A RESTful E-Governance Application Framework for People Identity Verification in Cloud

Ahmedur Rahman Shovon, Shanto Roy, Tanusree Sharma, and Md Whaiduzzaman

Institute of Information Technology,
Jahangirnagar University, Bangladesh,
shovon.sylhet@gmail.com, sroy.iitju@gmail.com,
tanusreesharma207@gmail.com, wzaman@juniv.edu

Abstract. An effective application framework design for e-governance is definitely a challenging task. The majority of the prior research has focused on designing e-governance architecture where people identity verification takes long time using manual verification system. We develop an efficient application framework that verifies peoples identity. It provides cloud based REST API using deep learning based recognition approach and stores face meta data in neural networks for rapid facial recognition. After each successful identity verification, we store the facial data in the neural network if there is a match between 80-95%. This decreases the error rate in each iteration and enhance the network. Finally, our system is compared with the existing system on the basis of CPU utilization, error rate and cost metrics to show the novelty of this framework. We implement and evaluate our proposed framework which allows any organization and institute to verify people identity in a reliable and secure manner.

Keywords: E-Governance, Cloud Computing, RESTful API, ID verification

1 Introduction

E-governance applications are based on Government to Government (G2G), Government to Business (G2B), Government to Citizens (G2C) and Government to Enterprise (G2E) [1]. It is nothing but a service that act as a medium of exchange of information , transactions and services. Cloud Computing is the delivery of computing services servers, storage, databases, networking, software, analytics and more over the Internet.

A number of issues have risen in e-governance application. The lack of appropriate framework, content development, citizen access are the main problems. Every government needs to record the data of its citizens including national ID (NID) information and photo which are not updated regularly. In the mean time a person's look changes with his age. Thus verification can be problematic in that

case. Here, updating photo is necessary which might be costly and requires more space that remains idle mostly. Again large amount of data can make chaos and redundancy. For that an effective E-Governance system should be cost effective, reliable and easy to maintain.Unfortunately,current technologies are not enough to meet the overall requirements of E-Governance [1].

To overcome the scenario, Cloud technology or mobile computing [2] is the solution to the problems. E-governance is facing failure in meeting requirements of public services, high IT budgets compared to low utilization of existing resources, poor security mechanisms, costly operations and maintenance (O & M) etc [3]. Hence cloud based E-Governance application framework provides many benefits in particular to government. It provides a comparatively better platform for efficient deployment of E-governance System [1] than the traditional manual processes.

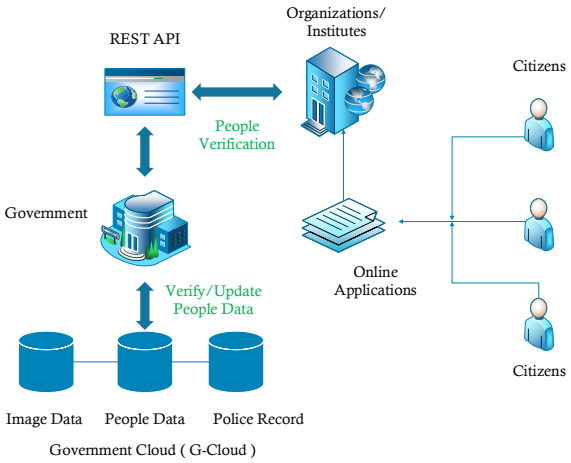

Fig. 1. Block Diagram of the proposed system

In this paper, we implemented a smart approach that the government can keep only the vector co-ordination information in cloud for further verification. By adding images of a unique face and updating it with AWS AI approaches decrease the error rate of people identity verification. The feature of deep learning based image recognition technique make the system efficient for authorized access. The increase of facial data of the citizens will open the door of future study. Our system provides better CPU utilization, scope for migration to new technologies.Here, we used police verification data which eliminate manual verification process and reduce the delay. Moreover, REST approach confirms the instant response from verification framework that can be integrated to any authorized application. Overall, the primary contributions of this work are a people

verification framework, utilization of cloud for E-Governance application, RESTful approach for better and secure service provision. For better understanding, the block diagram of the proposed system is showed on Figure 1.

