

EMPLOYEE PRODUCTIVITY ANALYSIS SYSTEM USING BLOCKCHAIN FOR SMART CITY GOVERNANCE

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

MULLAPUDI KOTESWARA RAO

(37110465)

CHERUKURI BHARGAV SRIRAM

(37110138)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bona fide work of **MULLAPUDI KOTESWARA RAO (Reg. No. 37110465) and CHERUKURI BHARGAV SRIRAM (Reg. No. 37110138)** who carried out the project entitled "**EMPLOYEE PRODUCTIVITY ANALYSIS SYSTEM USING BLOCKCHAIN FOR SMART CITY GOVERNANCE**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Dr. J.REFONAA.,M.E,Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **MULLAPUDI KOTESWARA RAO** and **CHERUKURI BHARGAV SRIRAM** hereby declare that the Project Report entitled “**EMPLOYEE PRODUCTIVITY ANALYSIS SYSTEM USING BLOCKCHAIN FOR SMART CITY GOVERNANCE**” is done by me under the guidance of Dr. **J.REFONAA.,M.E,Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J.REFONAA, M.E., Ph.D., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In recent years, the use of blockchain has gained attention in industrial interventions and research. Blockchain technologies can be particularly utilized in the domain of Smart Governance, which is a critical component of Smart Cities. Smart Governance aims to ensure smarter decision making, openness and collaborative participation of all entities involved as well as help minimize resource usage, reduce consumption and save cost. Smart Governance involves putting in place systems for monitoring and evaluating the performance of government workers. However, government workers lack trust in these systems because they are mostly centralized systems which are prone to single-point of failure problems and also data fed into these systems can be tampered with by the higher-ups or employers to favour individual workers. With these problems, there is a need for a decentralized system that can ensure trust, tamper- resistance, accountability, reliability, transparency and security. In this paper, we propose a decen- tralized employee evaluation system based on blockchain. We utilized a 3-chain model, namely; ID-Chain, Behavior Chain and Credit Chain fused to make a single united chain. Our system also makes use of an automated game-based employee evaluation approach. Our system is based on Hyperledger Fabric as the blockchain platform and operating mechanism. Our Proof of Concept (PoC) results shows that in utilizing the Blockchain system, security concerns such as trust, privacy, accountability, among others, that are present in current employee performance evaluation systems are tackled and also in the adoption of the game-based automated system, our system can make an effective decision regarding employee performance. Our system achieves trust, transparency, security and accountability among government workers under a Smart City governing environment.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	vii

CHAPTER NO	TITLE	PAGENO
1	INTRODUCTION	1
	1.1 INTRODUCTION	3
2	LITERATURE SURVEY	3
	2.1 Existing System	3
	2.2 Proposed System	4
3	SYSTEM DESIGN & METHODOLOGY	5
	3.1 PURPOSE	5
	3.2 PROJECT SCOPE	5
	3.3 PRODUCT PERSPECTIVE	5
	3.4 SYSTEM FEATURES	8
	3.5 DESIGN AND IMPLEMENTATION	9
	3.5.1 CONSTRAINTS IN ANALYSIS	10

	3.5.2 CONSTRAINTS IN DESIGN	11
	3.5.3 CONSTRAINTS IN IMPLEMENTATION	12
	3.6 OTHER NON-FUNCTIONAL REQUIREMENTS	14
	3.6.1 PERFORMANCE REQUIREMENTS	16
	3.6.2 SAFETY REQUIREMENTS	17
4	SOFTWARE DEVELOPMENT METHODOLOGY	22
	4.1 JAVA	22
	4.1.1 INTRODUCTION TO JAVA	23
	4.1.2 WORKING OF JAVA	25
	4.1.3 THE JAVA PROGRAMMING LANGUAGE	27
	4.1.4 THE JAVA PLATFORM	28
	4.1.5 APACHE TOMCAT SERVER	29
	4.2 INTRODUCTION TO BLOCKCHAIN	30
	4.3 K-NEAREST NEIGHBORS	31
5	RESULTS AND DISCUSSIONS	39
6	CONCLUSION AND FUTURE WORK	40
	REFERENCES	41
	APPENDIX	
	A. SOURCE CODE	54

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	BLOCK DIAGRAM	8
4.1.3.1	JAVA VIRTUAL MACHINE	11
4.1.4.1	JAVA PLATFORM	12
4.3.1	KNN ALGORITHM	18
4.3.2	KNN ALGORITHM GRAPH	19
4.3.3	CHOOSING FACTOR K	20
4.3.4	KNN GRAPH	21
4.3.5	KNN GRAPH	21
4.3.6	SHA256 HASH	23
4.3.7	BLOCK	23
4.3.8	BLOCKCHAIN	23
5.1	ORGANISATION FEEDBACK	25
5.2	ORGANISATION RESULT	25
5.3	MANAGER LOGIN	26
5.4	MANAGER	26
5.5	EMPLOYEE LIST	27
5.6	MANAGER FEEDBACK	27
5.7	FORM TASK ASSIGNMENT	28
5.8	ADMIN LOGIN	28
5.9	EMPLOYEE PERFORMANCE EVALUTION	29
5.10	OVERALL YEAR!!	29

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION:

Since the digital age, governments have been looking for ways to enhance the quality of lives of their citizens by improving the technological infrastructures and seize the opportunity to avoid disruptions. Governments now invest in an aggressive and different innovation strategy that focuses on change, revolution and results [1]. The concept of Smart City has been gaining attention recently due to its potential of creating a city whose system is sophisticated and able to sense and act independently [2]_[4]. A smart city should be able to embrace all the parties involved in political participation, functioning of local administration and services for its members (citizens). Authors in [2] de_ne Smart City as a city that functions well in a six- dimensional domain: smart people, smart economy, smart living, smart mobility, smart environment and smart governance. The ulti- mate purpose of constructing a smart city is to enhance the quality of living of the general citizens by seizing the bene_ts from the existing modern technologies. Smart City research is a strand of a local electronic government research area. As a component of Smart City, smart governance aims at applying adequate fair and transparent policies, geared towards improving the lives of its citizens [3]. Its prime focus is to ensure a creative, innovative, economic growth, excellent quality of life and must be environmentally and socially secure and safe [3]. Moreover, the performance of government officers working in the Smart City environment needs to be evaluated in order to reward excellent performance or improve on poor performance. In a general view, the implementation of smart city projects is a transformation process that requires policies, accompanied by objectives, managed by active leadership management which uses advanced technologies for improving its efficiency. With the same idea, we narrow it down to an organizational scenario where the processes, as mentioned earlier, is implemented. As employees are the primary backbone of every organization, and a proper performance assessment can provide an accurate image

of an employee status at any given time. The purpose of employee evaluation is not only to evaluate the lousy

performance and enhance it but also to promote staff to work better. An adequately designed employee assessment system can contribute to organizational goals and improve the efficiency of employees. It serves as a measure for current and future improvements of an organization in order to meet their goals and objectives. It has been proved by many research works that an effective appraisal system is directly related to employee motivation and productivity [5], [6]. Despite the enormous technological developments, employee performance in most organizations remains manual. The manual employee assessment systems are incredibly prone to human prejudice since managers can assess employee performance [7]. Biases can occur because of human nature or if managers receive insufficient training and produce impartial feedback. The managers generally concentrate exclusively on the latest worker results, while lousy ignored in the early portion of the assessment period. Besides, it is hard for the promoted managers to rate who were his workers once. Moreover, the performance benefits that come with these technologies. The motive for technology modernization is to be able to simplify complex processes while allowing innovation. Governments now seek to comprehend how these new concepts and technologies like Smart City and Distributed Ledger Technologies (DLT) can play a part. The Fourth Industrial Revolution (Industry 4.0) has arrived, and governments recognize the necessity to evaluations have very little priority from the managers, as they are a time-consuming task [8]. All these variables lead to a biased performance assessment which results in discontent among those employees who receive biased feedback. Consequently, the integrity of the performance assessment scheme is wiped off intentionally or unintended. According to a study in [9], the present performance assessment scheme discovered that only 36 percent of staff were satisfied. Therefore, human biases must be eliminated during performance assessments, and one of the options is to move towards an automated performance assessment scheme. However, automated employee assessment schemes are still facing similar challenges of trust accountability. Therefore, a technology that ensures confidence, transparency while at the end of performance assessments, offering the staff with a rewarding experience is needed. This is where blockchain comes in. Blockchain, which originated from the cryptocurrency Bitcoin [10], is one of the most ground-breaking technologies of this era. It can store and share

transparent, safe, controllable and unalterable information through distributed ledger technology (DLT). It is capable of making organizations decentralized, secure, accountable and

transparent. Thus, we propose a fully decentralized employee assessment system that is based on blockchain in order to ensure trust, transparency and accountability in a Smart City environment. Our investigations show that this is the first of its kind. In this paper, we denote **Government Officers** working in a Smart City as **Employees** and the **Government Institution** tasked with evaluating their performance as **Organization**.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

In the existing system each organization makes an effort of finding the employee performance in many ways. organization uses lot like sensors to get the behavior of employee but there is problem in it like any one can attack the network of the sensor or accidentally break the sensor some important information can be lost and also it get only one parameter of employee behavior other information are not gathered for the evaluation of the employee some time accurate data is not also produced by the sensor and all the employee data is stored in the database that can be tampered.

Problem Definition

Single Parameter of user behavior is considered but not getting the other parameters accurately

Sensors can be broken anytime and can happen malfunction

Data stored in single database so tampering is possible

2.2 PROPOSED SYSTEM

In the proposed system we propose an employee productivity analysis tool to evaluate the employee performance in an organization by collecting the employee feedback of the objective given to him and other aspects. First the application is done for software IT domain which operates in smart city. In this the first part is to get the employee feedback of the organization status by providing the review form to the employee and make him to write the feedback of the organization. Employee input text are forwarded to find the sentiments for that we use Machine learning (NLP) and based on that positive and negatives are found for the organization. Second manager module which assigns tasks for the user and reviews it and also views every employee and provides feedback for the employee it will be considered in the evaluation process. Third admin module contains the all details of the employee, manager, organization feedback result and mainly the employee performance analysis result also produced by the admin. Fourth module employee module contains tasks which is assigned by the manager and also contains self evaluation form that is given to employee to access their view of her/him. User details like biometric, rfid and employee performance data are stored in block chain for security purposes.

Advantage:

Multiple parameters of the employee is considered to evaluate performance to get best result

Employee data is stored in block chain so information cannot be changed

Employee view of organization is also viewed by the organization

CHAPTER 3

Aim and Scope of the present investigation

Project Purpose and Scope

3.1 Purpose

The main aim of this project is to access the employee performance and to predict employee future performance.

3.2 Project Scope

Blockchain can be defined as a decentralised and distributed digital ledger that is used to record transactions across many computers. In simple terms, we can consider any record within Blockchain technology as a document shared with a group of people (computers/servers that also known as blocks).by the use of we can maintain our data safely and avoid intruder and by the help of machine learning technique we can achieve to get sentiments of sentence and calculate score for the employee

3.3 Product Perspective

We propose a smart way to evaluate the performance of employees by using machine learning techniques.

3.4 System Features

In the existing system each organization makes an effort of finding the employee performance in many ways. organization uses lot like sensors to get the behavior of employee but there is problem in it these devices will monitor each

employee 24/7 it hugely affects employee privacy concerns and it also indicates lack

of trust in each employee which result in resentment and reduce employee productivity likewise the employee data can be misused by the organization after then each employee data has to be evaluated by the higher officials manually its time consuming process the higher official has to check each and every part and provide score and only single parameter of employee data is considered.

