**DocFace+: ID Document to Selfie\* Matching**

A PROJECT REPORT

in the partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

MAY 2020

**BONAFIDE CERTIFICATE**

**ABSTRACT**

Numerous activities in our daily life require us to verify who we are by showing our ID documents containing face images, such as passports and driver licenses, to human operators. However, this process is slow, labor intensive and unreliable. As such, an automated system for matching ID document photographs to live face images (selfies1) in real time and with high accuracy is required. In this paper, we propose DocFace+ to meet this objective. We first show that gradient-based optimization methods converge slowly (due to the underfitting of classifier weights) when many classes have very few samples, a characteristic of existing ID-selfie datasets. To overcome this shortcoming, we propose a method, called dynamic weight imprinting, to update the classifier weights, which allows faster convergence and more generalizable representations. Next, a pair of sibling networks with partially shared parameters are trained to learn a unified face representation with domain-specific parameters. Cross-validation on an ID-selfie dataset shows that while a publicly available general face matcher (InsightFace) only achieves a true accept rate (TAR) of 88.78 ± 1.30% at a false accept

rate of 0.01% on the problem, DocFace+ improves the TAR to 95.95 ± 0.54%.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No** | **TITLE** | **PAGE** |
|  | LIST OF FIGURES | Vi |
|  | LIST OF ABBREVATIONS | Vii |
| **1** | **INTRODUCTION** |  |
| 1.1 | About the Project |  |
| **2** | **SYSTEM ANALYSIS** |  |
| 2.1 | Existing system |  |
| 2.2 | Proposed system |  |
| **3** | **REQUIREMENTS SPECIFICATION** |  |
| 3.1 | Introduction |  |
| 3.2 | Hardware and Software specification |  |
| 3.3 | Technologies Used |  |
| 3.3.1 | Java |  |
| 3.3.3.1 | Introduction to Java |  |
| 3.3.1.2 | Working of Java |  |
| **4** | **Project Purpose and Scope** |  |
| 4.1 | Purpose |  |
| 4.2 | Project Scope |  |
| 4.3 | Product Perspective |  |
| 4.4 | System Features |  |
| 4.5 | Design and Implementation Constraints |  |
| 4.6 | Other Nonfunctional Requirements |  |
| **5** | **SYSTEM DESIGN** |  |
| 5.1 | Architecture Diagram |  |
| **6** | **SYSTEM DESIGN – DETAILED** |  |
| 6.1 | Modules |  |
| 6.2 | Module explanation |  |
| **7** | **CODING AND TESTING** |  |
| 7.1 | Coding |  |
| 7.2 | Coding standards |  |
| 7.3 | Test procedure |  |
| 7.4 | Test data and output |  |
|  | **SOURCE CODE** |  |
|  | **SNAP SHOTS** |  |
|  | **REFERENCES** |  |

**LIST OF FIGURES**

5.1 System Design

5.2 Sequence Diagram

5.3 Use Case Diagram

5.4 Activity Diagram

5.5 Collaboration Diagram:

5.6 Data Flow Diagram

5.7 Class Diagram

**LIST OF ABBREVATIONS**

JDK Java Development Toolkit

DEX Dalvik Executables

TCP Transmission Control Protocol

IP Internet Protocol

**HTTP** Hyper Text Transfer Protocol

**ADT** Android Development Tool

**CHAPTER 1**

**INTRODUCTION**

**Aim:**

The main aim of this project is to solve the problem of counterfeiting certificates we are proposing an digital certificate system based on block chain technology and to verify the traveler’s identity using live camera, which allows faster convergence and more generalizable representations.

**Synopsis:**

Numerous activities in our daily life require us to verify who we are by showing our ID documents containing face images, such as passports and driver licenses, to human operators. However, this process is slow, labor intensive and unreliable. As such, an automated system for matching ID document photos to live face images (selfies) in real time and with high accuracy is required. In this paper, we propose DocFace+ to meet this objective. We first show that gradient-based optimization methods converge slowly (due to the underfitting of classifier weights) when many classes have very few samples, a characteristic of existing ID-selfie datasets. To overcome this shortcoming, to update the classifier weights, which allows faster convergence and more generalizable representations. Next, a pair of sibling networks with partially shared parameters are trained to learn a unified face representation with domain-specific parameters. Cross-validation on an ID selfie dataset shows that while a publicly available general face matcher.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In the existing system, Identity verification plays an important role in our daily lives. For example, access control, physical security and international border crossing require us to verify our access (security) level and our identities. to verify who we are by showing our ID documents containing face images, such as passports and driver licenses, to human operators. However, this process is slow, labor intensive and unreliable. As such, an automated system for matching ID document photos to live face images (selfies) in real time and with high accuracy is required. After verifying a traveler’s identity by face comparison, the gate is automatically opened for the traveler to enter. For IDselfie matching, they are comparing a scanned or digital document photo.

**2.1.1Problem Statement:**

* The problem of ID-selfie matching poses numerous challenges that are different from general face recognition. For typical unconstrained face recognition tasks, the main challenges are due to pose, illumination and expression (PIE) variations.
* The low quality of document photos due to image compression1 and (2) the large time gap between the document issue date and the verification date remain as the primary difficulties.

**2.2PROPOSED SYSTEM**

We are proposing a certificate system based on blockchain to overcome the problem. Data are stored in different nodes, and anyone who wishes to modify a particular internal datum must request that other nodes modify it simultaneously. Thus, the system is highly reliable. We developed a decentralized application and designed a certificate system based on Ethereum blockchain. This technology was selected because it is incorruptible, encrypted, and trackable and permits data synchronization. By integrating the features of blockchain, the system improves the efficiency operations at each stage. The system saves on paper, cuts management costs, prevents document forgery, and provides accurate and reliable information on digital certificates and compare user live face with verified document face.