The remainder of the paper is organized as follows. Section 2 describes different E-governance system proposed by several researchers. Section 3 describes our proposed methodology. Section 4 presents the architecture of the framework, Section 5 shows the results. Finally, Section 6 presents the conclusion and future work of this study.

2 RELATED WORK AND MOTIVATION

In this section, we discuss several existing E-governance system and also describe our motivation for this study.

2.1 Related Work

Several researchers have developed and surveyed some system for E-governance. For example Witarsyah, Deden, et al in [3] studied on adding some important variables: trust, information quality, system quality of e-government adoption in improving service to the citizen. In [4] Rao et al. proposed a system comparing with existing system on the basis of browsing performance, response time, and security to show that novelty nature of the software architecture to support e-governance of India on highly distributed evaluation environment. Hence, information quality and data scaling are ignored here. In [5] Zissis et al. proposed a high level electronic governance and electronic voting solution, supported by cloud computing architecture and cryptographic technologies. It is quite clear from this paper that there is a large scope for further research.

WenJun Zhang et al in [6] introduced a new concept of C-government that utilizes cloud computing in e-governance. Authors designed the architecture and discussed about the implementation and deployment of the c-government system. The paper also explained how citizen engagement is maximized with enhanced collaboration towards government productivity and service delivery. Sajjad Hashemi et al. In [7], authors discussed about the benefits of using cloud computation in e-governance including improvement and optimization providing public services, governments enhanced ability to interact and collaborate, achieving transparency and accountability as well as overall improvement of the relationship between the government and its citizens etc. How developed countries like Singapore, USA and UK are adapting to the cloud technology is also mentioned here. In [9] Wojeich et al. researched on two levels of aggregation: organizational units and functional modules. Some mentionable advantages of cloud are: dynamic load of resources, proper maintenance and administration, higher performance etc. Authors also claims that REST based web technology is gaining a lot of attractions.

Face verification across ages has importances in some potential applications such as passport photo verification, image retrieval and post processing, surveillance etc [10]. As human faces change in course of time and still the face image

may need some verification. Photo identification means the matching accuracy in percentage using 3D modeling [11], discriminative methods [10], feature extraction and modeling through deep learning etc [12] [13] [14].

In [15] Christensen et al.implies to the integration of REST-based web service and cloud computing in order to create futuristic web and mobile applications along with optimization and security. As REST architecture ensures the data security as well as maintains a secure communication [16]; that is the reason behind choosing this technology in our work.

2.2 Motivation

From our observation, as discussed in Subsection A of Section II we can come to a conclusion that we needed to develop an framework that includes digitization, proper citizen access, increase facial data of each citizen, lowest cost, lower error rate of face recognition. To get the best performance of E-governance system we need to focus on this trade-off. Therefore, in our proposed policy we take a more holistic approach by considering a RESTful API can be accessed from any device that is connected with internet.

3 Methodology

We designed the architecture to verify people identity and personal information in G-cloud using the secure RESTful API. The process starts with searching information in the database using NID; matches the facial vector data with the stored data mapped with the NID. The system flowchart and algorithm are shown in Figure 2 and Algorithm 1 accordingly.

4 Architecture of the framework

Our proposed system developed an E-Governance application framework which includes hardware and software requirements and it uses cloud based artificial neural network.