3.5 Design and Implementation Constraints

3.5.1 Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts
- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

3.5.2 Constraints in Design

- ◆ Determination of the Involved Classes
- ◆ Determination of the Involved Objects
- ◆ Determination of the Involved Actions
- ◆ Determination of the Require Clauses
- ◆ Global actions and Constraint Realization

3.5.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or

functionality associated with a flat relation. A flat relation corresponds with the

relation concept of entity-relationship modeling and many object oriented methods.

3.6 Other Nonfunctional Requirements

3.6.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

embedded browser in charge of the navigation and accessing to the web service;

Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

3.6.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in the widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.

6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

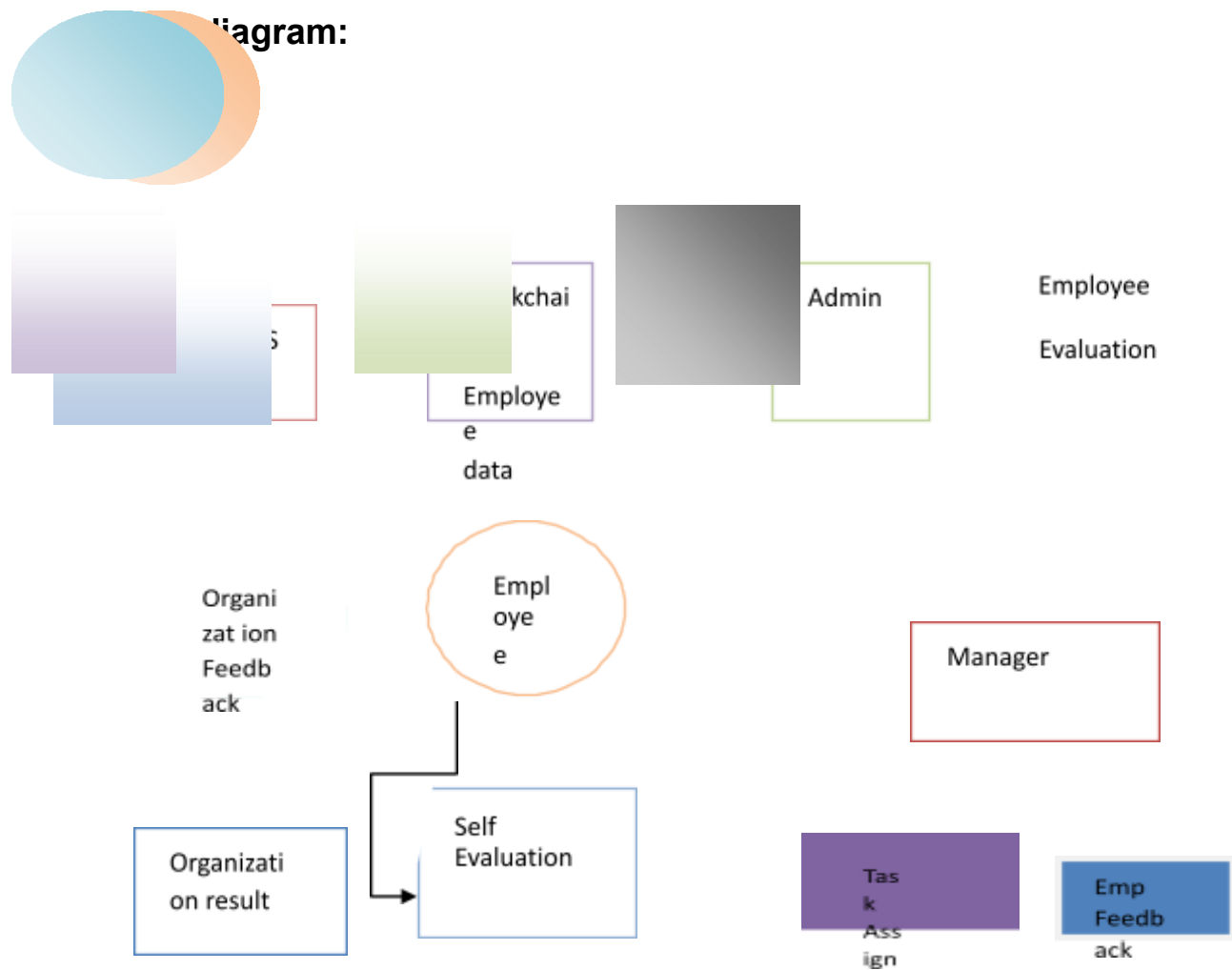


Fig 3.1:BLOCK DIAGRAM

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 JAVA

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

4.1.1 INTRODUCTION TO JAVA

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents, was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many

confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

4.1.2 WORKING OF JAVA

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of **main**, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the `argv []` of C), and is declared as a static method.

To output text from the program, we execute the **println** method of **System.out**, which is java's output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user's program.

Java consists of two things :

Programming language
Platform

4.1.3 THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

Simple
Object-oriented
Distributed
Interpreted

Robust

Secure

Architecture-neutral

Portable

High-performance

Multithreaded

Dynamic

The code can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

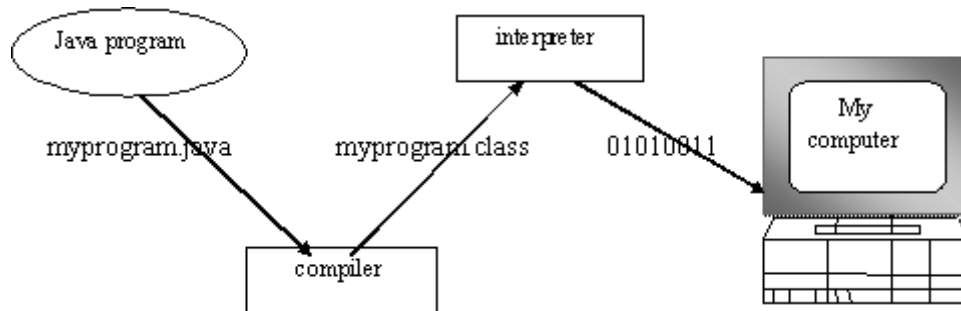
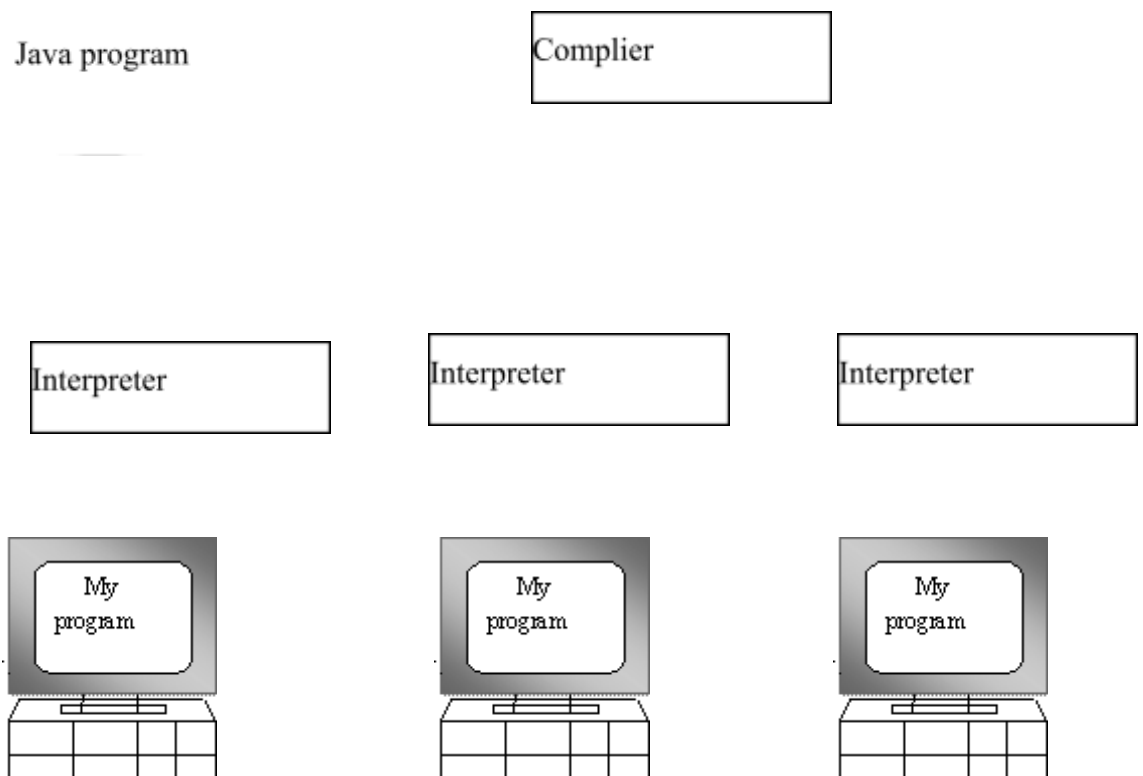


Fig.4.1.3.1

You can think of Java bytecodes as the machine code instructions for the **Java Virtual Machine (JVM)**. Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can be run on Windows NT, Solaris and Macintosh



PC-Compatible
Power macintosh

Sun Ultra Solaris

Windows NT

System 8

4.1.4 THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

The Java Virtual Machine (JVM)

The Java Application Programming Interface (Java API)

You've already been introduced to the JVM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (**packages**) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

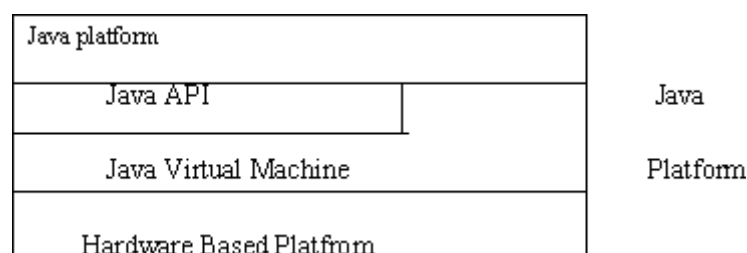


Fig.4.1.4.1

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

4.1.5 APACHE TOMCAT SERVER

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs and servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

Product features

Tomcat 3.x (initial release)

- implements the Servlet 2.2 and JSP 1.1 specifications

- servlet reloading
- basic HTTP functionality Tomcat 4.x
- implements the Servlet 2.3 and JSP 1.2 specifications
- servlet container redesigned as Catalina
- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based administration
- Tomcat 5.x
- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration
- faster JSP parsing

History

Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

4.2 Introduction to Blockchain

With the emergence of Digital Currency (aka Crypto currency), several enterprises or financial institutions are experimenting with the Distributed Ledger system as a trusted way to track the ownership of the assets without any central authority.

The core system behind the new currency system is Blockchain technology. A walkthrough of the basic building blocks of the Blockchain technology is described

below.

One way to fix it is to run mining and recalculate the Hash value of Block #1 which basically will generate a new value for Nonce and hence leading to a valid Hash value which starts with 4 zeroes. Copying this to next Block #2's Prev field will fix these 2 Blocks. However in order to fix the whole Blockchain, we need to continue with this process for all the Blocks in the chain so that all Blocks point to new & valid

Hash codes of their previous blocks.