**CHAPTER 3**

**REQUIREMENT SPECIFICATIONS**

**3.1 INTRODUCTION**

**I**DENTITY verification plays an important role in our daily lives. For example, access control, physical security and international border crossing require us to verify our access (security) level and our identities. A practical and common approach to this problem involves comparing an individual’s live face to the face image found in his/her ID document. For example, immigration and customs officials look at the passport photo to confirm a traveler’s identity. Clerks at supermarkets in the United States look at the customer’s face and driver license to check his/her age when the customer is purchasing alcohol. Instances of ID document photo matching can be found in numerous scenarios. However, it is primarily conducted by humans manually, which is time consuming, costly, and prone to operator errors. A study pertaining to the passport officers in Sydney, Australia, shows that even the trained officers perform poorly in matching unfamiliar faces to passport photos, with a 14% false acceptance rate [1]. Therefore, an accurate and automated system for efficient matching of ID document photos to selfies\* is required. In addition, automated ID-selfie matching systems also enable remote authentication applications that are otherwise not feasible, such as onboarding new customers in a mobile app (by verifying their identities for account creation), or account recovery in the case of forgotten passwords. One application scenario of our ID-selfie matching system (DocFace+) is illustrated in Figure 1.

A number of automated ID-selfie matching systems have been deployed at international border crossings. Deployed in 2007, SmartGate [2] in Australia (See Figure 2) is the earliest of its kind. Due to an increasing number of travelers to Australia, the Australian government introduced SmartGate at most of its international airports as an electronic passport check for ePassport holders. To use the SmartGate, travelers only need to let a machine read their ePassport chips containing their digital photos and then capture their face images using a camera mounted at the SmartGate. After verifying a traveler’s identity by face comparison, the gate is automatically opened for the traveler to enter Australia. Similar machines have also been installed in the U.K. (ePassport gates) [3], USA (U.S. Automated Passport Control) [4] and other countries. In

China, such verification systems have been deployed at various locations, including train stations, for matching Chinese ID cards with live faces [5]. In addition to international border

control, some businesses [6], [7] are utilizing face recognition solutions to ID document verification for online services. The problem of ID-selfie matching poses numerous challenges

that are different from general face recognition. For typical unconstrained face recognition tasks, the main challenges are due to pose, illumination and expression (PIE) variations. On the other hand, in ID-selfie matching, we are comparing a scanned or digital document photo to a digital camera photo of a live face. Assuming that the user is cooperative, both of the images are captured under constrained conditions and large PIE variations would not be present. However, (1) the low quality of document photos due to image compression2 and (2) the large time gap between the document issue date and the verification date remain as the primary difficulties

(See Figure 3). In addition, since state-of-the-art face recognition systems are based on deep networks, another issue faced in our problem is the lack of a large training dataset (pairs of ID photos and selfies).

**3.2 HARDWARE AND SOFTWARE SPECIFICATION**

**3.2.1 HARDWARE REQUIREMENTS**

* Hard Disk : 80GB and Above
* RAM : 4GB and Above
* Processor : P IV and Above

**3.2.2 SOFTWARE REQUIREMENTS**

* Windows 7 and above
* JDK 1.7
* J2EE
* Tomcat 7.0
* MySQL

**3.3 TECHNOLOGIES USED**

* + - J2EE (JSP, Servlet), JavaScript, HTML, CSS, AJAX.
    - Hibernate Framework
    - MVC Pattern
    - Design Pattern

3.3.1 **Algorithm Used:**

* KNN
* RSA
* SHA-256

**K-Nearest Neighbors**

**3.3.1.1 Introduction**

In four years of 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 of a bias towards classification models is that most analytical problem involves 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 effect the output/prediction.

## 3.3.1.2 KNN algorithm

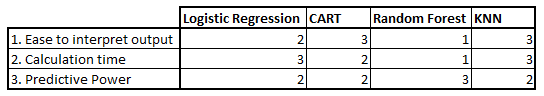
KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

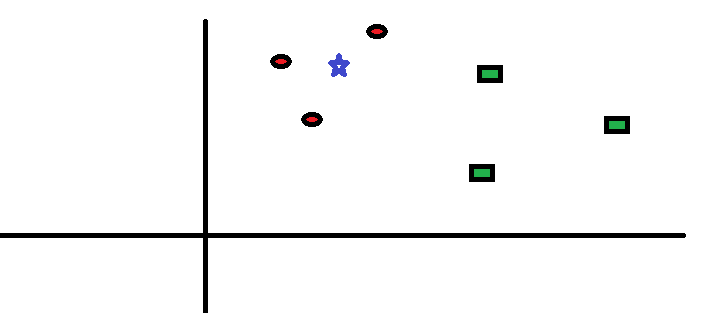
Let us take a few examples to place KNN in the scale:



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

## 3.3.1.3 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):



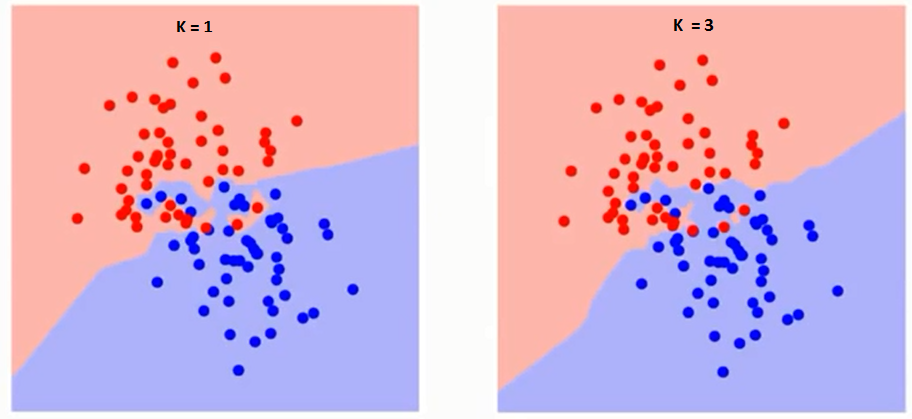
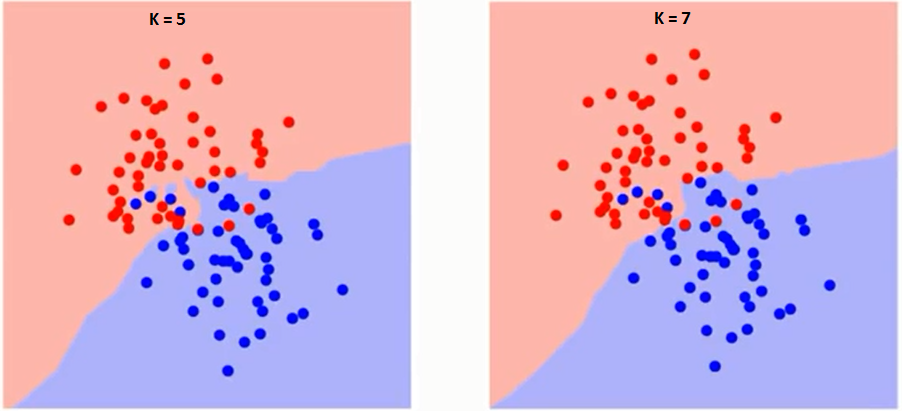
You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” is KNN algorithm is the nearest neighbors we wish to take vote from. Let’s say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three data points on the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



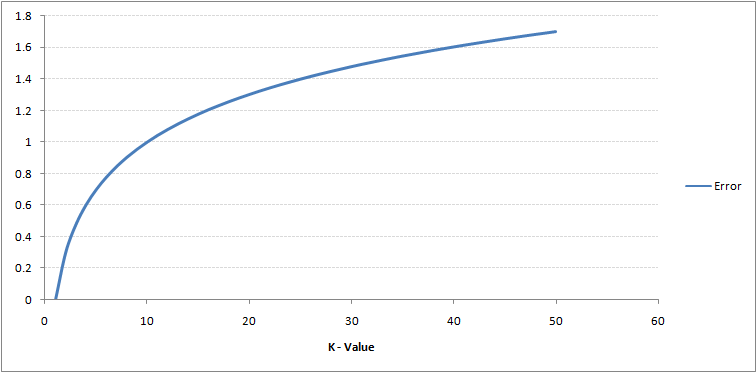
The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

## 3.3.1.4 How do we choose the factor K?

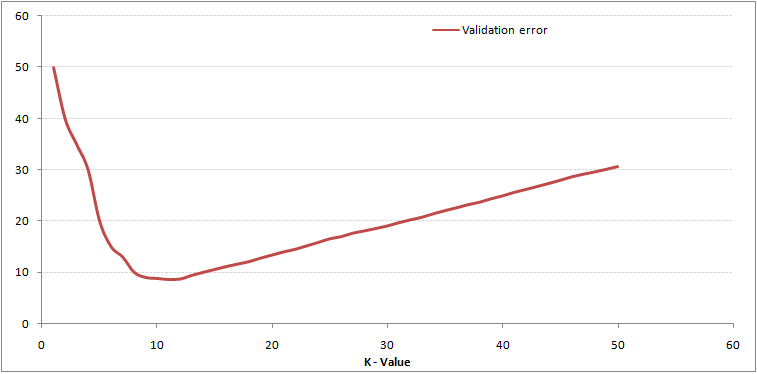
First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let’s try to see the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

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-value. Following is the curve for the training error rate with varying value of K:



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 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:



This makes the story more clear. At K=1, we were over fitting the boundaries. Hence, error rate initially decreases and reaches a minimal. After the minima 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.

**3.3.1.5 Breaking it down – Pseudo Code of KNN**

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

1. Load the data
2. Initialize the value of k
3. 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

## 3.3.1.6 Conclusion

KNN algorithm is one of the simplest classification algorithms. Even with such simplicity, it can give highly competitive results. KNN algorithm 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.

**3.3.2 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.

**3.3.2.1 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.

**3.3.2.2 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

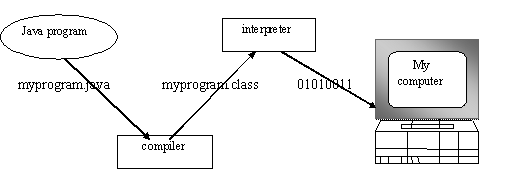
**3.3.2.3 THE JAVA PROGRAMMING LANGUAGE**

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 and 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 co implied 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.3.1**

You can think of Java byte codes 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.

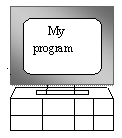
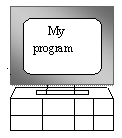
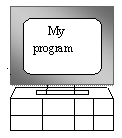
You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can e run on Windows NT, Solaris and Macintos

Complier

Interpreter

Interpreter

Interpreter

**  **

**PC-Compatible Sun Ultra Solaris Power macintosh**

**Windows NT System 8**

**3.3.2.4 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.

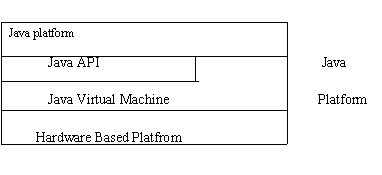
****

Fig.3.3

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

**3.3.3 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 into 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 paring

**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.

3.3.4 **Machine learning Introduction**:

Machine learning (ML) is the [scientific study](https://en.wikipedia.org/wiki/Branches_of_science) of [algorithms](https://en.wikipedia.org/wiki/Algorithm) and [statistical models](https://en.wikipedia.org/wiki/Statistical_model) that [computer systems](https://en.wikipedia.org/wiki/Computer_systems) use to perform a specific task without using explicit instructions, relying on patterns and [inference](https://en.wikipedia.org/wiki/Inference) instead. It is seen as a subset of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning algorithms build a [mathematical model](https://en.wikipedia.org/wiki/Mathematical_model) based on sample data, known as "[training data](https://en.wikipedia.org/wiki/Training_data)", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as [email filtering](https://en.wikipedia.org/wiki/Email_filtering) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.

