# ABSTRACT

Face Recognition is one of the revolution technology, which is based on various machine learning and deep learning algorithms, which is mostly used for bio metrics, but it is also used to detect criminals in traffic signals, identify the and track the lost persons, etc., This project is written in python, where a GUI appears with a button, which let us to open the face recognition program, that captures the video frames from the webcam of a laptop or a pc, and then analyses it, compares the faces inside it with the faces given in the data before, and if the faces matches, it displays the name of the face, which is also given before, else, it recognizes the face as an unknown face.

# INDEX

**1.INTRODUCTION**

1.1 INTRODUCTION TO PROJECT

## 2. SYSTEM ANALYSIS

2.1 INTRODUCTION

2.2 STUDY OF THE SYSTEM

2.3 HARDWARE AND SOFTWARE REQUIRMENTS

2.4 PROPOSED SYSTEM

2.5 INPUT AND OUTPUT

2.6 PROCESS MODULES USED WITH JUSTIFICATION

## 3. FEASIBILITY REPORT

3.1 TECHNICAL FEASIBILITY

3.2 OPERATIONAL FEASIBILITY

* 1. ECONOMICAL FEASIBILTY

**4.SOFTWARE REQUIREMENT SPECIFICATIONS**

* 1. FUNCTIONAL REQUIREMENTS
  2. PERFORMANCE REQUIREMENTS

## 5. SELECTED SOFTWARE

5.1INTRODUCTION TO PYTHON

5.2PYTHON

## 6. SYSTEM DESIGN

6.1INTRODUCTION

6.2SYSTEM WORK FLOW

6.3UML DIAGRAMS

1. **HISTORY OF FACE RECCOGNITION**
2. **CONCLUSION**

## INTRODUCTION

Artificial Intelligence is the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

Machine Learning is the application of Artificial Intelligence that provides the ability to automatically learn and improve from experience without being explicitly programmed. *Machine learning* focuses on the development of computer programs that can access data and use it learn for themselves.

Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

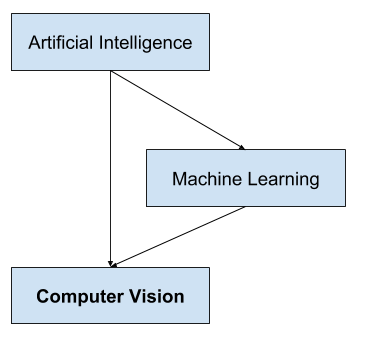
Computer vision is distinct from image processing. Image processing is the process of creating a new image from an existing image, typically simplifying or enhancing the content in some way. It is a type of digital signal processing and is not concerned with understanding the content of an image.

The 2010 textbook on computer vision titled “Computer Vision: Algorithms and Applications” provides a list of some high-level problems where we have seen success with computer vision.

* Optical character recognition (OCR)
* Machine inspection
* Retail (e.g. automated checkouts)
* 3D model building (photogrammetry)
* Medical imaging
* Automotive safety
* Match move (e.g. merging CGI with live actors in movies)
* Motion capture (mocap)
* Surveillance
* Fingerprint recognition and biometrics

A given computer vision system may require image processing to be applied to raw input, e.g. pre - processing images.

To be broad, Computer vision is the field of study focused on the problem of helping computers to see.



The above shown a picture showing the relationship between Artificial Intelligence and Machine Learning with Computer Vision.

Facial recognition is a way of recognizing a human face through technology. A facial recognition system uses bio-metrics to map facial features from a photograph or video. It compares the information with a database of known faces to find a match. Facial recognition can help verify personal identity, but it also raises privacy issues.

Facial Recognition is also called as face recognition.

Facial recognition technology is quickly becoming the most secure and reliable tool for user authentication. Facial recognition is mostly used for security purposes, though there is increasing interest in other areas of use. In fact, facial recognition technology has received significant attention as it has potential for a wide range of application related to law enforcement as well as other enterprises.

***What Is A Graphical User Interface(GUI)***

**GUI** is a desktop app which helps you to interact with the computers. They are used to perform different tasks in the desktops, laptops, other electronic devices, etc.., Here, we mainly talking about the laptops and desktops.

• **GUI** apps like **Text-Editors** are used to create, read, update and delete different types of files.

1. **GUI** apps like **Sudoku, Chess, Solitaire, etc..,** are games which you can play.

• **GUI** apps like **Chrome, Firefox, Microsoft Edge, etc..,** are used to surf the **Internet**.

They are some different types of **GUI** apps which we daily use on the laptops or desktops. We are going to learn how to create those type of apps.

***What Is Tkinter***

**Tkinter** is an inbuilt Python module used to create simple GUI apps. It is the most commonly used module for GUI apps in the Python.

There are some basic steps to create a GUI in tkinter.. they are..