The cost of fixing the tampered Blockchain as described in the above process is very high. Because we have to go and fix the Chain from the starting Block to the last one. In case the Chain is large, it becomes a costly operation. In the case of a Distributed Blockchain where several Peers are involved in the process and keeping the copy of the Blockchain, repairing the Blocks becomes an even more costly operation.

The other and more efficient process is to come up with the compensating data and add this Block at the end of the Chain. E.g. In case your Chain contains the financial transaction (money movement) in Data field of the Block, then instead of fixing each of the Block's Data with corrected financial transaction, come up with the adjusted financial transaction (aka compensating transaction) and create a Block (with Data=adjusted transaction record) and add this Block to the Blockchain (adds to the end of the Chain).

4.3 K-Nearest Neighbors

Introduction

In four years the analytics built more than 80% of classification models and just 15-20% regression models. These ratios can be more or less generalized throughout the industry. The reason for a bias towards classification models is that most analytical problems involve making a decision. For instance will a customer attrite or not, should we target customer X for digital campaigns, whether customer has a high potential or not etc. This analysis is more insightful and directly links to an implementation roadmap. In this article, we will talk about another widely used classification technique called K-nearest neighbors (KNN). Our focus will be primarily on how does the algorithm work and how does the input parameter affect the output/prediction.

1. KNN algorithm

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

Let us take a few examples to place KNN in the scale:

	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

KNN algorithm fares across all parameters of considerations. It is commonly used for its ease of interpretation and low calculation time.

2. The KNN algorithm work

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS):

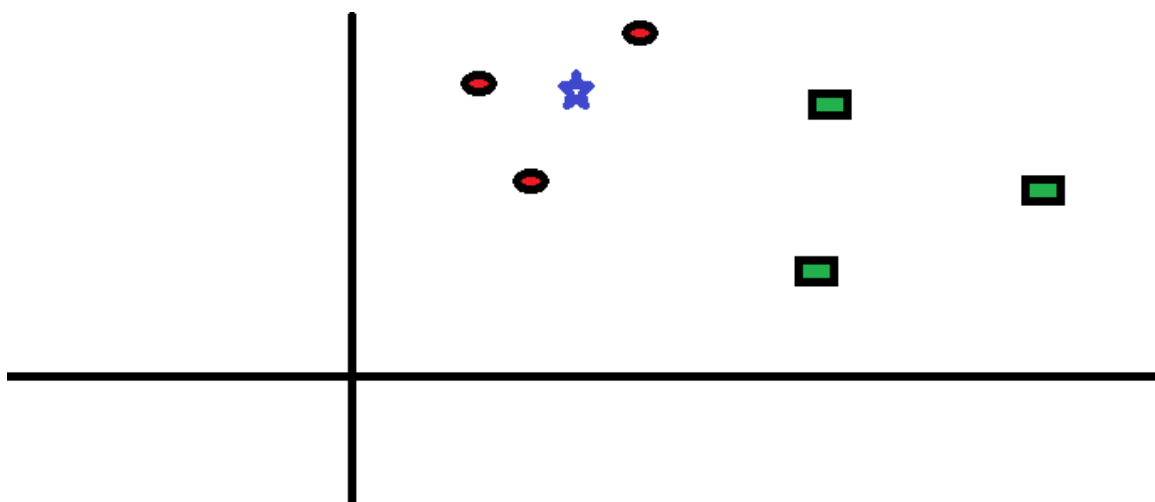


FIG 4.3.1

You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” algorithm is the nearest neighbors we wish to take vote from. Let’s say $K = 3$. Hence, we will now make a circle with BS as center just as big as to enclose only three data points on the plane. Refer to following diagram for more details:

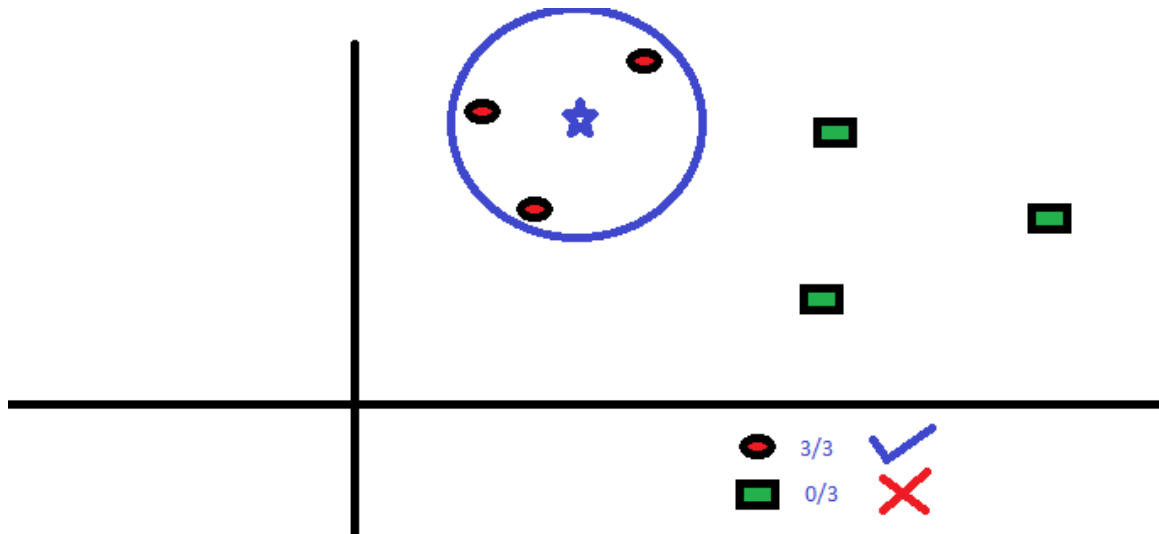


FIG 4.3.2

The three closest points to BS are all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

3.

4. How do we choose the factor K ?

First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observations remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let’s try to see the effect of value “ K ” on the class boundaries. Following are the different boundaries separating the two classes

with different values of K .

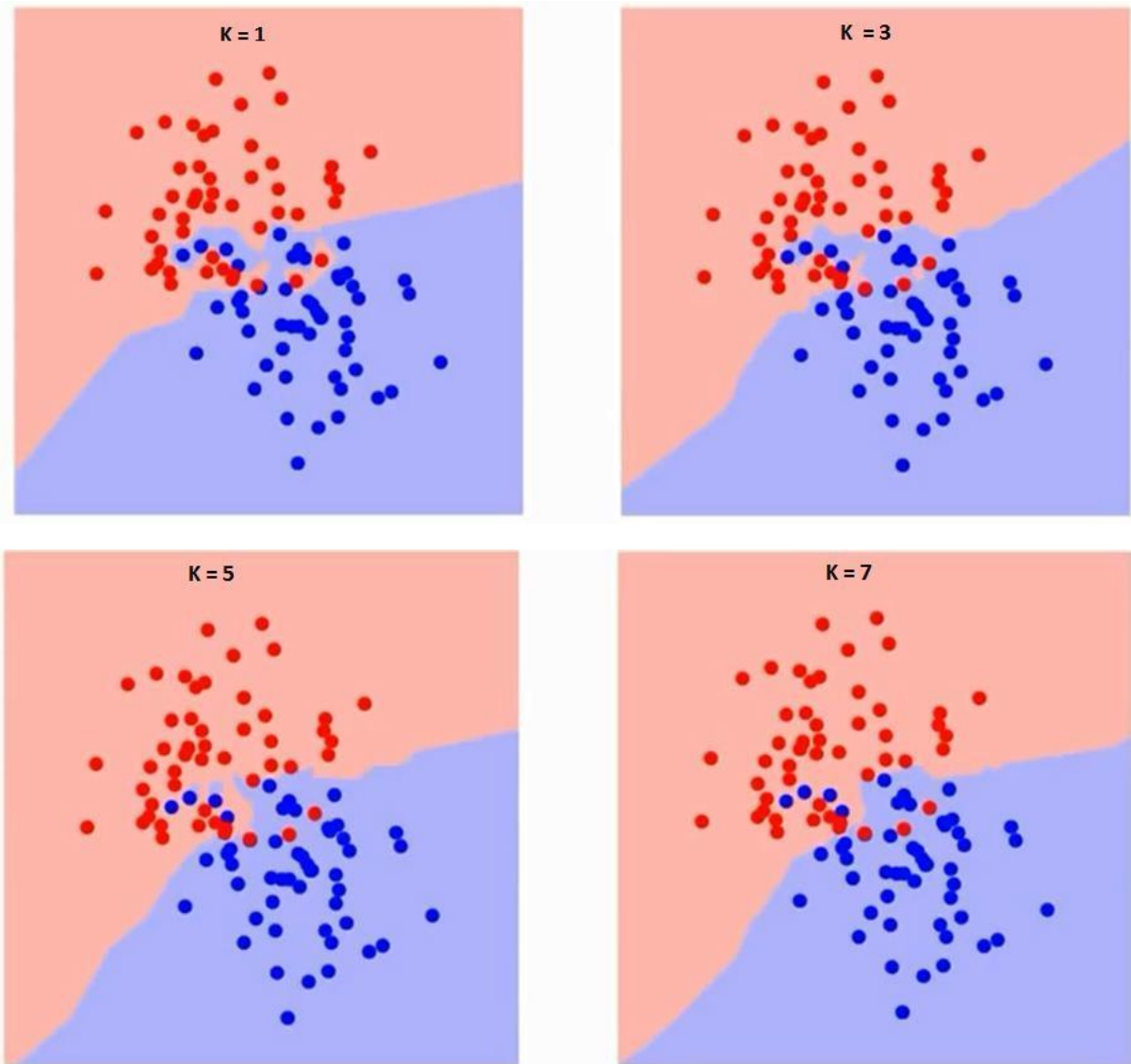


FIG 4.3.3

If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K-values. Following is the curve for the training error rate with varying value of K:

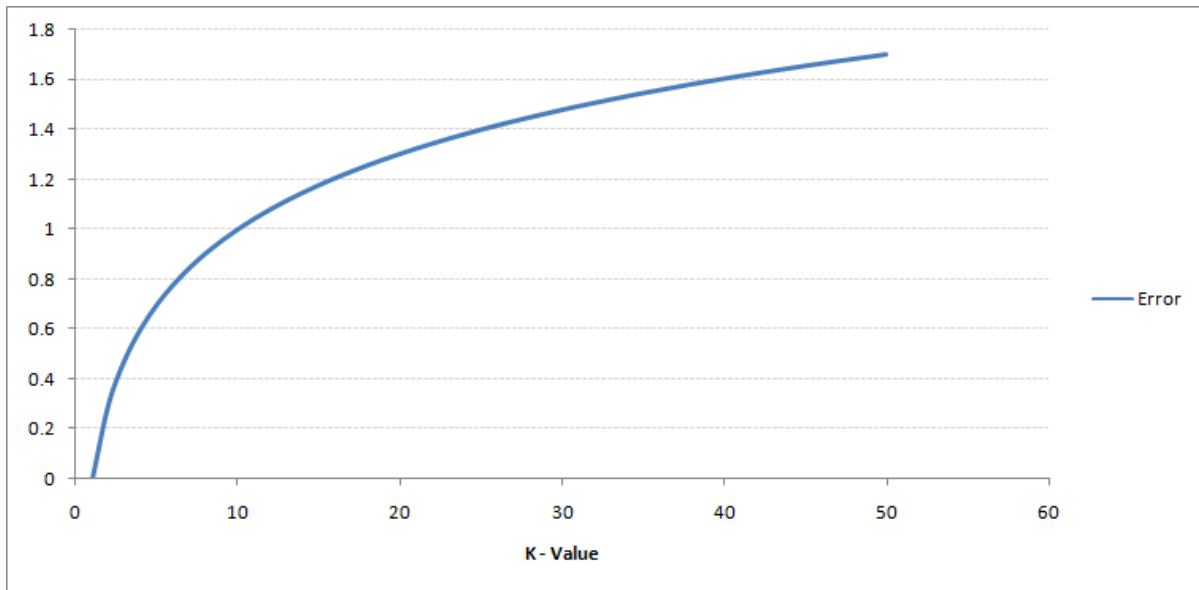


FIG 4.3.4

As you can see, the error rate at $K=1$ is always zero for the training sample. This is because the closest point to any training data point is itself. Hence the prediction is always accurate with $K=1$. If the validation error curve would have been similar, our choice of K would have been 1. Following is the validation error curve with varying value of K :