Machine learning is closely related to [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics), which focuses on making predictions using computers. The study of [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization) delivers methods, theory and application domains to the field of machine learning. [Data mining](https://en.wikipedia.org/wiki/Data_mining) is a field of study within machine learning, and focuses on [exploratory data analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis) through learning. In its application across business problems, machine learning is also referred to as [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics).

**Machine learning tasks:**

Machine learning tasks are classified into several broad categories. In [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning), the algorithm builds a [mathematical model](https://en.wikipedia.org/wiki/Mathematical_model) from a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the [training data](https://en.wikipedia.org/wiki/Training_data) for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

[Classification](https://en.wikipedia.org/wiki/Statistical_classification) algorithms and [regression](https://en.wikipedia.org/wiki/Regression_analysis) algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a [limited set](https://en.wikipedia.org/wiki/Discrete_number) of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "[spam](https://en.wikipedia.org/wiki/Email_spam)" or "not spam", represented by the [Boolean](https://en.wikipedia.org/wiki/Boolean_data_type) values true and false. [Regression](https://en.wikipedia.org/wiki/Regression_analysis) algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

In [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning), the algorithm builds a mathematical model from a set of data that contains only inputs and no desired output labels. Unsupervised learning algorithms are used to find structure in the data, like grouping or [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in [feature learning](https://en.wikipedia.org/wiki/Feature_learning). [Dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction) is the process of reducing the number of "[features](https://en.wikipedia.org/wiki/Feature_(machine_learning))", or inputs, in a set of data.

[Active learning](https://en.wikipedia.org/wiki/Active_learning_(machine_learning)) algorithms access the desired outputs (training labels) for a limited set of inputs based on a budget and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling. [Reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning) algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment and are used in [autonomous vehicles](https://en.wikipedia.org/wiki/Autonomous_vehicle) or in learning to play a game against a human opponent. Other specialized algorithms in machine learning include [topic modeling](https://en.wikipedia.org/wiki/Topic_modeling), where the computer program is given a set of [natural language](https://en.wikipedia.org/wiki/Natural_language) documents and finds other documents that cover similar topics. Machine learning algorithms can be used to find the unobservable [probability density function](https://en.wikipedia.org/wiki/Probability_density_function) in [density estimation](https://en.wikipedia.org/wiki/Density_estimation) problems. [Meta learning](https://en.wikipedia.org/wiki/Meta_learning_(computer_science)) algorithms learn their own [inductive bias](https://en.wikipedia.org/wiki/Inductive_bias) based on previous experience. In [developmental robotics](https://en.wikipedia.org/wiki/Developmental_robotics), [robot learning](https://en.wikipedia.org/wiki/Robot_learning) algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

### Types of learning algorithms:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

#### Supervised learning:

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as [training data](https://en.wikipedia.org/wiki/Training_data), and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an [array](https://en.wikipedia.org/wiki/Array_data_structure) or vector, sometimes called a feature vector, and the training data is represented by a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)). Through iterative optimization of an [objective function](https://en.wikipedia.org/wiki/Loss_function), supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. [Similarity learning](https://en.wikipedia.org/wiki/Similarity_learning) is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in [ranking](https://en.wikipedia.org/wiki/Ranking), [recommendation systems](https://en.wikipedia.org/wiki/Recommendation_systems), visual identity tracking, face verification, and speaker verification.

In the case of [semi-supervised](https://en.wikipedia.org/wiki/Semi-supervised_learning) learning algorithms, some of the training examples are missing training labels, but they can nevertheless be used to improve the quality of a model. In [weakly supervised learning](https://en.wikipedia.org/wiki/Weak_supervision), the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

#### Unsupervised learning:

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of [density estimation](https://en.wikipedia.org/wiki/Density_estimation) in [statistics](https://en.wikipedia.org/wiki/Statistics), though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more pre designated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated, for example, by *internal compactness*, or the similarity between members of the same cluster, and *separation*, the difference between clusters. Other methods are based on *estimated density* and *graph connectivity*.

**Semi-supervised learning:**

Semi-supervised learning falls between [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) (without any labeled training data) and [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

**CHAPTER 4**

**4.1 Project Purpose and Scope**

**4.1 Purpose**

The main aim of this project is to solve the problem of counterfeiting certificates we are proposing an digital certificate system based on block chain technology and to verify the traveler’s identity using live camera, which allows faster convergence and more generalizable representations.

**4.2 Project Scope**

We are proposing a certificate system based on blockchain to overcome the problem. Data are stored in different nodes, and anyone who wishes to modify a particular internal datum must request that other nodes modify it simultaneously. Thus, the system is highly reliable. We developed a decentralized application and designed a certificate system based on Ethereum blockchain. This technology was selected because it is incorruptible, encrypted, and trackable and permits data synchronization. By integrating the features of blockchain, the system improves the efficiency operations at each stage. The system saves on paper, cuts management costs, prevents document forgery, and provides accurate and reliable information on digital certificates and compare user live face with verified document face.

* 1. **4.3 Product Perspective**

Numerous activities in our daily life require us to verify who we are by showing our ID documents containing face images, such as passports and driver licenses, to human operators. However, this process is slow, labor intensive and unreliable. As such, an automated system for matching ID document photos to live face images (selfies) in real time and with high accuracy is required. In this paper, we propose DocFace+ to meet this objective. We first show that gradient-based optimization methods converge slowly (due to the underfitting of classifier weights) when many classes have very few samples, a characteristic of existing ID-selfie datasets. To overcome this shortcoming, to update the classifier weights, which allows faster convergence and more generalizable representations. Next, a pair of sibling networks with partially shared parameters are trained to learn a unified face representation with domain-specific parameters. Cross-validation on an ID selfie dataset shows that while a publicly available general face matcher.