### Steps:-

1. import the module **tkinter**.

• Initialize the window manager with the **tkinter.Tk()** method and assign it to a variable **window**. This method creates a blank window with close, maximize and minimize buttons.

1. Rename the title of the window as you like with the **window.title(title\_of\_the\_window)**.

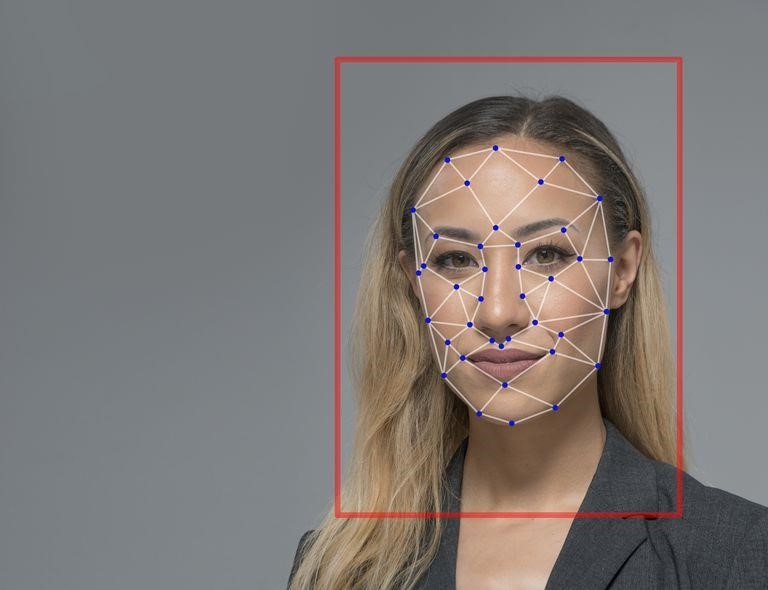
➢ **Label** is used to insert some objects into the **window**. Here, we are adding a **Label** with some text.

* **pack()** attribute of the widget is used to display the **widget** in a size it requires.
* Finally, the **mainloop()** method to display the **window** until you manually close it.

***How face recognition works :-***

You might be good at recognizing faces. You probably find it a cinch to identify the face of a family member, friend, or acquaintance. You’re familiar with their facial features like their eyes, nose, mouth and how they come together.

That’s how a facial recognition system works, but on a grand, algorithmic scale. Where you see a face, recognition technology sees data. That data can be stored and accessed. For instance, half of all Indian adults have their images stored in one or more facial-recognition databases that law enforcement agencies can search, according to a Georgetown University study.



Face Recognition Software is an application that is used to detect faces, an application can be written in many programming languages, like C++, Java, C #, Python etc., In this project, GUI was written in Python with Tkinter library.

Face Recognition Python is the latest trend in Machine Learning techniques. OpenCV, the most popular library for computer vision, provides bindings for Python. OpenCV uses machine learning algorithms to search for faces within a picture. Though new, Face Recognition Python code is a very popular concept. It is interesting to know about the different ways of face detection using Python.

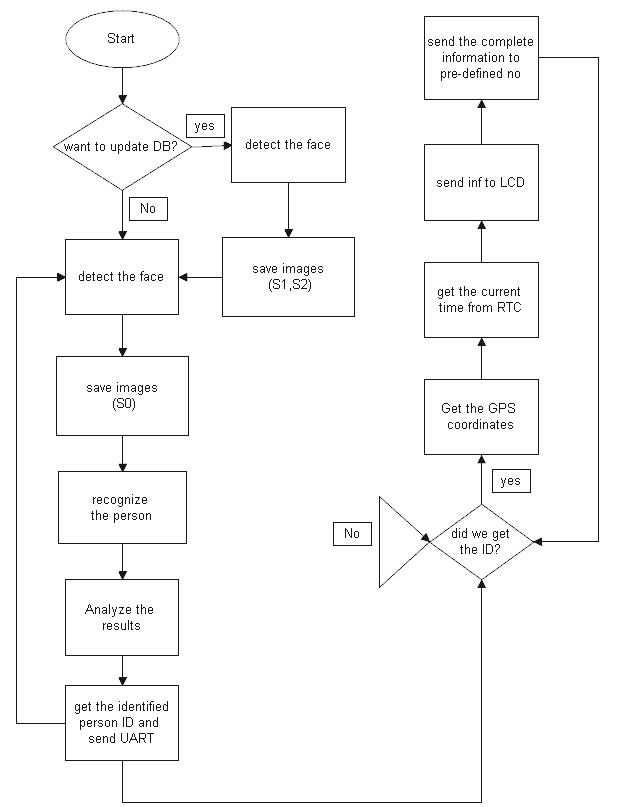
The software uses Deep Learning algorithms to compare a live capture or digital image to the stored face print in order to verify an individual’s identity.