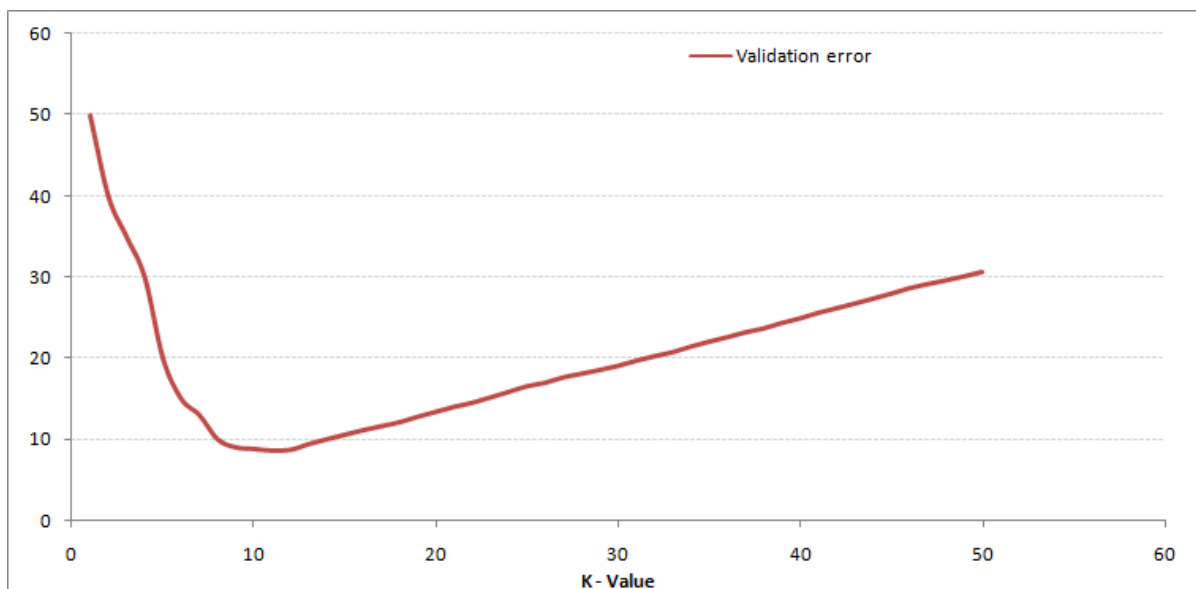


FIG 4.3.5

This makes the story more clear. At $K=1$, we were over fitting the boundaries. Hence, error rate initially decreases and reaches a minimum. After the minimum point, it then increases with increasing K . To get the optimal value of K , you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K . This value of K should be used for all predictions.

Breaking it down – Pseudo Code of KNN

We can implement a KNN model by following the below steps:

2. Load the data
3. Initialize the value of k
4. For getting the predicted class, iterate from 1 to total number of training data points
 1. Calculate the distance between test data and each row of training data.
Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
 2. Sort the calculated distances in ascending order based on distance values
 3. Get top k rows from the sorted array
 4. Get the most frequent class of these rows
 5. Return the predicted class

6. Conclusion

KNN algorithm is one of the simplest classification algorithms. Even with such simplicity, it can give highly competitive results. KNN algorithms can also be used for regression problems. The only difference from the discussed methodology will be using averages of nearest neighbors rather than voting from nearest neighbors.

SHA256 Hash

Data:

test data

Hash:

916f0027a575074ce72a331777c3478d6513f786a591bd892da1a577bf2335f9

FIG 4.3.6

Block

Block:

#1

Nonce:

72608

Data:

Hash:

0000f727854b50bb95c054b39c1fe5c92e5ebcf4bcb5dc279f56aa96a365e5a

Mine

FIG 4.3.7

Blockchain

Block:

#1

Nonce:

20839

Data:

test data 1

Prev:

00

Hash:

00002a70c8d0034addeab115689ba9e79c8b8dbbd81b083be396c199bf

Mine

Block:

#2

Nonce:

81984

Data:

test data 2

Prev:

00002a70c8d0034addeab115689ba9e79c8b8dbbd81b083be396c199bf

Hash:

0000a97e44b78fba8baabc2523d45e4a3cec092af26b185f29cd1336c94e

Mine

Block:

#3

Nonce:

79796

Data:

test data 3

Prev:

0000a97e44b78fba8baabc2523d45e4a3c

Hash:

0000f2926342c369daff2bcc3fdad34f792

Mine

FIG 4.3.8

CHAPTER 5

RESULTS AND DISCUSSION

In the proposed system we propose an employee productivity analysis tool to evaluate the employee performance in an organization by collecting the feedback. The objective of the system is to get feedback for employee performance from different part. First the application is done for software IT domain which operates in smart cities. In this the first part is to get the employee feedback of the organization status. Second manager module which assigns tasks for the user and reviews it and also views every employee and provides feedback for the employee it will be considered in the evaluation process. Third admin module contains the all details of the employee, manager, organization feedback result and mainly the employee performance analysis result also produced by the admin. Fourth module employee module contains tasks which is assigned by the manager and also contains self evaluation form that is given to employee to access their view of her/him. For employee login Biometric and Rfid is used and employee performance data are stored in a block chain server for security purposes.

MODULES

- **Organization Feedback**
- **Manager**
- **Admin**

MODULE EXPLANATION:

Organization Feedback:

The screenshot shows a web browser window with the URL `localhost:8888/BEMPS/organization.jsp?email=12`. The page is titled "Smart Institute" and "ORAGANIZATION FEEDBACK". It contains a form with the following fields:

- ID:
- Choose Year:
- About Infrastructure:
- About Management:
- About Food:
-

The Windows taskbar at the bottom shows the time as 4:47 PM on 1/30/2021.

FIG 5.1

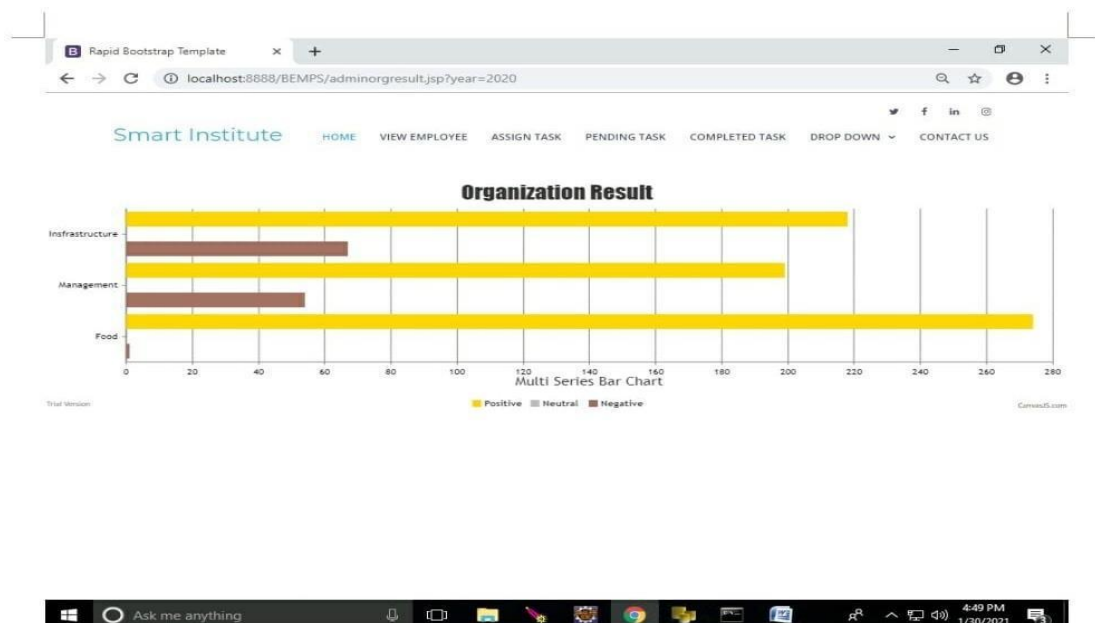


FIG 5.2

In this module an employee can make a review of the organization view of his\her organization and make use of this review to improve particular areas.