* 1. **4.4 SystemFeatures**

A new optimization method for classification-based embedding learning on shallow datasets3. A new recognition system containing a pair of partially shared networks for learning unified representations from IDselfie pairs. An evaluation of COTS and public-domain face matchers showing ID-selfie matching is a non-trivial problem with different challenges from general face matching.

**4.5 Design and Implementation Constraints**

**4.5.1 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

**4.5.2 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

**4.5.3 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.

* 1. **Other Nonfunctional Requirements**

**4.6.1 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.

**4.6.2 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 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.

**CHAPTER 5**

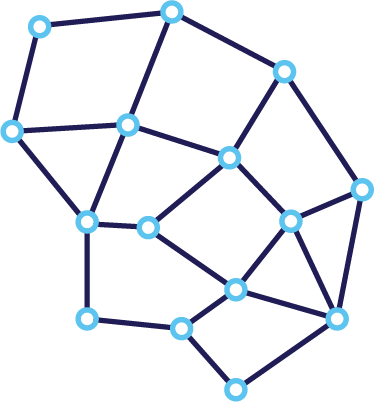
**5.1 Architecture Diagram:**



User Application



Central Board Server



Registration

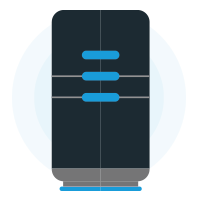
& Request

Approval

Request &

Response

Block chain



E.C.S

Upload Certificates

Upload

Certificates

Verifying

Authority

Face verification

QR Code

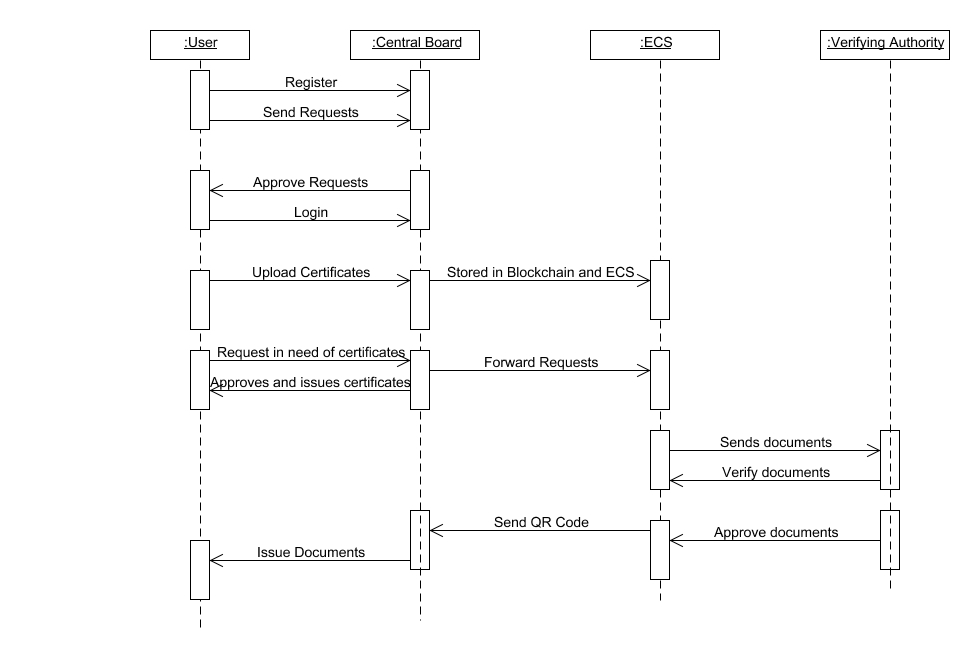
QR code

QR Request

**Fig: 5.1**

**5.2 Sequence Diagram:**

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

****

**5.3 Use Case Diagram:**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

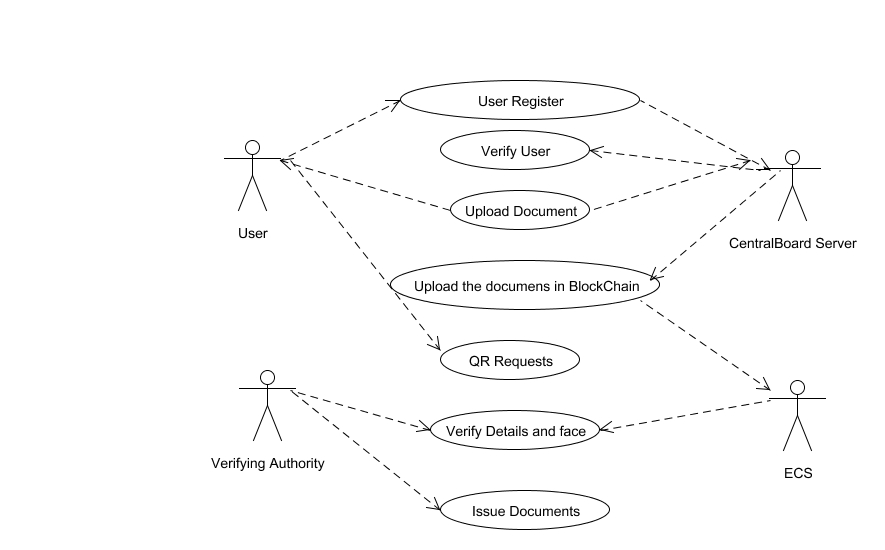
**5.3.1. USECASE DIAGRAM**

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

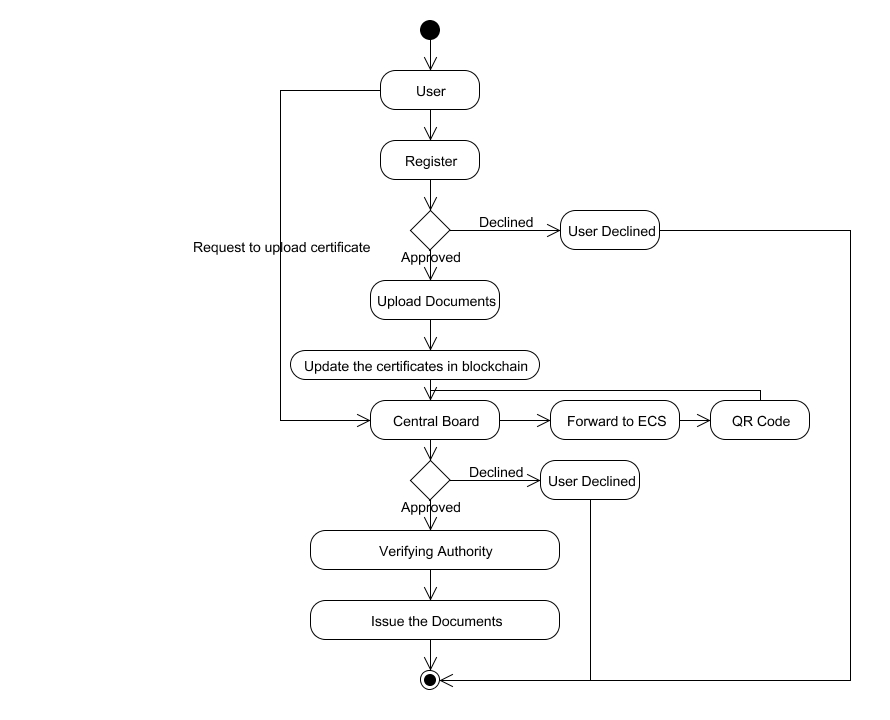
****

**5.4 Activity Diagram:**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

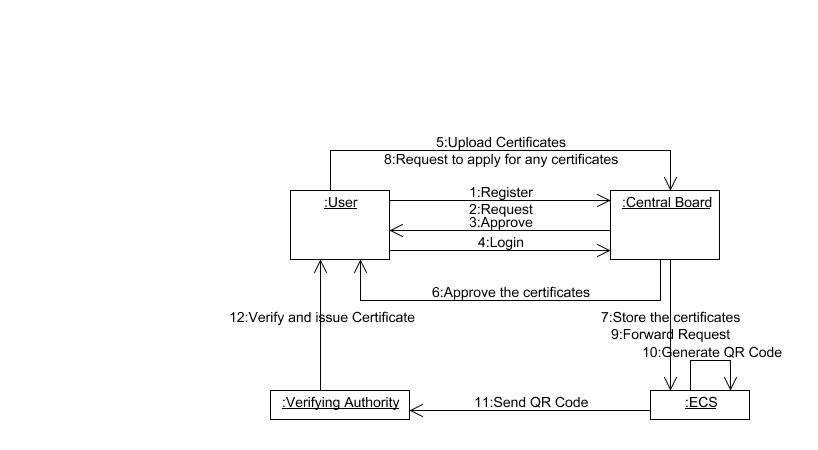
The most important shape types:

* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* A black circle represents the start of the workflow.
* An encircled circle represents the end of the workflow.

****

**5.5 Collaboration Diagram:**

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