High-quality cameras in mobile devices have made facial recognition a viable option for authentication as well as identification. Apple’s iPhone X, for example, includes Face ID technology that lets users unlock their phones with a face print mapped by the phone’s camera.

The phone’s software, which is designed with 3-D modeling to resist being spoofed by photos or masks, captures and compares over 30,000 variables. Face ID can also be used to authenticate purchases with Apple Pay and in the iTunes Store, App Store, and iBooks Store. Apple encrypts and stores face print data in the cloud, but authentication takes place directly on the device.

Faces are made of thousands of fine lines and features that must be matched. The face recognition using Python, break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to face Recognition Python is the latest trend in Machine Learning techniques. OpenCV uses Machine Learning algorithms to search for faces within a picture.

The most popular and probably the simplest way to detect faces using Python is by using the OpenCV package. Originally written in C/C++, OpenCV now provides bindings for Python. It uses machine learning algorithms to search for faces within a picture. Faces are very complicated, made of thousands of small patterns and features that must be matched. The face recognition algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve, known as classifiers.



The cascades are a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you. Since face detection is such a common case, OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands to legs.

You may use other alternatives to OpenCV, like dlib – that come with Deep Learning based Detection and Recognition models.

Face Recognition using Python and OpenCV follows a well-defined pattern. When you meet someone for the first time in your life, you look at his/her face, eyes, nose, mouth, color, and overall features. This is your mind learning or training for the face recognition of that person by gathering face data. Then the person tells you his/her name.

At this point, your mind knows that the face data it just learned belongs to the person. Now, your mind is trained and ready to do face recognition. Next time when you will see the person or his/her face in a picture you will immediately recognize. This is how Face Recognition works. The more you will meet, the more data your mind will collect about the person and the better you will become at recognizing him/her.

There are three stages of face recognition in any language.. they are:

**1.Training Data Gathering**: Gather face data (face images in this case) of the persons you want to recognize.

**2.Training of Recognizer**: Feed that face data (and respective names of each face) to the face recognizer so that it can learn.

**3.Recognition**: Feed new faces of the persons and see if the face recognizer you just trained recognizes them.

OpenCV provides the following three face recognizers:

1.Eigen face recognizer

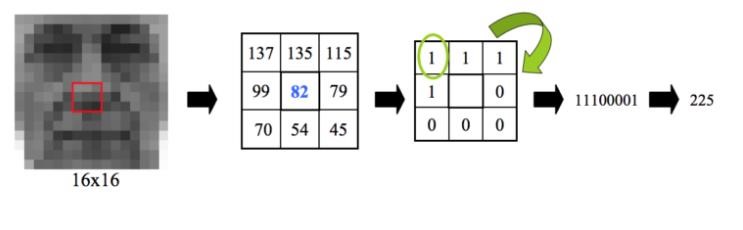
2.Fisher face recognizer

3.LBPH face recognizer

Although there are three face recognizers in OpenCV, the most effective and used recognizer is the LBPH Facce Recognizer.

### Local Binary Patterns Histograms (LBPH) Face Recognizer

Both Eigen faces and Fisherf aces are affected by light and in real life, perfect light conditions are not always available. LBPH face recognizer is an improvement to overcome this drawback. LBPH algorithm tries to find the local structure of an image and it does that by comparing each pixel with its neighboring pixels. With so much just on the horizon, it will be interesting to see where this rise in Facial Recognition technology takes us.

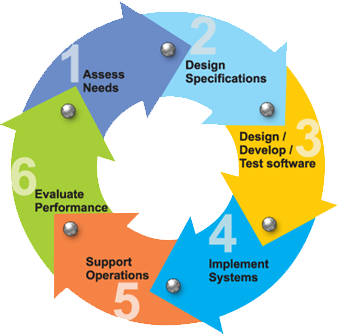


Face recognition is gaining importance very fast in various fields. We have entered an age when Facial Recognition technologies will soon be part of everyday life. China, for example, monitors by CCTV or by police wearing special glasses and then logs onto a database that checks on the habitual behavior of the people, their social credit and even their friends.

Cameras and facial recognition are increasingly being used in public and private buildings. Some schools in the United States are now installing facial recognition systems, to prevent gun attacks by students, given that most rampages are carried out by students whose faces will already be on a database and have full access to the premises. This has led to increased demand for coders and developers with knowledge of Face Recognition algorithms; Python and OpenCV, in particular.

#### 2.1 INTRODUCTION Software Development Life Cycle:-

There is various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.



**Requirements:**

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements. Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work. The overall result is the system as a whole and how it performs, not how it is actually going to do it.