4.1 Structural Description

The proposed framework uses the Amazon Web Service (AWS) for all operations. The REST API is hosted in EC2 instance and can be accessible via API calls from any other systems. The REST API is built using Python Flask Framework and it abides by the prescribed standard of HTTP status codes. JSON Web Token (JWT) which is a JSON based open standard for creating access tokens is used to authenticate the API calls. For storing the newly compared faces, the framework uses Rekognition and Elastic Load Balancer (ELB) from Amazon Web Services. For storing the public people data it utilizes the functionalities of SQLAlchemy, Python SQL Toolkit and Object Relational Mapper (ORM) that

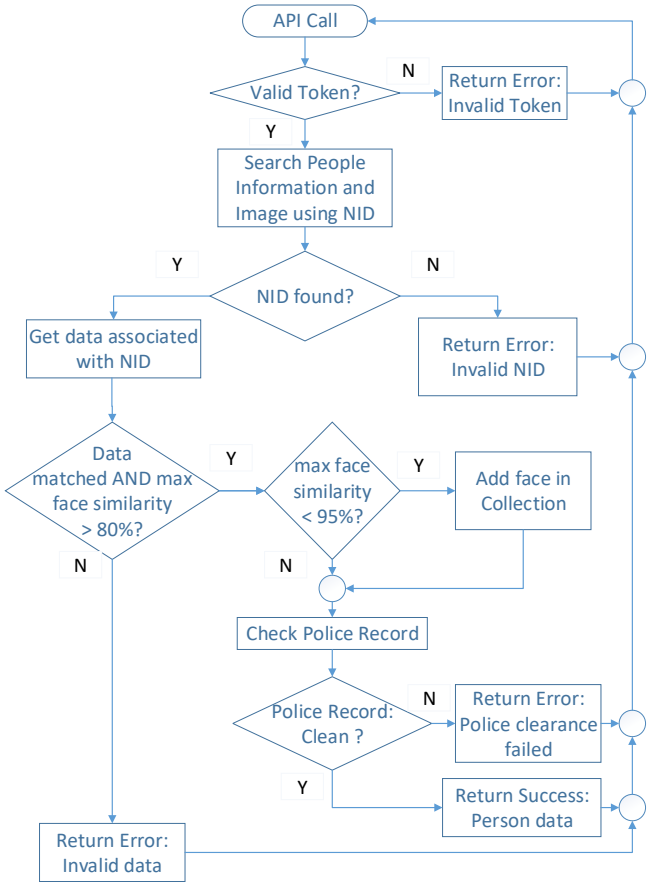

Fig. 2. System flowchart

provides full power and flexibility of SQL. The ORM is connected with MySQL database for handling the queries.

In this scenario, there are 4 subnets in the (Virtual Private Cloud) VPC. Three of them are private and one public. In each subnets the proposed framework used Application server, Database server, API Server which are in the private subnets. In the public subnet there is a bastion host which is used by the developer to SSH on each server in the VPC.

4.2 Functionalities

There are two Classic Load Balancers are used in this application. One is public facing load balancer in which users can connect to its port 80 and internally it connects to the web application. Other is internal load balancer by which the

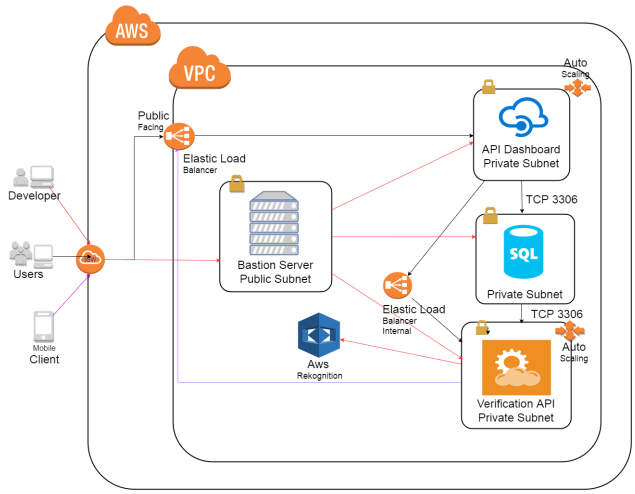

Fig. 3. System Architecture

application communicates with the API. Each load balancer there is an auto scaling group which scales the application and API servers. Mobile users can directly hit the API server through the public facing elastic load balancer.

