Implementation of Cloud based Application for Smart Cities

Pratibha Tiwari¹, Apurva Krishnappanavar², Neeraj Deshpande³, Siddharth Dongre⁴, M.R. Mahajan⁵

BE Student¹, ², ³, ⁴, Assistant Professor⁵

Department of Computer

PVG’s College of Engineering & Technology, Savitribai Phule Pune University, Pune, India

Abstract:
This paper describes a cloud based application for citizens of Metros and other metropolitan cities. User can get navigated to the fastest route. User can search; add nearby places such as Bus stop, Post office, parking area. The application will have normal question answer forum; we can think about giving some ratings to every used under this section. iReport is to raise a complaint to PMC or electricity board or police, officials etc. through our image uploading interface. In this implementation paper we focus on the main drawback of cloud computing i.e. security for which we have drafted our own light weight encryption algorithm.

I. INTRODUCTION

The implementation of an application which is a basic need for people living in smart cities. When we talk about smart cities it deals with the time utilization of the citizen of the city smartly this app is a step towards that implementation. In current scenario where there are thousands of applications available in the market which confuse the user by its tricky design, this app serves all-in-one solving the complexity. With the help of this app we can directly navigate to our destination. There is no blog were the common citizen can put up there questions or problems faced by the new comer’s in the city, here we have citizen forum for citizens to interact effectively with each other. There is no means through which we can get register our complaints to the government and get to know about its status but through this app we can raise our complaints easily. Most of the apps are bulky in size which leads to citizens to be reluctant for its installation of the app but we are designing in such a way that it should not exceed more than 5 MB. The greatest problem of cloud computing is security. Cloud computing has happened in the last decade to revolutionize the world of computing. Chronologically, there was the appearance in the 80's, virtualization and outsourcing; Also, in the 90s, the democratization of IT especially during the last decade, with the generalization of the Internet, the development of broadband networks, rental application, payment for the use and the quest for mobility. This gave birth to the new concept of infrastructure and IT solutions called Cloud Computing. Indeed, the cloud consists of outsourcing IT infrastructure to specialized providers. Users of cloud computing are gaining autonomy, ergonomics and simplicity. Also, the composition of applications on the "cloud" will open opportunities for collaborative work still unpredictable. [1] We have used AWS Amazon Web Server as our cloud server. This paper includes the implementation of light weight encryption and decryption algorithm. All public key cryptosystems, though being highly secure, have a common drawback: They require heavy computational effort, which is the advantage of our algorithm it is light weight, very less computation is required and at the same time it provides with full proof security. All these modules are blended together in our application to provide citizen a sleek and efficient highly functional with cloud servers.

II. SYSTEM REQUIREMENT

2.1 User Interface
The application will be running on an Android OS 4.0 or higher. Its user interface will provide all the features associated with the four modules. The features will be seamlessly integrated into the application in a way that the user won't have to select every feature manually; they will be active all the time as background processes. The user will be given a Notification panel for making posts and status updates. The user will be given a reporting panel for reporting complaints.

2.2 Hardware Interfaces
The application will require a server for storing user information along with other data gathered for processing by the server. For this purpose we will be using a server provided by Amazon Web Services which will be running an EC2 instance. It will also require a smart phone with GPS functionality and supported network type. Detailed of an in-text quote, but before the final punctuation functionality and supported network type. Detailed requirements are:

- **AWS Requirements:**
  - 2.5 GHz Intel Xeon Processor
  - 1 GB RAM
  - 8 GB General Purpose SSD

- **Mobile Requirements**
  - 1GHz single-core processor
  - 1GB RAM
  - GSM / HSPA / LTE Network Type

2.3 Database Requirements
The user application gathers an immense amount of data from the user. This data needs to be organized in an easily accessible format. For this purpose a MySQL Database Server is used for storing all the user data. The data is stored in JSON format.
III. SYSTEM IMPLEMENTATION

The two major pillars of our system is specified as follows:

- User Interface : Android App
- Processing System : Aws Server

User Interface: Android app

Android app is the interface through which user can access the functionality in order to avail the access to our app user initially need to create an account providing its credentials as shown in figure (1). User need to specify name, email id and mobile number to get started.

Next the interfaces of our four main modules are as follow:

1. Traffic Navigation and Nearby Places:
Here the markets with different color show different locations, where red color marker shows user location as shown in figure (2). If we click on the nearby places we will be navigated to the nearest hospital garage so that user does not face the problem of explicitly specifying the location.

2. Citizen Forum:
In this module user can post up there questions and others can comment upon it. User basically can interact with each other actively to solve the problems of new comer’s to the city as shown in figure (3). This particular module leads the implementation of encryption and decryption algorithm where the data to be stored in cloud is encrypted.

3. IReport (Complaint Forum):
This module deals with the complaint to be registered with the PMC where the user needs to fill a form as shown in figure(4). User needs to select the department and enter complaint title, details and location with further extra information.