Manager

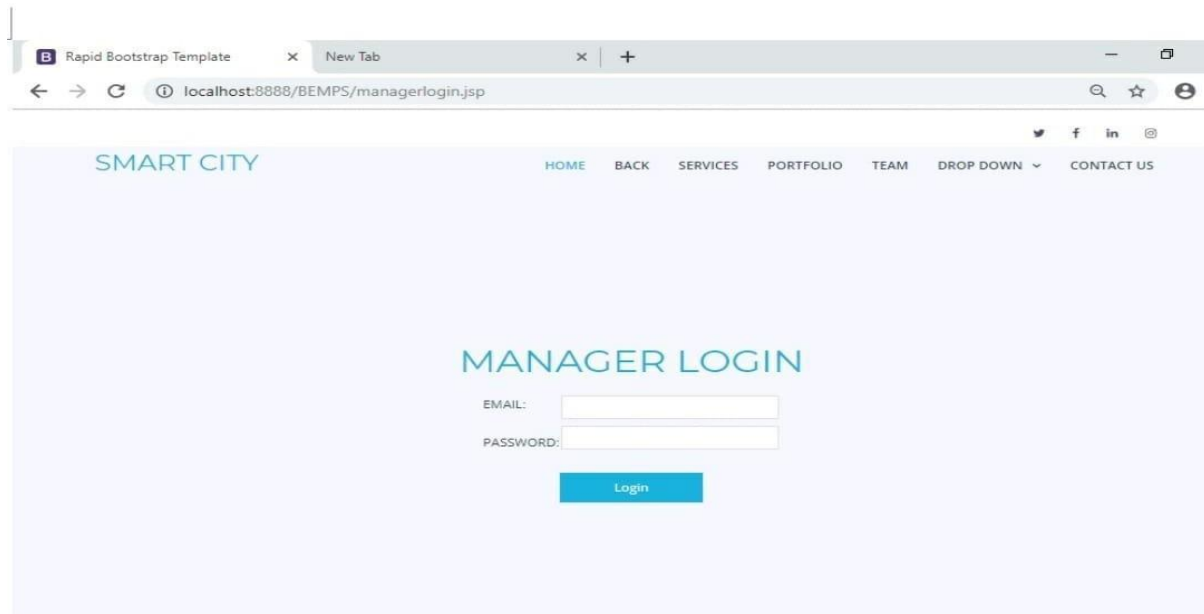


FIG5.3

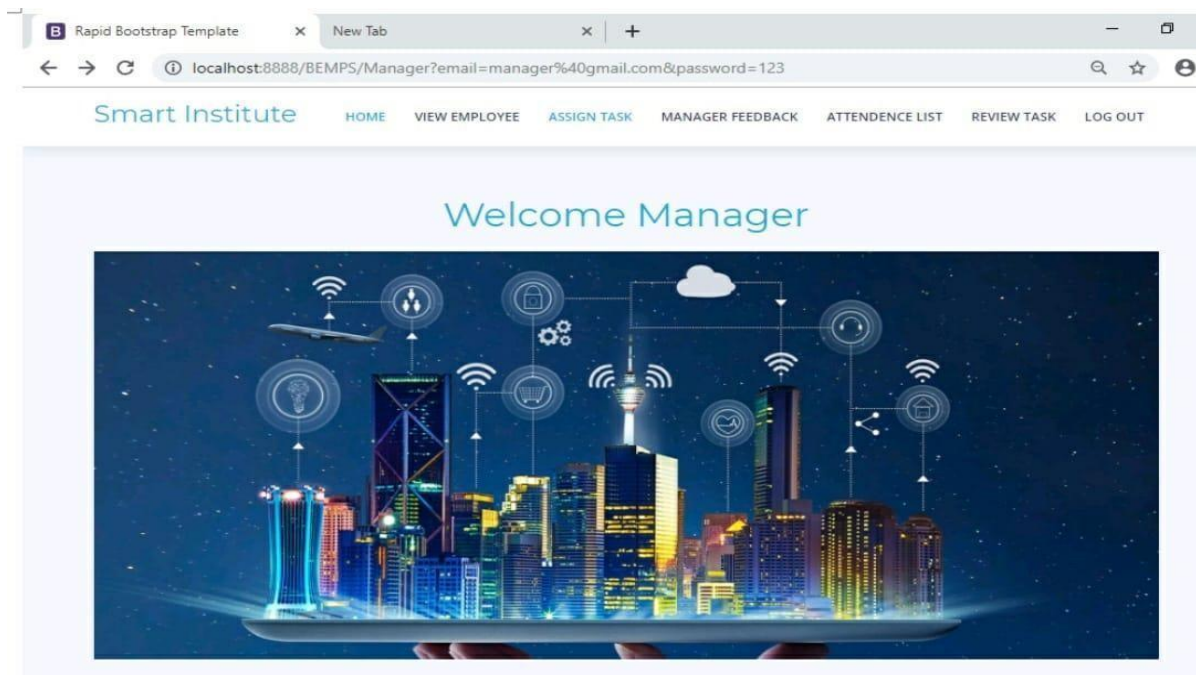


FIG 5.4

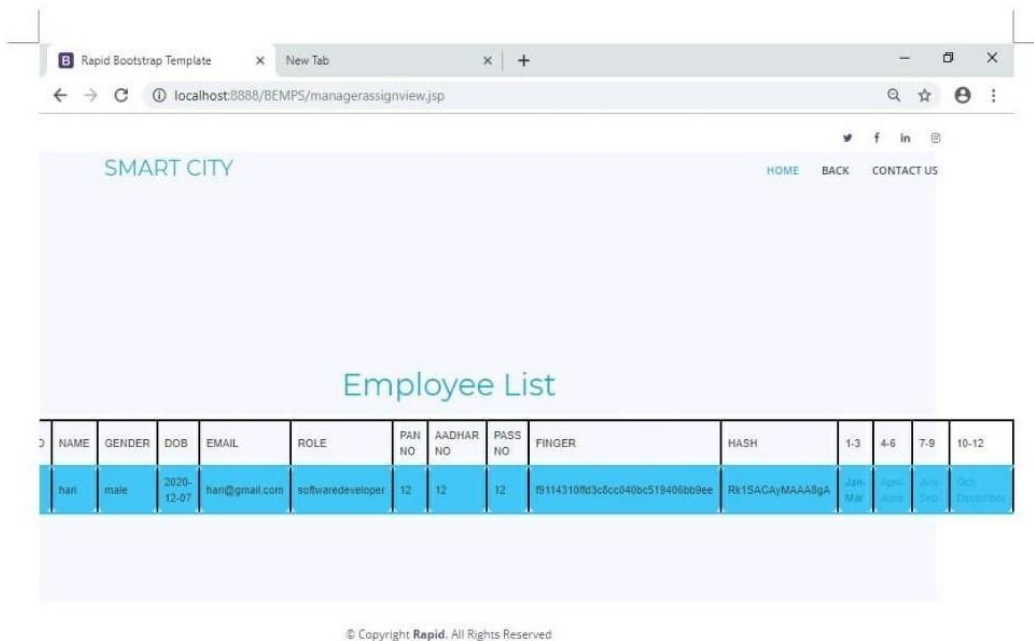


FIG 5.5

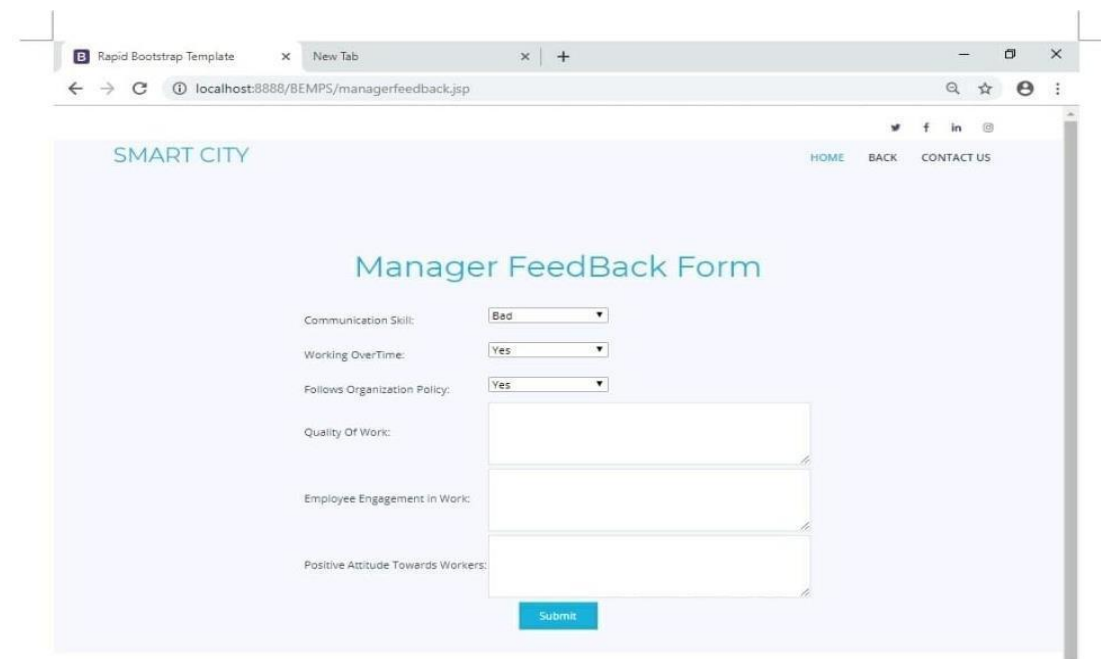


FIG 5.6

SMART CITY

HOME BACK CONTACT US

Task Assignment

ID:

Name:

Role:

Project Name:

Task:

Start Date:

End date:

Deadline:

Resources:

Month:

FIG 5.7

In this module manager can assign task for the employee and provide feedback for the employee behavior towards organization and work

Admin

SMART CITY

HOME SERVICES PORTFOLIO TEAM DROP DOWN CONTACT US

Admin Login

Email:

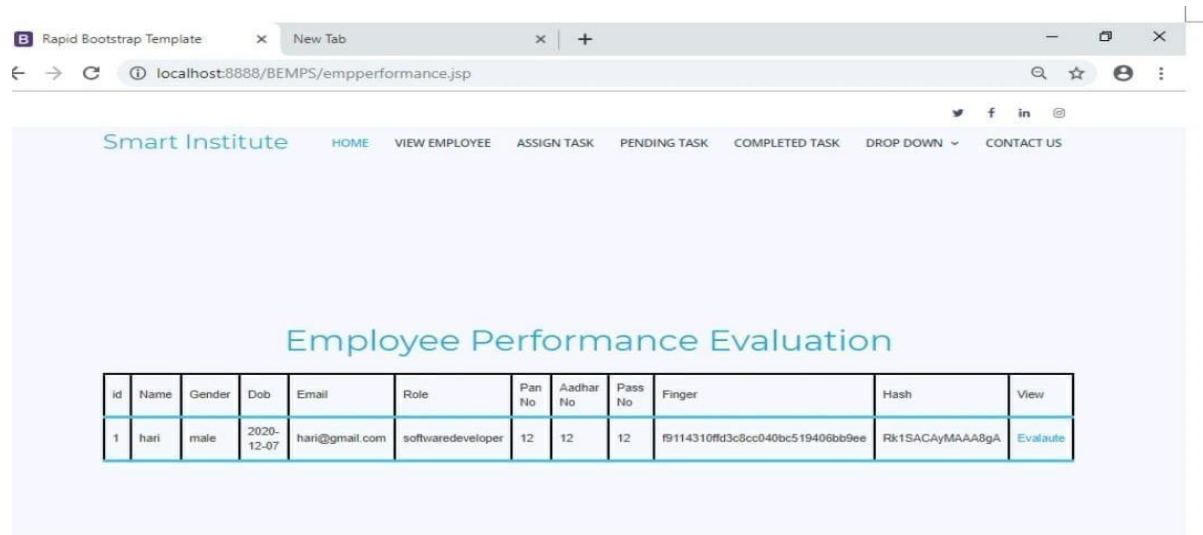
Password:

FIG 5.8

In this module admin can view all information of the employee and manager and can view the result of employee assessments and organization result

Personalized Itinerary

Employee:



Smart Institute

HOME VIEW EMPLOYEE ASSIGN TASK PENDING TASK COMPLETED TASK DROP DOWN CONTACT US

Employee Performance Evaluation

id	Name	Gender	Dob	Email	Role	Pan No	Aadhar No	Pass No	Finger	Hash	View
1	hari	male	2020-12-07	hari@gmail.com	softwaredeveloper	12	12	12	f9114310fd3c8cc040bc519406bb9ee	Rk1SACAyMAAA8gA	Evaluate