### Design

The software system design is produced from the results of the requirements phase. Architects have the ball in their court during this phase and this is the phase in which their focus lies. This is where the details on how the system will work is produced. Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

### Implementation

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle. For a developer, this is the main focus of the life cycle because this is where the code is produced. Implementation my overlap with both the design and testing phases. Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

### Testing

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model. Now let’s delve into some of the traditional and widely used variations.

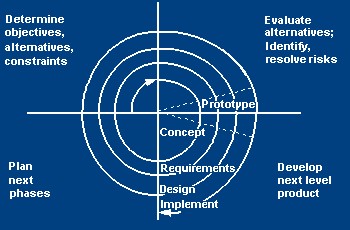
### SDLC METHDOLOGIES

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:
  1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
  2. Defining the requirements of the second prototype.
  3. Planning an designing the second prototype.
  4. Constructing and testing the second prototype.
* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

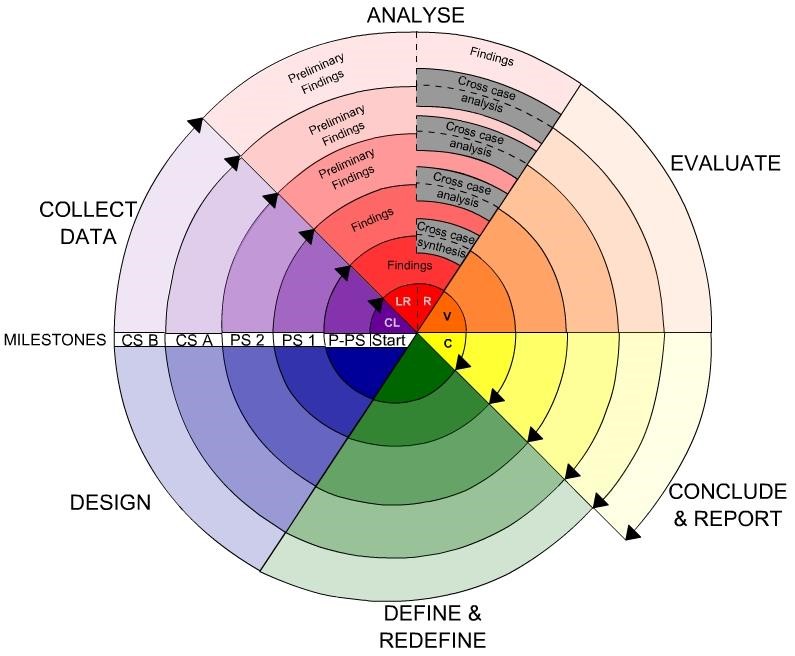
#### 2.2 STUDY OF THE SYSTEM

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface. The GUI’s at the top level has been categorized as follows

1. Administrative User Interface Design
2. The Operational and Generic User Interface Design

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The Interface helps the administration with all the transactional states like data insertion, data deletion, and data updating along with executive data search capabilities.

The operational and generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.



#### 3. Fundamental Concepts on (Domain)

What is Cloudera?

Cloudera is revolutionizing enterprise data management by offering the first unified Platform for Big Data: The Enterprise Data Hub. Cloudera offers enterprises one place to store, process, and analyze all their data, empowering them to extend the value of existing investments while enabling fundamental new ways to derive value from their data.

Why do customers choose Cloudera?

Cloudera was the first commercial provider of python-related software and services and has the most customers with enterprise requirements, and the most experience supporting them, in the industry. Cloudera’s combined offering of differentiated software (open and closed source), support, training, professional services, and indemnity brings customers the greatest business value, in the shortest amount of time, at the lowest TCO.

#### Data Mining

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it. Extraction of information is not the only process we need to perform; data mining also involves other processes such as Data Cleaning, Data Integration, Data Transformation, Data Mining, Pattern Evaluation and Data Presentation. Once all these processes are over, we would be able to use this information in many applications such as Fraud Detection, Market Analysis, Production Control, Science Exploration, etc.

What is Data Mining?

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications:

Market Analysis

Fraud Detection

Customer Retention

Production Control

Science Exploration

Data Mining Applications

Data mining is highly useful in the following domains: Market Analysis and Management

Corporate Analysis

& Risk Management Fraud Detection

Apart from these, data mining can also be used in the areas of production control, customer retention, science exploration, sports, astrology, and Internet Web SurfAid

Market Analysis and Management

Listed below are the various fields of market where data mining is used: Customer Profiling - Data mining helps determine what kind of people buy what kind of products.

Identifying Customer Requirements - Data mining helps in identifying thebest products for different customers. It uses prediction to find the factors that may attract new customers.

Cross Market Analysis - Data mining performs Association/correlations between product sales.

Target Marketing - Data mining helps to find clusters of model customers who share the same characteristics such as interests, spending habits, income, etc.