Processing System: Aws Server

MobaXterm is the Remote Login Console we normally use to connect to the AWS terminal. You can also use similar software
like Apache Putty to do the same, we just prefer this because it’s easier to use. Once you install MobaXterm you will see the screen as shown in the figure (5)

![MobaXterm Software](image)

Figure 5. MobaXterm Software

Now with the help of SSH command we can connect to AWS Server, we need to start a local terminal and window will open up and now we need to write command to connect with our AWS Server. Run the Command:

```
ssh -i "SD_AWS_Sep2016.pem" ubuntu@52.220.137.50
```

If it works, we will be successfully logged into the AWS remote terminal. From here we can execute any Linux command, as long as it doesn’t involve installing new software or compiling new programs. To do that you need to add “sudo” before every command.

IV. ALGORITHMS

We have used cloud data storage in our application where the backend is AWS cloud server. As the data is stored in public cloud the security and privacy can’t be guaranteed.

A. Origin of cloud computing

This study offers a business prototypical for cloud computing based on the perception of using a discrete encryption and decryption service. In this typical, storing of data and decryption of user data is provided distinctly by two distinctive benefactors. In addition, those occupied by the data storage system will have no right of entry to decrypted user data, and those functioning with user data encryption and decryption will remove all encrypted and decrypted user data after transmitting the encrypted data to the system[2].

B. Secured data storage

In order to ensure security we have drafted our own encryption and decryption algorithm. The flow of Encryption and Decryption algorithm at the time of storage is shown in the figure (6), where the data to be stored in database is initially encrypted in android app and then transferred over the network so that if someone even try to intercept the data could not understand the actual content in it and then the data is decrypted and stored in database.[3]

![Data from App to Server](image)

Figure 6. Data from App to Server

The flow of Encryption and Decryption algorithm at the time of extracting the data and display to user is shown in the figure (7), where the data to be displayed in android app is initially encrypted by AWS Server and then transferred over the network so that if someone even try to intercept the data could not understand the actual content in it and then when data reach the android app it is decrypted and displayed.

![Data Flow from Server to App](image)

Figure 7. Data Flow from Server to App

The Algorithm is as follows:

Encryption Algorithm:

```java
char key[] = "1hello";
char key1[] = "Baseiscomp";
res=0;
for(int i=0;i<7;i++)
temp = ASCII(key[i]) + ASCII(key1[i])
if(temp<128)
    store[i] = char(temp);
el
```n
Decryption Algorithm:

```java
for(int i=0;i<7;i++)
temp1 = ASCII(store[i]) - ASCII(store[i])
if(temp1>0)
decryp[i] = char(temp1);
el
```
temp1 = temp1+128;
decryp[i] = char(temp1);
Hence we implement the light weight encryption algorithm in our implementation.

Haversine Algorithm:
We will be using Haversine’s algorithm for searching and displaying the nearby places.

Haversine formula is an important in Navigation. It gives great circle distances between two points on a sphere from their latitudes and longitudes. For any two points on a sphere, the haversine of the central angle between them is given by

\[ \text{hav}(d/r) = h = \text{hav}(\phi 2 - \phi 1) + \cos \text{hav}(\phi 1) \cos \text{hav} (\phi 2) \cdot \text{hav}(\lambda 2 - \lambda 1) \]

where \( \text{hav} \) is the have sine function:

\[ \text{hav}(\theta) = \sin^{2}(\theta/2) = 1 - \left(\cos(\theta)\right)/2 \]

\[ d = r \cdot \text{hav}^{-1}(h) = 2r \sin^{-1}(\sqrt{h}) \]

In our algorithm, we define a specific distance for e.g. (1 KM). Then we will compare the distance found by Haversine’s algorithm with our specified distance. If the distance is less than the distance specified by us, this particular place will be displayed else it will not get displayed.

Algorithm:
#define R 6471
#define D 1
#define To_Rad(3.14/180)

Double dist (double lat1, double long1, double lat2, double long2):

double dx, dy, dz;
lat1 -> To_Rad;
lat2 -> To_Rad;
long1 -> To_Rad;
long2 -> To_Rad;
dx = (sin^2(lat2-lat1)/2);
dy = cos(lat2) * cos(lat1);
dz = (sin^2(long2-long1)/2);
return sin^(-1)(sqrt(dx+dy+dz)) * 2*R;
d = dist(36.65, -85.54, 33.94, -112.67);
if(d<=D):
    accept;
else:
    Reject;

V. CONCLUSION

This Implementation paper helps us to solve the problems faced by the citizens of big cities where population, traffic, searching places, lodging complaints, forum for interaction, etc. is actually a problem. We thus create an efficient crowd sourced app which will solve these problems to a great extent. We have used a light weight encryption algorithm to provide security in cloud server. We have thus stored all information of citizens as well as the locations in the database using MongoDB. We have also built our servers in the cloud by using Amazon web services cloud platform to make use of cloud computing and cloud storage.