4.3 Requirement Analysis

The necessary software and hardware tools needed to complete this work are described in Table 1 and Table 2 accordingly.

4.4 Security

The instances in the public subnet can send outbound traffic directly to the Internet, whereas the instances in the private subnet can't. Instead, the instances in the private subnet can access the Internet by using a network address translation (NAT) gateway that resides in the public subnet. The database servers can connect to the Internet for software updates using the NAT gateway, but the Internet cannot establish connections to the database servers. Web application, API and database cannot accessible from the Internet. Each subnet has its own security groups. In the security groups inbound and outbound rules are configured for the security and application needs.

5 Implementation and Results

We used Machine Learning approaches in E- governance system and developed a framework with RESTful API for citizen access.

5.1 Implementation Environment

The deployment procedure of our proposed framework in started with creating new VPC which includes three public and one private subnet. Later we create an Internet gateway and a NAT gateway from which NAT gateway is placed in public subnet and associate an EIP to it Then finally create route by adding IGW to the public subnet and NAT Gateway to the private subnet.After that we initialize security group for each subnet for security purpose and create dashboard-sec for dashboard subnet, db-sec for db-subnet, api-sec for api segment and jump sec for jump subnet. Then we create an EC2 instance for dashboard application. We deployed the code base and create an Internet facing classic load balancer.After creating a launch configuration with dashboard application, we also create auto scaling group and add load balancer that previously created. We add the scaling policy on CPU utilization which scales the servers and send alerts through emails.

We designed four tables for our database named authenticated_organization, people, aws_face, and police_record. The fields needed for the tables are listed in Table 3.People table has relation with aws_face and police_record using the national_id field. To create JSON Web Token for API call authenticity, a combination of organization's id, email address and password will be used.

5.2 Performance evaluation

Primarily the neural network contains one picture per person mapped to the persons NID. For this reason, falsification of face recognition is occurred in some cases. We used the face recognition technology (FERET) database [18] which is a dataset used for facial recognition system evaluation to measure the error rate. We mapped one individual photo with one national id. Then we tested the system using same individual's other photos available in the FERET database.

We calculate the error rate of the framework by measuring number of false results returned by the API. At first, the neural network has only one image per person. We mark this stage as single image stage. In each verification request if there is 80-95 % similarity then the framework adds the new image to neural network. By using this machine learning approach provided by AWS, it becomes easy and efficient to verify people by the increasing number of iterations. As the number of iteration is increases, the error rate is decreases. Even for this deep learning approach, this image data can be used further for video verification.

Error rate in this single image per person scenario is 1.60%. After this iteration, each searched persons have two facial data (on average) available. The error rate in this scenario is 1.13%. In third iteration, there are three facial data (on average) is available for each person. The error rate in this step is 0.27%. The improvement of each iteration is shown in Figure 4.

5.3 Scalability evaluation

To evaluate the scalability of our framework we chose CPU utilization as primary metric. The auto load testing script which is developed is used to generate

Iteration	Error Rate	Improvement
1	1.60%	N/A
2	1.13%	29.38%
3	0.27%	76.11%

Fig. 4. Performance chart in each verification iteration

artificial hit on the API endpoints. The CPU utilization of EC2 instances of People Identity Verification API is shown in Figure 5. The Y axis reflects the CPU utilization and is measured in percentage which go up by 5 percent at each level. The X axis represents time frame. The instances are configured to add multiple instances instantly in case of high CPU utilization.