****

**5.6 DATA FLOW DIAGRAM:**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

**Level 0:**

Registration

Login

Central Board

**Level 1:**

Registration

Login

Upload certificate

Central Board

Download Certificate

**Level 2:**

Registration

Login

Upload certificate

Central Board

Download Certificate

Block chain

QR code scanner

**Level 3:**

Registration

Login

Upload certificate

Central Board

Download Certificate

Block chain

QR code scanner

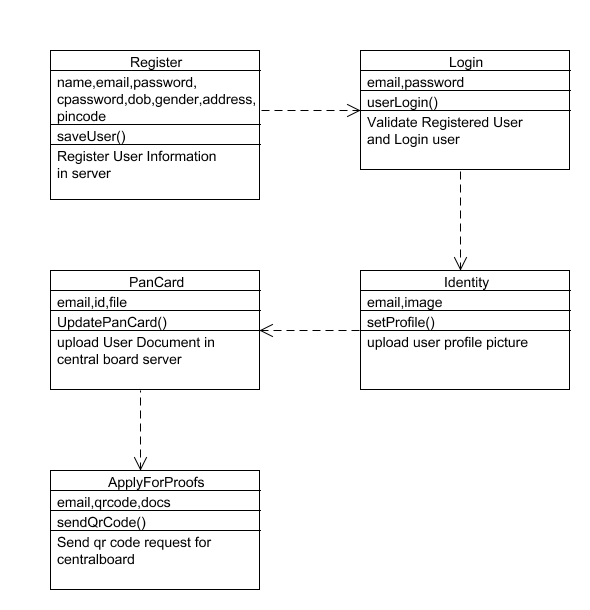
face

Verification

Receive certificate

**5.7 Class Diagram**

A Class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

****

**CHAPTER 6**

**SYSTEM DESIGN**

User needs to registers into his application and a request will be sent to central board server for authentication. Unless the central board server approves the request user cannot login into his account. When central board server approves the request a key will be generated and user can login into his account. After user login into his account he needs to upload certificates namely pan card, aadhar card, voter id, ssc certificates to central board server. Central board server will review the certificates and accepts or decline the certificates. If central board server accepts the accepts the certificate those details will be stored in E.C.S and Blockchain. If central board server declines the certificate it won’t be stored in E.C.S. or Block Chain. If user needs a certificate he will send request to central board server. If central board server found the user details to be genuine he accepts the request and forward a request to E.C.S where all the certificates will be there. E.C.S. responds for the request and certificates will be provided to the user. If user wants to apply for any certificates he will send request to central board server and central board server will check the details and forward the request to E.C.S. E.C.S will generate the QR Code and forwarded to user via central board server. User forwards the QR code to the verifying authority and if all details are correct and face matches with live face Verifying authority will issue the document.