Determining Customer purchasing pattern - Data mining helps in determining customer purchasing pattern.

Providing Summary Information - Data mining provides us various multidimensional summary reports.

Corporate Analysis andRisk Management

Data mining is used in the following fields of the Corporate Sector: Finance Planning and Asset Evaluation - It involves cash flow analysis and prediction, contingent claim analysis to evaluate assets.

Resource Planning - It involves summarizing and comparing the resources and spending. Competition - It involves monitoring competitors and market directions.

Fraud Detection

Data mining is also used in the fields of credit card services and telecommunication to detect frauds. In fraud telephone calls, it helps to find the destination of the call, duration of the call, time of the day or week, etc. It also analyzes the patterns that deviate from expected norms.

Data mining essential step in the process of knowledge discovery

1.Data cleaning (to remove noise and inconsistent data)

1. Data integration (where multiple data sources may be combined)
2. Data selection (where data relevant to the analysis task are retrieved from the database)
3. Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance)
4. Data mining (an essential process where intelligent methods are applied in order to extract data patterns)
5. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures)
6. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user)

Based on this view, the architecture of a typical data mining system may have the following major components

Database, data warehouse, World Wide Web, or other information repository: This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories.

Data cleaning and data integration techniques may be performed on the data. Database or data warehouse server: The database or data warehouse server is responsible for fetching the relevant data, based on the user’s data mining request.

Knowledge base: This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern’s interestingness based on its unexpectedness, may also be included. Other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

Data mining engine: This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

Pattern evaluation module: This component typically employs interestingness measures and interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used.

For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.

User interface: This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

**DATA MINING ON WHAT KIND OF DATA?**

### DATA WAREHOUSES

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing

### RELATIONAL DATABASES

A database system, also called a database management system (DBMS), consists of a collection of interrelated data, known as a database, and a set of software programs to manage and access the data A relational database is a collection of tables, each of which is assigned a unique name. Each table consists of a set of attributes (columns or fields) and usually stores a large set of tuples (records or rows)

Each tuple in a relational table represents an object identified by a unique key and described by a set of attribute values A semantic data model, such as an entity-relationship (ER) data model, is often constructed for relational databases.

### OBJECT-RELATIONAL DATABASES

Based on an object-relational data model Extends the relational model by providing a rich data type for handling complex objects and object orientation Objects that share a common set of properties can be grouped into an object class. Each object is an instance of its class. Object classes can be organized into class/subclass hierarchies

### ADVANCED DATA AND INFORMATION SYSTEMS

With the progress of database technology, various kinds of advanced data and information systems have emerged and are undergoing development to address the requirements of new applications handling spatial/temporal data (such as maps) engineering design data (such as the design of buildings, system components, or integrated circuits) hypertext and multimedia data (including text, image, video, and audio data) time-related data (such as historical records or stock exchange data) stream data (such as video surveillance and sensor data, where data flow in and out like streams) the World Wide Web (a huge, widely distributed information repository made available by the Internet)

### THE WORLD WIDE WEB

The World Wide Web and its associated distributed information services, such as Yahoo! and Google provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining)

Database or data warehouse server responsible for fetching the relevant data, based on the user’s data mining request can be decouples/loose coupled/tightly coupled with the database layer

Knowledge base the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns interestingness constraints or thresholds, metadata, concept hierarchies, etc.

Data mining engine this is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis query languages (DMQL) based on mining primitives to access the data

Pattern evaluation module interacts with the data mining modules so as to focus the search toward interesting patterns may use interestingness thresholds to filter out discovered patterns may be integrated with the mining module

User interface communicates between users and the data mining system allows the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms

#### 4. System Analysis

The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in [systems engineering,](http://en.wikipedia.org/wiki/Systems_engineering) [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering,](http://en.wikipedia.org/wiki/Software_engineering) is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of [software development methodologies.](http://en.wikipedia.org/wiki/Software_development_methodologies) These methodologies form the framework for planning and controlling the creation of an information system the [software development process.](http://en.wikipedia.org/wiki/Software_development_process)

**SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

**2.4.Proposed System:**

We have implemented SVR algorithm(Support Vector Regression) of Machine learning using Python. The predictions are most approximate with SVR Algorithms as they Linear or Gaussian. The algorithm automatically uses the kernel function that is most appropriate to the data.SVM uses the linear kernel when there are many attributes (more than 100) in the training data, otherwise it uses the Gaussian kernel. In the proposed system we have takes taken the datasets which has the price and days based on the dataset we have made feature list and target list where the target\_list is price values andfeature list is the days. After the analysis of data is done we have fitted both feature list and target list using Python Machine learning SVN Algorithm and predicted the values for 1,30 and 365 days from the last day of the dataset values. Finally we have plotted a graph based on the results from the predicted analysis done with SVN Algorithm.