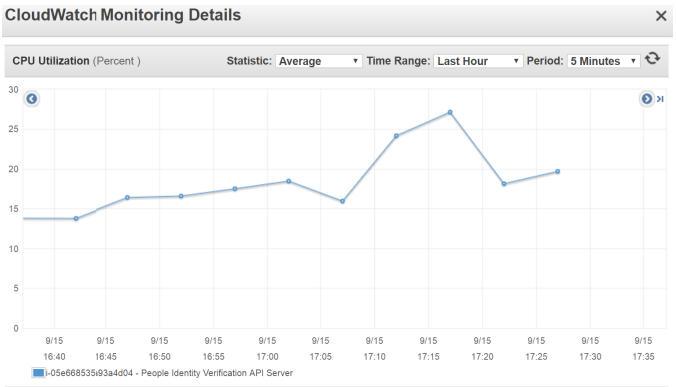

Fig. 5. CPU Utilization of People Identity Verification API

Furthermore, we generate excessive hit on the API endpoints using test script and exceeds the maximum limit of CPU power. Our system is configured to generate alert for low and excessive CPU utilization within a time range and add additional computing resources as needed. The additional resources are detached from the system when the CPU utilization becomes normal. Figure 6 shows a scenario of live monitoring the instances.

5.4 Cost evaluation

For any application framework, cost assumption is important to evaluate the efficiency and usability. We deployed our application in Amazon Web Services

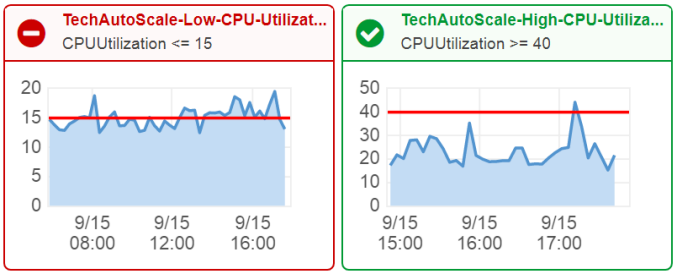

Fig. 6. Auto Scaling Alert Monitoring

(AWS) which offers a pay-as-you-go approach. With AWS the application needs to pay only for the individual services it needs, for as long as it uses them. It requires to pay for the services the application framework consumes, and once it stop using them, there are no additional costs or termination fees. If we used non cloud hosting for our application framework neither it would not scalable nor efficient. The charges of various services utilization are shown in Figure 7. This figure proves that costing is reliable and efficient for this application framework.

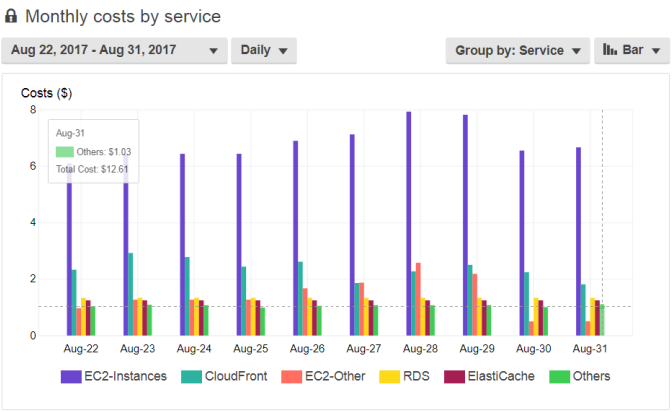

Fig. 7. People Identity Verification API Costing Graph

6 Discussion

In this section, we discuss the advantages, application areas, scopes and limitations of our framework.

6.1 Advantages

The work is useful for updating and verifying public data as well as for utilizing the e-government processes along with better government to citizen (G2C) relationship. The REST-based web API approach secures the access to the government cloud. It also enhance the artificial neural network of citizens facial data in each use of the verification API. So, our work ensures the shortening of time updating and verifying public citizen data by digitalizing the processes in a secure way.

6.2 Potential Application areas

Government and different organizations or institutions verifies applicant's data while recruiting them. They can easily verify people by accessing the g-cloud where all the citizen's data are stored. As the access mechanism is completely secure due to the RESTful API where all of them can access to the database using JWT (JSON Web Token); government need not to worry about data security. In the mean time, the appearance and other variable information can be changed; now, face image or personal profile can be updated alongside verification. Also, government can construct a secure, scalable and highly available hybrid g-cloud utilizing this framework.