**6.1 MODULES**

* **User Registration And Authentication**
* **User Upload Certificate**
* **Get Certificate**
* **QR Request And Response From Verification Authority**

**6.2 MODULE EXPLANATION:**

* **6.2.1 User Registration And Authentication**

In this module user needs to registers into his application and a request will be sent to central board server for authentication. Unless the central board server approves the request user cannot login into his account. When central board server approves the request a key will be generated and user can login into his account.

**User Upload Certificate:**

After user login into his account he needs to upload certificates namely pan card, aadhar card, voter id, ssc certificates to central board server. Central board server will review the certificates and accepts or decline the certificates. If central board server accepts the accepts the certificate those details will be stored in E.C.S and Blockchain. If central board server declines the certificate it won’t be stored in E.C.S. or Block Chain.

**Get Certificate:**

If user needs a certificate he will send request to central board server. If central board server found the user details to be genuine he accepts the request and forward a request to E.C.S where all the certificates will be there. E.C.S. responds for the request and certificates will be provided to the user.

**QR Request and Face verification:**

If user wants to apply for any certificates he will send request to central board server and central board server will check the details and forward the request to E.C.S. E.C.S will generate the QR Code and forwarded to user via central board server. User forwards the QR code to the verifying authority and if all details are correct and face matches with live face Verifying authority will issue the document.

**CHAPTER 7**

**CODING AND TESTING**

**7.1 CODING**

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

**7.2 CODING STANDARDS**

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

**7.2.1 NAMING CONVENTIONS**

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names**

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

**Member Function and Data Member name**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7**.2.2 VALUE CONVENTIONS**

Value conventions ensure values for variable at any point of time. This involves the following:

* Proper default values for the variables.
* Proper validation of values in the field.
* Proper documentation of flag values.

**7.2.3 SCRIPT WRITING AND COMMENTING STANDARD**

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

**7.2.4 MESSAGE BOX FORMAT**

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

* X – User has performed illegal operation.
* ! – Information to the user.

**7.3 TEST PROCEDURE**

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

**7.4 TEST DATA AND OUTPUT**

**7.4.1 UNIT TESTING**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

**7.4.2 FUNCTIONAL TESTS**

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* Performance Test
* Stress Test
* Structure Test

**7.4.3 PERFORMANCE TEST**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**7.4.4 STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

**7.4.5 STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file condition, I/O errors, buffer problems and textual errors in output information

**7.4.6 INTEGRATION TESTING**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

**7.5 TESTING TECHNIQUES / TESTING STRATERGIES**

**7.5.1 TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**7.5.1.1 WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving test cases
* Graph matrices Control

**7.5.1.2 BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**7.5.2 SOFTWARE TESTING STRATEGIES:**

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**7.5.2.1 INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**7.5.2.2 PROGRAM TESTING:**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

**7.5.2.3 SECURITY TESTING:**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

**7.5.2.4 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

\* The function or performance characteristics confirm to specifications and are accepted.

\* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

**7.5.2.5 USER ACCEPTANCE TESTING**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* Input screen design.
* Output screen design.

**Source Code**

**Screenshots:**

**REFERENCES**

[1] D. White, R. I. Kemp, R. Jenkins, M. Matheson, and A. M. Burton, “Passport officers’ errors in face matching,” *PLoS ONE*, vol. 9, no. 8, 2014, Art. no. e103510.

[2] Wikipedia. (2018). *Australia Smartgate*. [Online]. Available: https://en.wikipedia.org/wiki/SmartGate

[3] Wikipedia. (2018). *Epassport Gates*. [Online]. Available: <https://en>. wikipedia.org/wiki/EPassport\_gates

[4] U.S. Customs and Border Protection. (2018). *Automated Passport Control (APC)*. [Online]. Available: https://www.cbp.gov/travel/uscitizens/apc

[5] Xinjiang Heng An Perimeter Security Equipment Company. (2018). *What Is ID-Person Matching?* [Online]. Available: http://www.xjhazj.

com/xjhazj/vip\_doc/8380983.html

[6] Jumio. (2018). *Netverify ID Verification*. [Online]. Available: https://www.jumio.com/trusted-identity/netverify

[7] Mitek. (2018). *Mitek ID Verification*. [Online]. Available: https://www.miteksystems.com/mobile-verify

[8] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” Univ. Massachusetts, Amherst, MA, USA, Rep. 07-49, Oct. 2007 V. Starovoitov, D. Samal, and B. Sankur, “Matching of faces in camera images and document photographs,” in *Proc. ICASSP*, 2000, pp. 2349–2352. [10] T. Bourlai, A. Ross, and A. Jain, “On matching digital face images against scanned passport photos,” in *Proc. IEEE Int. Conf. Biometrics* *Identity Security (BIDS)*, 2009, pp. 1–10.

[11] T. Bourlai, A. Ross, and A. K. Jain, “Restoring degraded face images: A case study in matching faxed, printed, and scanned photos,” *IEEE* *Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 371–384, Jun. 2011.

[12] V. V. Starovoitov, D. I. Samal, and D. V. Briliuk, “Three approaches for face recognition,” in *Proc. Int. Conf. Pattern Recognit. Image Anal.*, 2002, pp. 707–711.

[13] Y. Shi and A. K. Jain, “DocFace: Matching ID document photos to selfies,” in *Proc. BTAS*, 2018, pp. 1–8.

[14] X. Zhu *et al.*, “Large-scale bisample learning on ID vs. spot face recognition,” *arXiv:1806.03018*, 2018.

[15] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “VGGFace2: A dataset for recognising faces across pose and age,” in *Proc. IEEE Conf.* *Autom. Face Gesture Recognit.*, 2018, pp. 1–8.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. NIPS*, 2012, pp. 1–9.