#### 2.5Functional requirements

##### user

* Load data
* Data analysis
* Data preprocessing
* Model building
* Prediction

#### Non-Functional Requirements

1. Secure access of confidential data (user’s details). SSL can be used.
2. 24 X 7 availability.
3. Better component design to get better performance at peak time
4. Flexible service based architecture will be highly desirable for future extension

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operational Feasibility
* Economical Feasibility

#### 3.1. TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

#### 3.2. OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

#### 3.3. ECONOMICAL FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

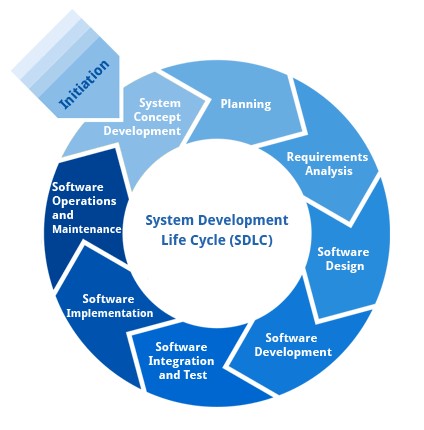
The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**4.1Functional requirements**  Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization,.
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.
* Understanding user’s preferences, expertise level and his business requirements through a friendly questionnaire.
* Input data can be in four different forms - Relational DB, text files, .xls and xml files. For testing and demo you can choose data from any domain. UserB can provide business data as input.  **Non-Functional Requirements**
  1. Secure access of confidential data (user’s details). SSL can be used.
  2. 24 X 7 availability.
  3. Better component design to get better performance at peak time
  4. Flexible service based architecture will be highly desirable for future extension

##### Software Development Life Cycle

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



##### Requirement Analysis and Design

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

##### Implementation

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator.

Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like C, are used for coding. With respect to the type of application, the right programming language is chosen.

###### Testing

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

##### Maintenance

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

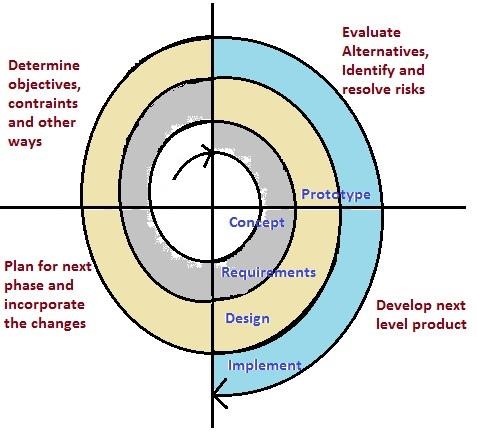
### SDLC METHDOLOGIES

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:
  + 1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning an designing the second prototype.
    4. Constructing and testing the second prototype.
* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**4.1. FUNCTIONAL REQUIREMENTS**

#### OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

#### OUTPUT DEFINITION

**The outputs should be defined in terms of the following points:**

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

#### INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

#### ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

#### ERROR DETECTION

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

#### DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

#### USER INTERFACE DESIGN

It is essential to consult the system users and discuss their needs while designing the user interface:

**USER INTERFACE SYSTEMS CAN BE BROADLY CLASSIFIED AS:**

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

#### USER\_INITIATED INTERFACES

User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

#### COMPUTER-INITIATED INTERFACES

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**ERROR MESSAGE DESIGN:**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

##### 4.2. PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**IMPLIMENTATION ON (PYTHON):**

**What Is A Script?**

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

##### Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

##### Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

##### You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,* or just about any word processor if you want to.

##### Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

**Python** what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If your not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

* Open source general-purpose language.
* Object Oriented, Procedural, Functional
* Easy to interface with C
* Easy-ish to interface with C++ (via SWIG)
* Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

##### History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

##### Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C,.

###### 6.1. INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue. Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

**Data Flow Diagrams:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts. The Basic Notation used to create a DFD’s are as follows:

1. **Dataflow:** Data move in a specific direction from an origin to a destination.

2.**Process:** People, procedures, or devices that use or produce (Transform) Data.

The physical component is not identified.

**3.Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

1. **Data Store:** Here data are stored or referenced by a process in the System.

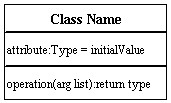
**What is a UML Class Diagram?**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

##### Basic Class Diagram Symbols and Notations

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



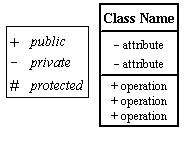
###### Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



###### Visibility

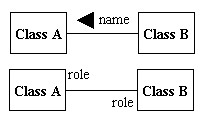
Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. [.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)



###### Associations

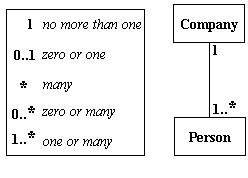
Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

***Note:*** It's uncommon to name both the association and the class roles.



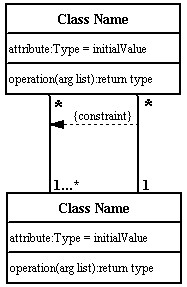
###### Multiplicity (Cardinality)

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.