FIG 5.9

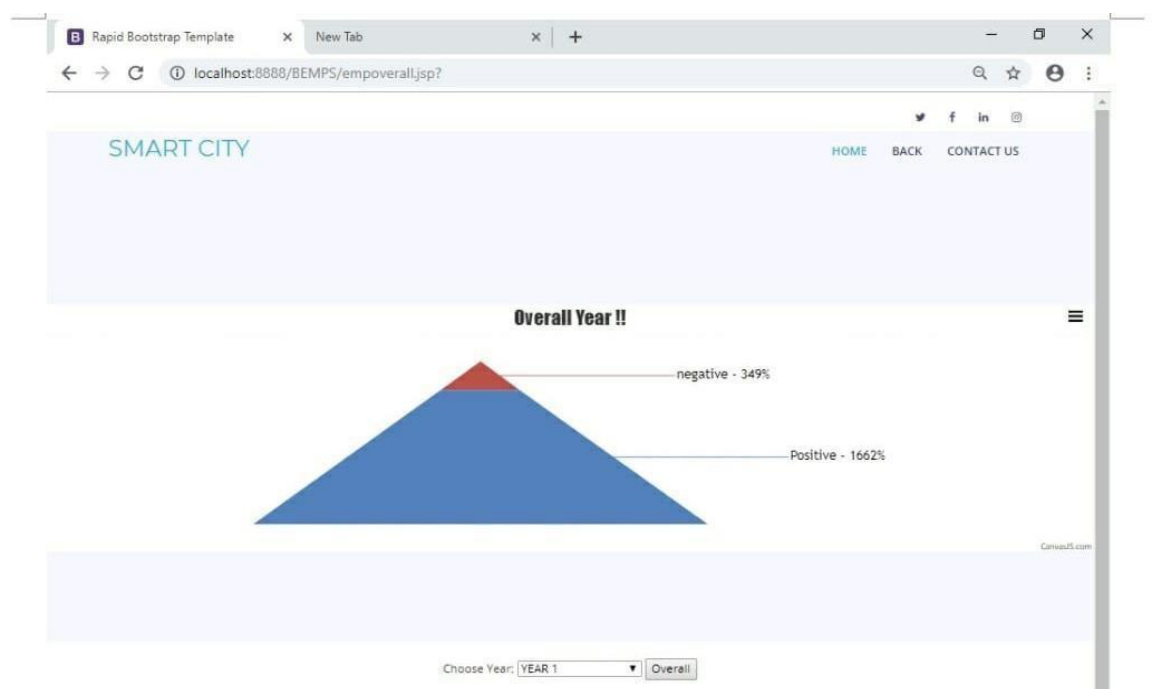


FIG 5.10

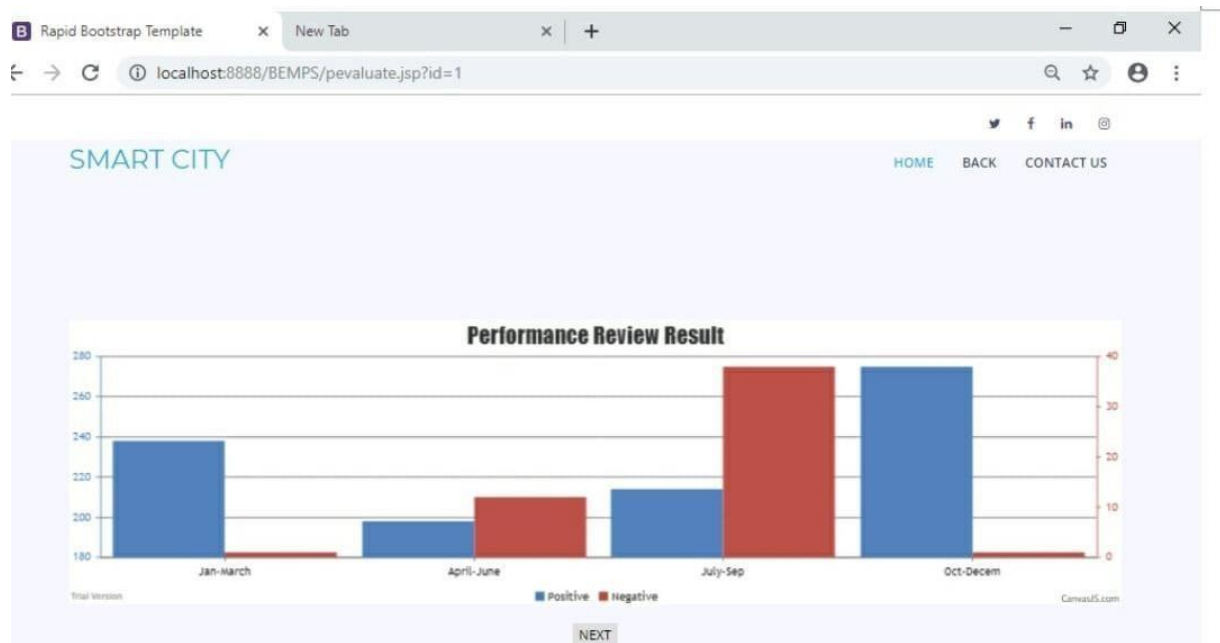


FIG 5.11

In this module an employee can view the task that is assigned to him and give status to the given task and a self evaluation form is provided to get the employee view.

CHAPTER 6

CONCLUSION AND FUTURE

WORK

In this paper, we proposed BEMPAS, an employee performance evaluation system based on blockchain for Smart city governance. The blockchain system is composed of 3-chains referred to as ID-Chain, Behavior Chain and Credit chain that works together to evaluate employee performance. We utilized a game-theoretical approach where a game is played between the Government Institution denoted as Organization and Government Officers denoted as Employees. The organization tries to get more benefits by incentivizing its employees to work harder towards the good of the company. Our Proof of Concept (PoC) result shows that our system is effectively able to motivate the employees in addition to effectively evaluating their performance. The employees feel encouraged by the incentives they receive for achieving organizational objectives. Also, our system achieves trust, transparency, security and accountability among government workers under a Smart City governing environment. Future works might include the use of blockchain in other Smart City areas.

REFERENCES

- [1] Carson, G. Romanelli, P. Walsh, and A. Zhumaev. (2018). Blockchain Beyond the Hype: What is the Strategic Business Value? | McKinsey & Company. [Online]. Available: <https://www.mckinsey.com/businessfunctions/digital-mckinsey/our-insights/blockchain-beyond-the-hype-what-is-the-strategic-business-value>
- [2] D. M. Ni and R. H. Liu, "Study on the enlightenment from the EU smart city evaluation system," in *Applied Mechanics and Materials*, vol. 641. Stafa, Switzerland: Trans Tech, 2014, pp. 624_628.
- [3] H. J. Scholl and M. C. Scholl, "Smart governance: A roadmap for research and practice," in *Proc. IConference*, 2014, pp. 163_176.
- [4] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, . A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *Proc. 45th Hawaii Int. Syst. Sci.*, Jan. 2012, pp. 2289_2297.
- [5] L. Tanti, R. Puspasari, and B. Triandi, "Employee performance assessment with profile matching method," in *Proc. 6th Int. Conf. Cyber IT Service Manage. (CITSM)*, Aug. 2018, pp. _6.
- [6] G. E. Roberts, "Employee performance appraisal system participation: A technique that works," *Public Personnel Manage.*, vol. 32, no. 1, pp. 89_98, Mar. 2003.
- [7] A. M. Awadh and M. S. Alyahya, "Impact of organizational culture on employee performance," *Int. Rev. Manage. Bus. Res.*, vol. 2, no. 1, p. 168, 2013.
- [8] N. Kaur and S. K. Sood, "A game theoretic approach for an IoT-based automated employee performance evaluation," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1385_1394, Sep. 2017.
- 9] R. Intartaglio, "Evaluation of the need for a performance appraisal system for the South Trail Fire Department," *Nat. Fire Acad.*, Fort Myers, FL, USA, *Appl. Res. Rep.*,

2000. [Online]. Available: <http://www.usfa.fema.gov/pdf/efop/efo14655.pdf>

- [10] S. Nakamoto. (Oct. 2018). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [11] M. H. Onik, M. H. Miraz, and C. S. Kim, "A recruitment and human resource management technique using blockchain technology for industry 4.0," in Proc. Smart Cities Symp. (SCS), Manama, Bahrain, 2018, pp. 11_16.
- [12] Q. Xia, E. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, p. 44, Apr. 2017.
- [13] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeD-Share: Trust-medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757_14767, 2017.
- [14] J. Gao, K. O. Asamoah, E. B. Sifah, A. Smahi, Q. Xia, H. Xia, X. Zhang, and G. Dong, "GridMonitoring: Secured sovereign blockchain based monitoring on smart grid," *IEEE Access*, vol. 6, pp. 9917_9925, 2018.
- [15] D. Bhowmik and T. Feng, "The multimedia blockchain: A distributed and tamper-proof media transaction framework," in Proc. 22nd Int. Conf. Digit. Signal Process. (DSP), Aug. 2017, pp. 1_5.
- [16] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst. Cham, Switzerland: Springer, Jun. 2017, pp. 206_220.
- [17] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview blockchain technology: Architecture, consensus, and future trends," in Proc. IEEE Int. Congr. Big Data (BigData Congr.), Jun. 2017, pp. 557_564.
- [18] K. Kuppusamy, N. Shanmugasundaram, E. N. Ganesh, and S. Kumar, "Implementation of performance evaluation of employees through an online system," *Int. J. Manage., Technol. Eng., Delhi, India*, vol. 8, no. 12, p. 5482, 2019.

[19] I. Ahmed, I. Sultana, S. K. Paul, and A. Azeem, "Employee performance evaluation: A fuzzy approach," *Int. J. Productiv. Perform. Manage.*, vol. 62, no. 7, pp. 718_734, Sep. 2013.

[20] Z. Gao, L. Xu, G. Turner, B. Patel, N. Diallo, L. Chen, and W. Shi, "Blockchain-based identity management with mobile device," in *Proc. 1st Workshop Cryptocurrencies Blockchains Distrib. Syst. (CryBlock)*, 2018, pp. 66_70.

[21] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[22] W.-C. Chang, T.-H. Wang, F. H. Lin, and H.-C. Yang, "Game-based learning with ubiquitous technologies," *IEEE Internet Comput.*, vol. 13, no. 4, pp. 26_33, Jul. 2009.

[23] X. Luo, X. Wei, and J. Zhang, "Guided game-based learning using fuzzy cognitive maps," *IEEE Trans. Learn. Technol.*, vol. 3, no. 4, pp. 344_357, Oct. 2010.

[24] M. Hamdi and H. Abie, "Game-based adaptive security in the Internet of Things for eHealth," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 920_925.

[25] Y. Huang, S. Shekhar, and H. Xiong, "Discovering colocation patterns from spatial data: A general approach," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1472_1485, Dec. 2004.

[26] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, and S. Muralidharan, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, p. 30.

[27] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*,

vol. 99, Feb. 1999, pp. 173_186.

- [28] S. E. Haddouti and M. D. E.-C. El Kettani, "Analysis of identity management systems using blockchain technology," in Proc. Int. Conf. Adv. Commun. Technol. Netw. (CommNet), Apr. 2019, pp. 1_7.
- [29] Q. Stokkink and J. Pouwelse, "Deployment of a blockchain-based self-sovereign identity," in Proc. IEEE Int. Conf. Internet Things(iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCCom) IEEE Smart Data (SmartData), Jul. 2018, pp. 1336_1342.
- [30] S.Y. Lim, P. T. Fotsing, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail, "Blockchain technology the identity management and authentication service disruptor: A survey," Int. J. Adv. Sci., Eng. Inf. Technol., vol. 8, nos. 2_4, pp. 1735_1745, 2018.
- [31] M. Takemiya and B. Vanieiev, "Sora identity: Secure, digital identity on the blockchain," in Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC), vol. 2, Jul. 2018, pp. 582_587.
- [32] S. Li, L. D. Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and Internet of Things," IEEE . Ind. Informat., vol. 9, no. 4, pp. 2177_2186, Nov. 2013.