[17] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” in *Proc. CVPR*, 2014, pp. 1701–1708.

[18] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *Proc. NIPS*, 2014, pp. 1988–1996.

[19] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. CVPR*, 2015, pp. 815–823.

[20] I. Masi, A. T. Tr`ân, T. Hassner, J. T. Leksut, and G. Medioni, “Do we really need to collect millions of faces for effective face recognition?” in *Proc. ECCV*, 2016, pp. 579–596.

[21] W. Liu *et al.*, “SphereFace: Deep hypersphere embedding for face recognition,” in *Proc. CVPR*, 2017, pp. 212–220.

[22] A. Hasnat, J. Bohné, J. Milgram, S. Gentric, and L. Chen, “DeepVisage: Making face recognition simple yet with powerful generalization skills,” in *Proc. ICCV Workshops*, 2017, pp. 1682–1691.

[23] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 926–930, Jul. 2018.

[24] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks,” in *Proc. ICML*, 2016, pp. 507–516.

[25] W. Liu *et al.*, “Deep hyperspherical learning,” in *Proc. NIPS*, 2017, pp. 1–11.

[26] S. Z. Li and A. Jain, *Encyclopedia of Biometrics*. New York, NY, USA: Springer, 2015.

[27] S. Z. Li, R. Chu, S. Liao, and L. Zhang, “Illumination invariant face recognition using near-infrared images,” *IEEE Trans. Pattern Anal.* *Mach. Intell.*, vol. 29, no. 4, pp. 627–639, Apr. 2007.

[28] J. Choi, S. Hu, S. S. Young, and L. S. Davis, “Thermal to visible face recognition,” in *Proc. SPIE DSS-DS107 Biometric Technol. Human* *Identification IX*, 2012, pp. 1–10.

[29] X. Tang and X. Wang, “Face photo recognition using sketch,” in *Proc. ICIP*, 2002, p. 1.

[30] X. Wang and X. Tang, “Face photo-sketch synthesis and recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 1955–1967, Nov. 2009.

[31] Q. Liu, X. Tang, H. Jin, H. Lu, and S. Ma, “A nonlinear approach for face sketch synthesis and recognition,” in *Proc. CVPR*, 2005, pp. 1005–1010.

[32] X. Gao, J. Zhong, J. Li, and C. Tian, “Face sketch synthesis algorithm based on E-HMM and selective ensemble,” *IEEE Trans. Circuits Syst.* *Video Technol.*, vol. 18, no. 4, pp. 487–496, Apr. 2018.

[33] S. Liao, D. Yi, Z. Lei, R. Qin, and S. Z. Li, “Heterogeneous face recognition from local structures of normalized appearance,” in *Proc. ICB*, 2009, pp. 209–218.

[34] B. Klare and A. K. Jain, “Heterogeneous face recognition: Matching NIR

to visible light images,” in *Proc. IEEE ICPR*, 2010, pp. 1513–1516.

[35] B. F. Klare and A. K. Jain, “Heterogeneous face recognition using kernel prototype similarities,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1410–1422, Jun. 2013.

[36] X. Liu, L. Song, X. Wu, and T. Tan, “Transferring deep representation for NIR-vis heterogeneous face recognition,” in *Proc. ICB*, 2016, pp. 1–8.

[37] R. He, X. Wu, Z. Sun, and T. Tan, “Learning invariant deep representation for NIR-vis face recognition,” in *Proc. AAAI*, 2017, pp. 2000–2006

[38] X. Wu, L. Song, R. He, and T. Tan, “Coupled deep learning for heterogeneous face recognition,” in *Proc. AAAI*, 2018.

[39] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *Proc. ICML Workshop Deep Learn.*, 2015, pp. 1–8.

[40] O. Vinyals *et al.*, “Matching networks for one shot learning,” in *Proc. NIPS*, 2016, pp. 3637–3645.

[41] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for fewshot learning,” in *Proc. NIPS*, 2017, pp. 1–9.

[42] S. Chopra, R. Hadsell, and Y. Lecun, “Learning a similarity metric discriminatively, with application to face verification,” in *Proc. CVPR*, 2005, pp. 539–546.

[43] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *Proc. ICLR*, 2017, pp. 1–11.

[44] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi, “Learning feed-forward one-shot learners,” in *Proc. NIPS*, 2016, pp. 523–531.

[45] H. Qi, M. Brown, and D. G. Lowe, “Low-shot learning with imprinted weights,” in *Proc. CVPR*, 2018, pp. 5822–5830.

[46] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, “No fuss distance metric learning using proxies,” 2017, pp. 360–368.

[47] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, “MS-Celeb-1M: A dataset and benchmark for large-scale face recognition,” in *Proc. ECCV*, 2016, pp. 1–17.

[48] H. Wang *et al.*, “Cosface: Large margin cosine loss for deep face recognition,” in *Proc. CVPR*, 2018, pp. 5265–5274.

[49] R. Ranjan, C. D. Castillo, and R. Chellappa, “L2-constrained softmax loss for discriminative face verification,” *arXiv:1703.09507*, 2017.

[50] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, “NormFace: L2 hypersphere embedding for face verification,” in *Proc. ACM MM*, 2017.

[51] M. A. Hasnat, J. Bohné, J. Milgram, S. Gentric, and L. Chen, “von Mises-Fisher mixture model-based deep learning: Application to face verification,” *arXiv:1706.04264*, 2017.

[52] L. V. D. Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[53] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *Proc. ECCV*, 2016, pp. 499–515.

[54] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal* *Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[55] S. Liao, Z. Lei, D. Yi, and S. Z. Li, “A benchmark study of large-scale unconstrained face recognition,” in *Proc. IJCB*, 2014, pp. 1–8.

[56] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” *arXiv:1801.07698*, 2018.