6.3 Scope and Limitations

While updating the data still there is a possibility of fraud. Government must ensure that the request comes from authenticated organization as well as prioritize the previous data primarily stored in the g-cloud. Although, our work eradicate the possibility of updating wrong images as we are collecting meta-data of that face and considering the maximum similarities before storing new meta-data.

7 Conclusion

We developed an efficient approach towards the verification of people through securely accessing the public database supervised by government. We utilized the benefits of RESTful approach for ensuring security. All governments can utilize the hybrid architecture to ease the entire process thereby. Overall, this system provides public services in a faster way eradicating long time-wasting analog processes with its' features of digitization, citizen access, developed content. Here, the deep learning based people verification approach improves the performance rapidly in each iteration. It is proved by the improvement of identity verification by 29.38% and 76.11% in first and second iteration. Again application pay only for the individual services it needs by using AWS which ultimately reduce the cost. We are storing the meta-data of every updated face image of people in a neural network. Increasing number of stored meta-data opens door to track all the people in a country or within a particular region boundary. The framework can be integrated with any authorized application to verify people's identity with ease.

References

1. K. K. Smitha, Tony Thomas and K. Chitharanjan, *Cloud based e-governance system: A survey*, Procedia Engineering 38 (2012): 3816-3823.
2. Whaiduzzaman, Md, Anjum Naveed, and Abdullah Gani. "MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response." IEEE Transactions on Services Computing (2016).
3. Jian Liang, *Government cloud: enhancing efficiency of e-government and providing better public services*, In Service sciences (IJCSS), 2012 international joint conference on, pp. 261-265. IEEE, 2012.
4. Witarsyah, Deden, Teddy Sjafrizal, M. D. Fudzee, Mohd Farhan, and Mohamad Aizi Salamat, *The critical factors affecting E-Government adoption in Indonesia: A conceptual framework.*, International Journal on Advanced Science, Engineering and Information Technology 7, no. 1 (2017): 160-167.
5. Rao, Manju Natha, and S. Rama Krishna , *Efficient and Ubiquitous Software Architecture of e-Governance for Indian Administrative Services.*, International Journal of Advanced Research in Computer Science 4, no. 11 (2013).
6. Zissis, Dimitrios, and Dimitrios Lekkas, *Securing e-Government and e-Voting with an open cloud computing architecture.*, Government Information Quarterly 28, no. 2 (2011): 239-251.
7. WenJun Zhang and Qi Chen, *From E-government to C-government via Cloud Computing*, In E-Business and E-Government (ICEE), 2010 International Conference on, pp. 679-682. IEEE, 2010.
8. Sajjad Hashemi, Khalil Monfaredi and Mohammad Masdari, *Using cloud computing for e-government: challenges and benefits*, International Journal of Computer, Information, Systems and Control Engineering 7, no. 9 (2013): 596-603.
9. Wojciech Cellary and Sergiusz Strykowski, *E-government based on cloud computing and service-oriented architecture*, In Proceedings of the 3rd international conference on Theory and practice of electronic governance, pp. 5-10. ACM, 2009.
10. Haibin Ling, Stefano Soatto, Narayanan Ramanathan, and David W. Jacobs, *A study of face recognition as people age*, In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1-8. IEEE, 2007.
11. Unsang Park, Yiying Tong, and Anil K. Jain, *Age-invariant face recognition*, IEEE transactions on pattern analysis and machine intelligence 32, no. 5 (2010): 947-954.
12. Haibin Ling, Stefano Soatto, Narayanan Ramanathan, and David W. Jacobs, *Face verification across age progression using discriminative methods*, IEEE Transactions on Information Forensics and security 5, no. 1 (2010): 82-91.
13. Yi Sun, Xiaogang Wang, and Xiaoou Tang, *Deep learning face representation from predicting 10,000 classes*, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1891-1898. 2014.
14. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, *Deepface: Closing the gap to human-level performance in face verification*, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1701-1708. 2014.
15. Christensen, Jason H., *Using RESTful web-services and cloud computing to create next generation mobile applications*, In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, pp. 627-634. ACM, 2009.
16. Adamczyk, Paul, Patrick H. Smith, Ralph E. Johnson, and Munawar Hafiz, *Rest and web services: In theory and in practice*, In REST: from research to practice, pp. 35-57. Springer New York, 2011.