APPENDIX

SOURCE CODE

Employee

```
package com.controller;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import
java.io.ObjectOutputStream;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.sql.SQLException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.dao.InsertEmp;
import com.model.EmpRegm;

/**
 * Servlet implementation class Employee
```

```
*/
```

```
public class Employee extends HttpServlet {
```

```

private static final long serialVersionUID = 1L;

/**
 * @see HttpServlet#HttpServlet()
 */

public Employee() {

    super();

    // TODO Auto-generated constructor stub

}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    try {

        String id = request.getParameter("id");

        String name = request.getParameter("name");

        String gender = request.getParameter("gender");

        String dob = request.getParameter("dob");

        String email = request.getParameter("email");

        String role = request.getParameter("role");

        String panno = request.getParameter("pan");

        String adharno = request.getParameter("aadhar");

        String passno = request.getParameter("passno");

```

```
String fin = request.getParameter("fin");
```

```

String bhash = "bhash";

EmpRegm em = new
EmpRegm(); em.setld(id);

em.setName(name);

em.setGender(gender);

em.setDob(dob);

em.setEmail(email);

em.setRole(role);

em.setPancard(panno);

em.setAadharno(adharno);

em.setPassport(passno);

em.setBhash(bhash);

int status = InsertEmp.ins(em);

if(status > 0)

{

try {

HttpSession session=request.getSession();

session.setAttribute("empid",em.getld());

request.setAttribute("success", "Register Success");

RequestDispatcher
rd=request.getRequestDispatcher("Fingerregister.jsp");

rd.forward(request, response);

} catch (IOException ex) {

ex.printStackTrace();

}

```

```

        System.out.println("Success...");

    }

    else

    {

        request.setAttribute("fail",
        "Data Already Exists Enter
        Details!");

        RequestDispatcher
        rd=request.getRequestDispatcher("smartins2.jsp");

        rd.forward(request, response);

    }

}

catch(Exception e)

{

}

}

```

Orgfeedback

```

package com.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import com.dao.InsertEmpFeed;
```

```
import com.dao.OrgFeedIns;
```

```
import com.model.OrgFeeddBean;
```

```
/**
```

```
 * Servlet implementation class OrganizationFeedBack
```

```
 */
```

```
@WebServlet("/OrganizationFeedBack")
```

```
public class OrganizationFeedBack extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
/**
```

```
 * @see HttpServlet#HttpServlet()
```

```
 */
```

```
public OrganizationFeedBack()
```

```
    { super();
```

```
    // TODO Auto-generated constructor stub
```

```
}
```

```
/**
```

```
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse  
response)
```

```
 */
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {
```

```
String id = request.getParameter("id");
```

```

String year = request.getParameter("year");
String oq1 = request.getParameter("oq1");
String oq2 = request.getParameter("oq2");
String oq3 = request.getParameter("oq3");

OrgFeeddBean orgf = new OrgFeeddBean();
orgf.setld(id);
orgf.setYear(year);
orgf.setOq1(oq1);
orgf.setOq2(oq2);
orgf.setOq3(oq3);

int status = OrgFeedIns.orginsfeed(orgf);
if(status > 0)
{
try {
    request.setAttribute("success", "Org FeedBack Submitted");

    RequestDispatcher
rd=request.getRequestDispatcher("organization.jsp");
    rd.forward(request, response);

} catch (IOException ex) {
    ex.printStackTrace();
}

    System.out.println("Success...");

```

```

    }
    else
    {
        request.setAttribute("fail
        ", "Error Submit
        again!");
        RequestDispatcher
rd=request.getRequestDispatcher("organization.jsp");

        rd.forward(request, response);
    }
}

/**
 * @see HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
 */

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    // TODO Auto-generated method stub

    doGet(request, response);

}

}

```

ManagerFeed

```
package com.controller;
```

```
import java.io.IOException;
```

```

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import

javax.servlet.http.HttpServletResponse;


import com.dao.InsertEmpFeed;

import

com.dao.InsertManagerFeed;

import com.model.EmployeeFeedBack;

import com.model.ManagerFeedBack;

import com.rest.server.BlockchainAdd;


/**
 * Servlet implementation class ManagerFeed
 */

public class ManagerFeed extends HttpServlet {

    private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */

    public ManagerFeed()

    { super();

        // TODO Auto-generated constructor stub

```

}

/**

```
    * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse  
    response)
```

```
    */
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse  
    response) throws ServletException, IOException {
```

```
        String ans ;
```

```
        String          mq1          =  
        request.getParameter("mq1"); String mq2  
        = request.getParameter("mq2"); String  
        mq3    = request.getParameter("mq3");  
        String          mq4          =  
        request.getParameter("mq4"); String mq5  
        = request.getParameter("mq5"); String  
        mq6 = request.getParameter("mq6");
```

```
        BlockchainAdd b = new BlockchainAdd();  
        ans = b.managerfeed(mq1, mq2, mq3, mq4,mq5, mq6);
```

```
        System.out.println("ans====="+ans);
```

```
        if(ans.equalsIgnoreCase("success"))
```

```
        {
```

```
            try {
```

```
                request.setAttribute("success", "FeedBack Submitted");
```

```
                RequestDispatcher
```

```
                rd=request.getRequestDispatcher("managerfeedback.jsp");
```

```
                rd.forward(request, response);
```

```
} catch (IOException ex) {
```

```

        ex.printStackTrace();
    }

    System.out.println("Success...");

}

else

{
    request.setAttribute("fail", "Error Submit
again!"); RequestDispatcher
rd=request.getRequestDispatcher("managerfeedback.jsp");
    rd.forward(request, response);
}

```

Selfevaluate

```

package com.controller;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

```

```
import javax.servlet.http.HttpSession;
```

```
import com.dao.InsertEmpFeed;
```

```
import com.dao.InsertSelfEval;
```

```
import com.dao.SelfEvalTotal;
```

```
import com.mdel.evaluation.SelfPerformanceBean;
```

```
import com.model.SelfEvaluationEmp;
```

```
import com.rest.server.BlockchainAdd;
```

```
/**
```

```
 * Servlet implementation class SelfEvaluate
```

```
 */
```

```
public class SelfEvaluate extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
/**
```

```
 * @see HttpServlet#HttpServlet()
```

```
 */
```

```
public SelfEvaluate() {
```

```
    super();
```

```
    // TODO Auto-generated constructor stub
```

```
}
```

```
/**
```

```
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse  
response)
```

```
*/
```

```
protected void doGet(HttpServletRequest request,  
HttpServletResponse response) throws ServletException, IOException {
```

```
    System.out.println("Self Evaluate procesing");
```

```
    String ans ;
```

```
    try {
```

```
        String q1 =
```

```
        request.getParameter("q1"); String q2
```

```
        = request.getParameter("q2"); String
```

```
        q3 = request.getParameter("q3");
```

```
        String q4 =
```

```
        request.getParameter("q4"); String q5
```

```
        = request.getParameter("q5"); String
```

```
        q6 = request.getParameter("q6"); float
```

```
        pospercentage;
```

```
        float negpercentage;
```

```
        BlockchainAdd b = new BlockchainAdd();
```

```
        ans = b.selfeval(q1, q2, q3, q4,q5,q6);
```

```
        SelfEvaluationEmp se = new SelfEvaluationEmp();
```

```
        se.setQ1(q1);
```

```
        se.setQ2(q2);
```

```
int status = InsertSelfEval.selfeval(se);
```

```

        if(status > 0)
        {
            try {
                request.setAttribute("success", "FeedBack Submitted");

                RequestDispatcher
rd=request.getRequestDispatcher("selfeval.jsp");

                rd.forward(request, response);

            } catch (IOException ex) {
                ex.printStackTrace();
            }

            System.out.println("Success...");

        }
        else
        {
            request.setAttribute("fail", "Error Submit
again!"); RequestDispatcher
rd=request.getRequestDispatcher("selfeval.jsp");

            rd.forward(request, response);

        }
    }
    catch(Exception e)
    {

    }
}

```

```

    }

    /**
     *      @see      HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
     */

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        // TODO Auto-generated method stub

        doGet(request, response);

    }

}

```

Survey.java

```

package com.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

import com.rest.server.BlockchainAdd;

```

/**

```

* Servlet implementation class Survey

*/

@WebServlet("/Survey")

public class Survey extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Survey() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

        String ans ;

        String          mq1          =
        request.getParameter("mq1"); String mq2
        = request.getParameter("mq2"); String
        mq3    = request.getParameter("mq3");
        String          mq4          =

```

```
request.getParameter("mq4"); String mq5  
= request.getParameter("mq5"); String  
mq6 = request.getParameter("mq6");
```

```

BlockchainAdd b = new BlockchainAdd();

ans = b.sur(mq1, mq2, mq3, mq4,mq5, mq6);


System.out.println("ans====="+ans);


        if(ans.equalsIgnoreCase("success"))
        {
            try {
                request.setAttribute("success", "Survey Submitted");

                RequestDispatcher
rd=request.getRequestDispatcher("empanemp.jsp");

                rd.forward(request, response);

            } catch (IOException ex) {
                ex.printStackTrace();
            }

            System.out.println("Success...");

        }
        else
        {
            request.setAttribute("fail", "Error Submit
again!"); RequestDispatcher
rd=request.getRequestDispatcher("empanemp.jsp");

            rd.forward(request, response);

        }

```

```
/**
 *      @see      HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
 */

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    // TODO Auto-generated method stub

    doGet(request, response);
}
```

```

/*
    * ManagerFeedBack mfb = new ManagerFeedBack(); mfb.setId(id);
mfb.setQ1(q1);
    * mfb.setQ2(q2);
    *
    *
    *
    *
    *
    *
    *
    *
    *
    * int status = InsertManagerFeed.insmanfeed(mfb); if(status > 0) { try {

```

```

        * request.setAttribute("success", "FeedBack Submitted");
RequestDispatcher

        * rd=request.getRequestDispatcher("managerfeedback.jsp");
rd.forward(request,

        * response);

        *

        * BlockchainAdd b = new BlockchainAdd(); // ans =
b.selfeval(mfb.getId(),

        * mfb.getQ1(), m, q4,q5);

        *

        * } catch (IOException ex) { ex.printStackTrace(); }

        *

        * System.out.println("Success...");

        *

        * } else { request.setAttribute("fail", "Error Submit again!");

        *

RequestDispatcher
rd=request.getRequestDispatcher("managerfeedback.jsp");

        * rd.forward(request, response); }

        *

        *

        *

        *

        *

        */

```

Sentiment classification

```
package com.stanford_nlp.model;
```

```
public class SentimentClassification {

    double veryPositive;

    double positive;

    double neutral;

    double negative;

    double veryNegative;


    public double getVeryPositive() {

        return veryPositive;

    }

    public void setVeryPositive(double veryPositive) {

        this.veryPositive = veryPositive;

    }

    public double getPositive() {

        return positive;

    }

    public void setPositive(double positive) {

        this.positive = positive;

    }

    public double getNeutral()

        { return neutral;

    }

    public void setNeutral(double neutral) {
```

```
this.neutral = neutral;
```

```
}
```

```

    public double getNegative() {
        return negative;
    }

    public void setNegative(double negative)
    { this.negative = negative;
    }

    public double getVeryNegative() {
        return veryNegative;
    }

    public void setVeryNegative(double veryNegative) {
        this.veryNegative = veryNegative;
    }
}

```

Sentiment analyzer

```

package com.stanford_nlp.SentimentAnalyzer;

import java.util.Properties;

import org.ejml.simple.SimpleMatrix;

import
com.stanford_nlp.model.SentimentClassification;
import com.stanford_nlp.model.SentimentResult;