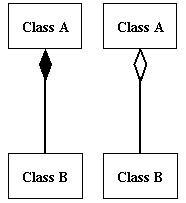
###### Constraint

Place constraints inside curly braces {}.

*Simple Constraint* 

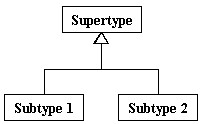
###### Composition and Aggregation

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate **composition** with a filled diamond. Use a hollow diamond to represent a simple **aggregation** relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



###### Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

**What is a UML Use Case Diagram?**

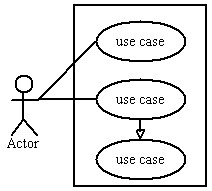
Use case diagrams model the functionality of a system using actors and use cases.

Use cases are services or functions provided by the system to its users.

**Basic Use Case Diagram Symbols and Notations**

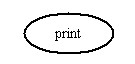
##### System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



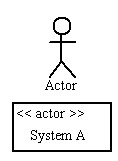
###### Use Case

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.



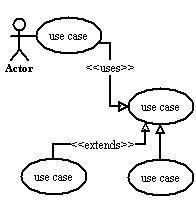
###### Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



###### Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



##### Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

**Basic Sequence Diagram Symbols and Notations**

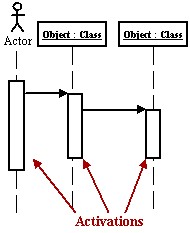
###### Class roles

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



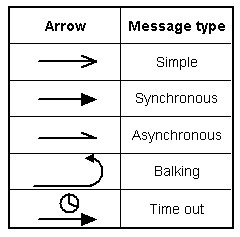
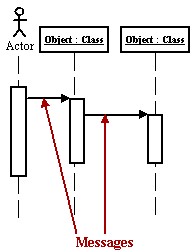
###### Activation

Activation boxes represent the time an object needs to complete a task.



###### Messages

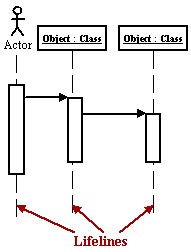
Messages are arrows that represent communication between objects. Use halfarrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



*Various message types for Sequence and Collaboration diagrams*

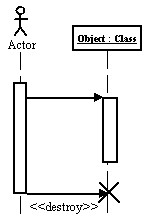
###### Lifelines

Lifelines are vertical dashed lines that indicate the object's presence over time.



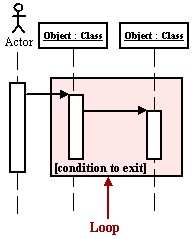
###### Destroying Objects

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



###### Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



###### Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

##### Basic Collaboration Diagram Symbols and Notations

###### Class roles

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.



###### Association roles

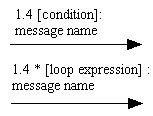
Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.



###### Messages

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in square brackets immediately following the sequence number. Use a \* after the sequence number to indicate a loop.

[Learn how to add arrows to your lines.](http://www.smartdraw.com/resources/tutorials/Lines)



###### Activity Diagram

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

##### Basic Activity Diagram Symbols and Notations

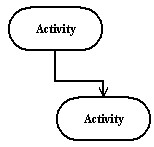
###### Action states

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



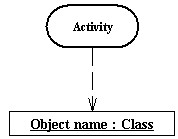
###### Action Flow

Action flow arrows illustrate the relationships among action states.



###### Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



###### Initial State

A filled circle followed by an arrow represents the initial action state.



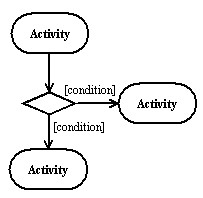
###### Final State

An arrow pointing to a filled circle nested inside another circle represents the final action state.



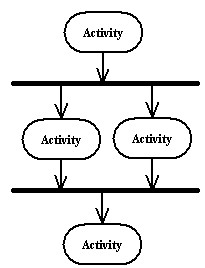
###### Branching

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



###### Synchronization

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



**Swimlanes**

Swimlanes group related activities into one column.

###### State chart Diagram

A state chart diagram shows the behavior of classes in response to external stimuli.

This diagram models the dynamic flow of control from state to state within a system.

##### Basic State chart Diagram Symbols and Notations

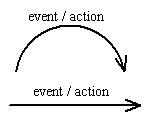
###### States

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



###### Transition

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



###### Initial State

A filled circle followed by an arrow represents the object's initial state.



