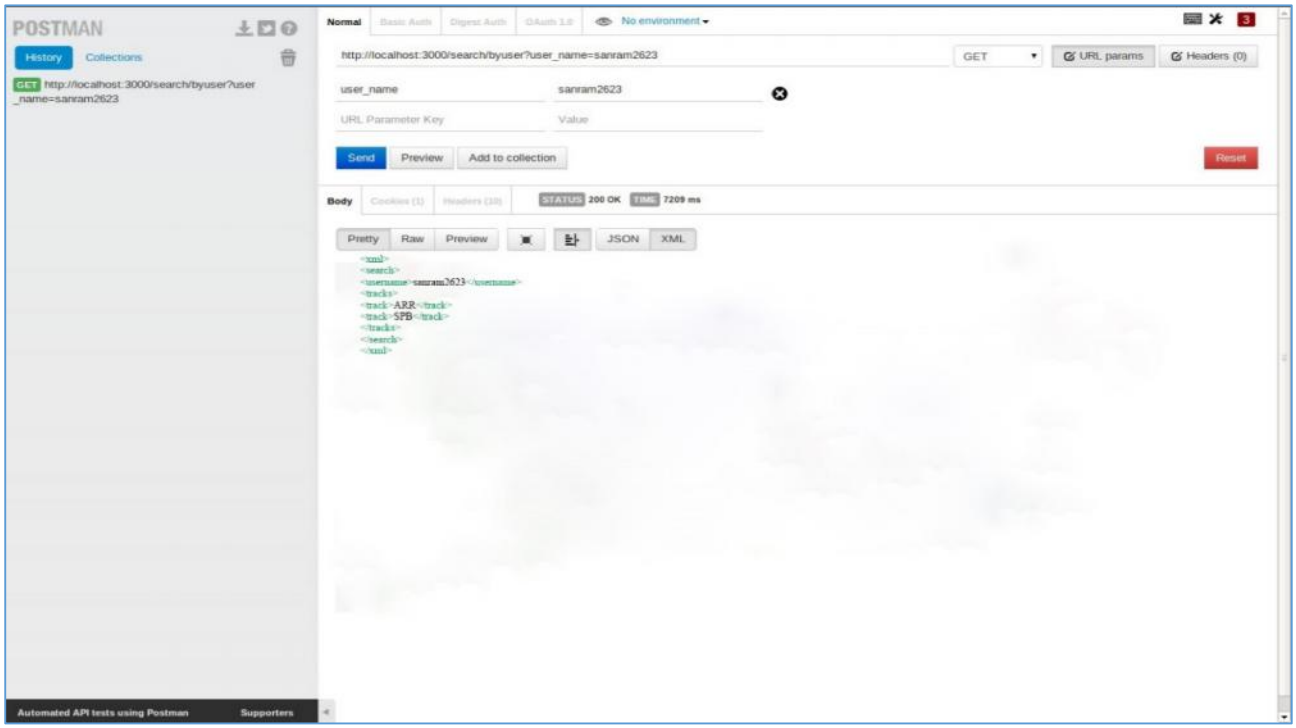

Figure7. XML output of search module in a REST client (Postman)

IV. CONCLUSION

Thus, by creating a REST API, we have paved way for the future developers to develop their own APIs for the cloud and has made the API more device centric. The REST API was useful in creating viable User friendly and customizable interfaces, thus enabling the people to have easy access to the cloud to serve their purpose. By using REST API instead of the traditional SOAP, we have proved that REST is more dynamic and powerful for communication with the cloud. Developing the REST API with ruby on top of rails has made the API more scalable and has enhanced the security mechanisms. We have even used the PaaS model of the cloud computing more effectively. Finally, the purpose of accessing the service using cloud has been served well by Sound cloud which provides different SDKs to build API to support developers of various platforms.

REFERENCES

[1] Qi Zhang, Lu Cheng, Raouf Boutaba ”Cloud computing: state-of-the-art and research challenges”

[2] Guidelines for Implementation of REST - Enterprise Applications Division of the Systems and Network Analysis Centre (SNAC) Information Assurance Directorate, National Security Agency

[3] Xiaofei Wang, MinChen“AMES-Cloud: A Framework of Adaptive Mobile Video Streaming and Efficient Social Video Sharing in the Clouds”

[4] Ming Feng Tan1, Xiao Su1 and Haiyan Wu "Media Cloud: When Media revolution meets Rise of Cloud Computing"

[5] Yu Wu *†, Zhizhong Zhang†, Chuan Wu†, Zongpeng Li‡, Francis C.M. Lau “CloudMoV: Cloud-based Mobile Social TV”.

[6] http://developers.soundcloud.com/docs/api/reference

[7] http://rest.elkstein.org/2008/02/how-simple-is-rest.html

[8] http://www.api.rubyonrails.org

[9] http://en.wikipedia.org/wiki/Cloud_computing