17. "Amazon Web Services (AWS) - Cloud Computing Services" [Online] Available: https://aws.amazon.com/ (Date last accessed 05-September-2017)
18. "Color FERET Database" [Online] Available: https://www.nist.gov/itl/iad/image-group/color-feret-database (Date last accessed 03-October-2017)

Algorithm 1: Algorithm for people verification

```
API call;
if TOKEN in Request = True then
    if Validity_TOKEN = True then
        if NID, personal information & image in Request then
            if NID_found = True then
                if personal information & image matches with database then
                    Get Face_List from Database;
                    Set Max_Score = 0 ;
                    foreach Face in Face_List do
                        Calculate similarity score;
                        Set Max_Score = max(score, max_score);
                    end
                    if Max_Score > 80 then
                        if Max_Score < 95 then
                            Store image in Collection;
                        end
                        Get Records_Police ;
                        if Clearence = True then
                            Return: Verification Success with all data;
                        end
                        else
                            Return Error with case record details;
                        end
                    end
                    else
                        Return Error: Image match failed;
                    end
                end
                else
                    Return Verified = True;
                end
            end
            else
                Return Error: NID not found;
            end
        end
        else
            Return Error: Missing mandatory fields;
        end
    end
    else
        Return: TOKEN not valid;
    end
end
else
    Return: TOKEN not found;
end
```

Table 1. Software Description

Software	Version
Python	Python 3.6.1
Flask	Flask version 0.12.2
MySQL	MySQL version 5.7.19
SQLAlchemy	Flask-SQLAlchemy version 2.2
JSON Web Tokens	PyJWT version 1.5.2 and Flask-JWT 0.3.2
AWS CLI	awscli version 1.11.112, boto3 version 1.4.4 and botocore version 1.5.75

Table 2. Hardware Description

Hardware	Description
VPC	A VPC with three private subnets and one public subnet is created to host the whole application.
Bastion Node	Bastion server is needed to access the others servers as it includes the public inbound rules and private outbound rules. We have chosen m1-large EC2 instance for this node.
API and API Dashboard Node	The main Flask API and its dashboard are deployed in EC2 instances. For both operation, we have chosen m1-large instance which include Auto scaling groups. The AWS services like Rekognition, AMI, AWSCLI are performed from these nodes.
Database Node	The database is hosted in this node. It is also a m1-large EC2 instance and has only private access from the other nodes.
Load Balancers	Two Elastic Load Balancers ensure to handle the inbound traffic for both API access and API dashboard access.
Security Groups	Proper security groups are created for handing the inbound and outbound traffic in each node and services to restrict unauthenticated access of the application.

Table 3. Fields of Database

Table Name	Fields
authenticated_organization	organization_id (PK), organization_email_address, organization_password, organization_name, organization_address, organization_phone_number, organization_email_address, organization_registration_id, organization_registration_address, organization_details
people	national_id (PK), full_name, father_name, mother_name, date_of_birth, gender, birth_district, permanent_address, present_address, email_address, phone_number, blood_group, nationality, religion, marital_status, spouse_name, tin_number, passport_number, driver_license_number, photo_url, created_timestamp, updated_timestamp
aws_face	face_id (PK), face_created_time, face_url, national_id (FK)
police_record	case_id (PK), case_status, case_description, case_location, case_outcome, case_created_time, case_updated_time, case_complainer_national_id (FK), case_investigator_national_id (FK), case_defendant_national_id (FK)