```

```
import edu.stanford.nlp.ling.CoreAnnotations;
```

```
import
edu.stanford.nlp.neural.rnn.RNNCoreAnnotations;

import edu.stanford.nlp.pipeline.Annotation;

import edu.stanford.nlp.pipeline.StanfordCoreNLP;

import edu.stanford.nlp.sentiment.SentimentCoreAnnotations;

import edu.stanford.nlp.trees.Tree;

import edu.stanford.nlp.util.CoreMap;


public class SentimentAnalyzer {

    /*
     * "Very negative" = 0 "Negative" = 1 "Neutral" = 2 "Positive" = 3
     * "Very positive" = 4
     */

    static Properties props;

    static StanfordCoreNLP pipeline;

    public void initialize() {

        // creates a StanfordCoreNLP object, with POS tagging, lemmatization,
        NER, parsing, and sentiment

        props = new Properties();

        props.setProperty("annotators", "tokenize, ssplit, parse, sentiment");

        pipeline = new StanfordCoreNLP(props);

    }
```

```
public SentimentResult getSentimentResult(String text) {
```

```

        SentimentResult sentimentResult = new SentimentResult();

        SentimentClassification sentimentClass = new
SentimentClassification();

        if (text != null && text.length() > 0) {

            // run all Annotators on the text

            Annotation annotation = pipeline.process(text);

            for (CoreMap sentence :
annotation.get(CoreAnnotations.SentencesAnnotation.class)) {

                // this is the parse tree of the current sentence

                Tree tree =
sentence.get(SentimentCoreAnnotations.SentimentAnnotatedTree.class);

                SimpleMatrix sm =
RNNCoreAnnotations.getPredictions(tree);

                String sentimentType =
sentence.get(SentimentCoreAnnotations.SentimentClass.class);

                sentimentClass.setVeryPositive((double)Math.round(sm.get(4) * 100d));
                sentimentClass.setPositive((double)Math.round(sm.get(3)
* 100d));

                sentimentClass.setNeutral((double)
Math.round(sm.get(2)
* 100d));

                sentimentClass.setNegative((double)Math.round(sm.get(1) * 100d));

                sentimentClass.setVeryNegative((double)Math.round(sm.get(0) * 100d));

```

```

tree));
sentimentResult.setSentimentScore(RNNCoreAn
notations.getPredictedClass(

sentimentResult.setSentime
ntType(sentimentType);
sentimentResult.setSentime
ntClass(sentimentClass);

    }

    }
    return sentimentResult;
}

}

```

Blockchain

```

package com.rest.server;

import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;

//import com.sun.xml.bind.v2.runtime.property.Property;

public class BlockChain {

//    public static String Store = "http://10.0.0.42:5000/transactions/new";

```

```
// public static String Mine = "http://10.0.0.42:5000/mine";  
// public static String Retrive =  
    "http://10.0.0.42:5000/chain"; public static String Store =  
    "transactions/new";
```

```
public static String Mine = "mine";

public static String Retrive = "chain";


public static String addTransaction(String jsonInput, String urlLink) {
    try {
        URL url = new URL(urlLink);

        HttpURLConnection httpCon = (HttpURLConnection) url
            .openConnection();

        httpCon.setRequestProperty("Content-Type", "application/json");
        httpCon.setRequestMethod("POST");
        httpCon.setDoInput(true);
        httpCon.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(httpCon
            .getOutputStream());

        out.write(jsonInput);
        out.close();

        System.out.println(httpCon.getResponseCode());
        System.out.println(httpCon.getResponseMessage());

        InputStreamReader ir = new InputStreamReader(httpCon
            .getInputStream());

        char b[] = new char[1024];
        int l = 0;

        while (ir.ready()) {
            l = ir.read(b);

            System.out.println(new String(b, 0, l));
        }
    }
}
```

}

```

    } catch (Exception e) {
        return "no";
    }
    return "ok";
}

```

```

public static void mineChain(String urlLink)
{
    try {
        URL url = new URL(urlLink);
        HttpURLConnection httpCon = (HttpURLConnection) url
            .openConnection();
        httpCon.setDoOutput(false);
        httpCon.setRequestMethod("GET");
        System.out.println(httpCon.getResponseCode());
        System.out.println(httpCon.getResponseMessage());
        InputStreamReader ir = new InputStreamReader(httpCon
            .getInputStream());
        char b[] = new char[1024];
        int l = 0;
        while (ir.ready()) {
            l = ir.read(b);
            System.out.println(new String(b, 0, l));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

}

```
}
```

Blockchain add

```
package com.rest.server;
```

```
import org.json.simple.JSONObject;
```

```
public class BlockchainAdd {
```

```
    String s = "success";
```

```
    String j;
```

```
    String sq1;
```

```
    String sq2;
```

```
    String sq3;
```

```
    String sq4;
```

```
    String sq5;
```

```
    String sq6;
```

```
    /*
```

```
    * String j ;
```

```
    *
```

```
    * public BlockchainAdd(String q1 ,String q2,String q3 ,String q4) { this.sq1 =
```

```
    * q1; this.sq2 = q2; this.sq3 = q3; this.sq4 = q4;
```



```

        * System.out.println("SQ1======" + sq1); }

        */

public String selfeval(String sq1 ,String sq2,String sq3 ,String sq4,String sq5,String
sq6 )
{

    System.out.println("=====" + sq1);

    System.out.println("=====" + sq2);

    System.out.println("=====" + sq3);

    System.out.println("=====" + sq4);


    JSONObject jsonObject = new
    JSONObject(); jsonObject.put("sq1", sq1);

    jsonObject.put("sq2", sq2);

    jsonObject.put("sq3", sq3);

    jsonObject.put("sq4", sq4);

    jsonObject.put("sq5", sq5);

    jsonObject.put("sq6", sq6);

    j = jsonObject.toJSONString();

    System.out.println("

    "+j);

    Blockchain call = new Blockchain();

    call.addTransaction(j,"http://localhost:5000/"+Blockchain.Store);

    Blockchain.mineChain("http://localhost:5000/"+Blockchain.Mine);


    return s;

```

```
}
```

```
public String managerfeed(String sq1 ,String sq2,String sq3 ,String sq4,String  
sq5,String sq6 )
```

```
{
```

```
    System.out.println("====="+sq1);
```

```
    System.out.println("====="+sq2);
```

```
    System.out.println("====="+sq3);
```

```
    System.out.println("====="+sq4);
```

```
    JSONObject jsonObject = new
```

```
    JSONObject(); jsonObject.put("sq1", sq1);
```

```
    jsonObject.put("sq2", sq2);
```

```
    jsonObject.put("sq3", sq3);
```

```
    jsonObject.put("sq4", sq4);
```

```
    jsonObject.put("sq5", sq5);
```

```
    jsonObject.put("sq6", sq6);
```

```
    j = jsonObject.toJSONString();
```

```
    System.out.println("
```

```
    "+j);
```

```
    Blockchain call = new Blockchain();
```

```
    call.addTransaction(j,"http://localhost:5000/"+Blockchain.Store);
```

```
    Blockchain.mineChain("http://localhost:5000/"+Blockchain.Mine);
```

```
    return s;
```

```
}
```

```

public String attend(String sq1 ,String sq2,String sq3 ,String sq4,String sq5,String sq6
)
{

    System.out.println("====="+sq1);
    System.out.println("====="+sq2);
    System.out.println("====="+sq3);
    System.out.println("====="+sq4);


    JSONObject jsonObject = new
    JSONObject(); jsonObject.put("sq1", sq1);
    jsonObject.put("sq2", sq2);
    jsonObject.put("sq3", sq3);
    jsonObject.put("sq4", sq4);
    jsonObject.put("sq5", sq5);
    jsonObject.put("sq6", sq6);
    j = jsonObject.toJSONString();
    System.out.println("
    "+j);

    Blockchain call = new Blockchain();
    call.addTransaction(j,"http://localhost:5000/"+Blockchain.Store);
    Blockchain.mineChain("http://localhost:5000/"+Blockchain.Mine);

    return s;

}

public String sur(String sq1 ,String sq2,String sq3 ,String sq4,String sq5,String sq6 )

```

{

```
System.out.println("===="+sq1);  
System.out.println("===="+sq2);  
System.out.println("===="+sq3);  
System.out.println("===="+sq4);
```

```
JSONObject jsonObject = new  
JSONObject(); jsonObject.put("sq1", sq1);  
jsonObject.put("sq2", sq2);  
jsonObject.put("sq3", sq3);  
jsonObject.put("sq4", sq4);  
jsonObject.put("sq5", sq5);  
jsonObject.put("sq6", sq6);  
j = jsonObject.toJSONString();  
System.out.println("  
"+j);  
Blockchain call = new Blockchain();  
call.addTransaction(j,"http://localhost:5000/"+Blockchain.Store);  
Blockchain.mineChain("http://localhost:5000/"+Blockchain.Mine);  
  
return s;  
  
}  
  
}  
  
}
```

/**

```

        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
        }
    }
}

```

Attendance

```

package com.controller;

import java.io.IOException;
import
java.sql.Connection;
import java.sql.PreparedStatement;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import

```

```
javax.servlet.http.HttpServletResponse;
```

```
import com.dao.ConnectionProvider;
```

```
import com.rest.server.BlockchainAdd;
```

```

/**
 * Servlet implementation class Attendance
 */
@WebServlet("/Attendance")
public class Attendance extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Attendance() {

        super();

        // TODO Auto-generated constructor stub

    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

        int

        status=0;

        String ans ;

        String          mq1          =

        request.getParameter("mq1"); String mq2

```

```
= request.getParameter("mq2"); String  
mq3 = request.getParameter("mq3");  
String mq4 =  
request.getParameter("mq4");
```

```
String mq5 = request.getParameter("mq5");
```

```
String mq6 = request.getParameter("mq6");
```

```
BlockchainAdd b = new BlockchainAdd();
```

```
ans = b.attend(mq1, mq2, mq3, mq4, mq5, mq6);
```

```
System.out.println("ans====="+ans);
```

```
//adding attendence in db
```

```
try {
```

```
    Connection
```

```
    con=ConnectionProvider.getCon();
```

```
    System.out.println("Inserting self evaluate");
```

```
    PreparedStatement
```

```
    ps=con.prepareStatement("insert into
```

```
    ps.setStri
```

```
    ng(1,
```

```
    mq3);
```

```
    ps.setStri
```

```
    ng(2,
```

```
    mq4);
```

```
    ps.setStri
```

```
    ng(3,
```

```
    mq5);
```

```
    ps.setStri
```

```
attend values (?, ?, ?, ?)");
```

```

ng(4,                                .executeU
mq6);                                pdate());
status=ps

        System.out.println(status+"    insert success attendance");

    }
catch(Exception e){

    e.printStackTrace();
}

    if(ans.equalsIgnoreCase("success"))
    {

```

```

        try {
            request.setAttribute("success", "Attendance Submitted");

            RequestDispatcher
rd=request.getRequestDispatcher("attendance.jsp");

            rd.forward(request, response);

        } catch (IOException ex) {
            ex.printStackTrace();
        }

        System.out.println("Success...");

    }

    else

    {
        request.setAttribute("fail", "Error Submit
again!"); RequestDispatcher
rd=request.getRequestDispatcher("attendance.jsp");

        rd.forward(request, response);

    }

}

/**
 *      @see      HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
 */

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    // TODO Auto-generated method stub

    doGet(request, response);

```

```
    }

}

}

/**
 *      @see      HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
 */

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    // TODO Auto-generated method stub

    doGet(request, response);

}

}
```