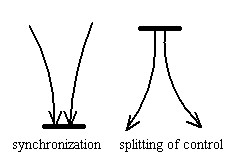
###### Final State

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



###### Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



**STATE CHART DIAGRAM:**

**What is a UML Component Diagram?**

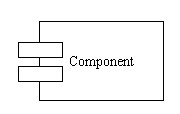
A component diagram describes the organization of the physical components in a system.

##### Basic Component Diagram Symbols and Notations

###### Component

A component is a physical building block of the system. It is represented as a rectangle with tabs.

[Learn how to resize grouped objects like components.](http://www.smartdraw.com/resources/tutorials/Objects)



###### Interface

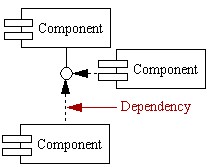
An interface describes a group of operations used or created by components.



###### Dependencies

Draw dependencies among components using dashed arrows.

[Learn about line styles in SmartDraw.](http://www.smartdraw.com/resources/tutorials/Lines)



**COMPONENT DIAGRAM:**

**What is a UML Deployment Diagram?**

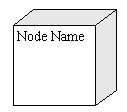
Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

##### Basic Deployment Diagram Symbols and Notations

###### Component

A node is a physical resource that executes code components.

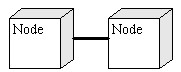
[Learn how to resize grouped objects like nodes.](http://www.smartdraw.com/resources/tutorials/Objects)



###### Association

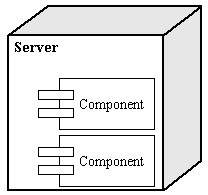
Association refers to a physical connection between nodes, such as Ethernet.

[Learn how to connect two nodes.](http://www.smartdraw.com/resources/tutorials/Lines)

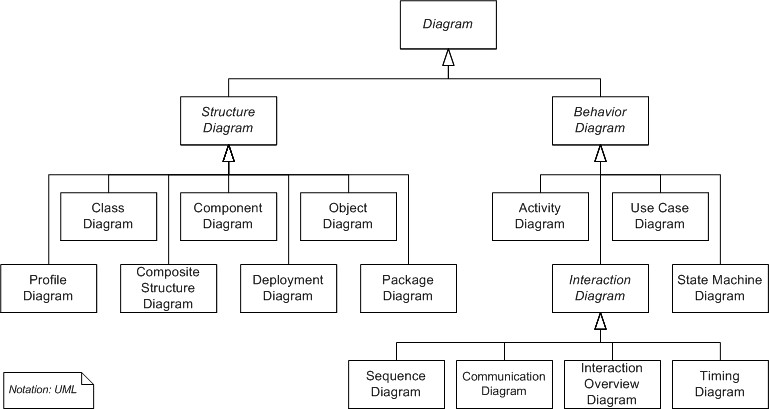


###### Components and Nodes

Place components inside the node that deploys them.



UML Diagrams Overview



UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

USECASE DAIGRAM:

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

* Functionalities to be represented as an use case
* Actors
* Relationships among the use cases and actors.

**user**

**take dataset**

**preparing dateset**

**clasification**

**supervised**

**unsupervised**

**classifying**

**regression**

**features**

**lables**

**testing&training**

**predending the data using alogorithms**

**SVM**

**SVR ALGORITHM**

**predict(result)**

**plotting**

SEQUENCE DAIGRAM

A **sequence diagram** in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

user

dataset

classification

supervised

classifying

testing&training

algorithms

plotting

predict(result)

1

: supervised

()

2

: unsupervised

()

3

: classifying

()

4

: regrission

()

5

: features

()

6

: lables

()

7

: SVM algorithm

()

8

: SVR algorithm

()

9

: using SVM algorithm

()

10

: results

()

COLLEBARATION DAIGRAM:

A **collaboration diagram**, also called a communication diagram or interaction diagram. A Collaboration diagram is easily represented by modeling objects in a system and representing the associations between the objects as links. The interaction between the objects is denoted by arrows. To identify the sequence of invocation of these objects, a number is placed next to each of these arrows. A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice versa. Hence, the elements of a Collaboration diagram are essentially the same as that of a Sequence diagram.

user

dataset

classification

supervised

classifying

testing&training

algorithms

predict(result)using SVM algorithms

plotting

predict(result)

()

: supervised

1

: unsupervised

()

2

()

3

: classifying

4

: regrission

()

:

features()

5

6

: lables

()

7

M algorithm()

8

: SVR algorithm

()

9

: using SVM algorithm

()

10

: results

()

ACTIVITY DAIGRAM:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An

activity diagram shows the overall flow of control.

user

dadaset

clasification

supervised

unsupervised

regression

classifying

features(days)

lables(price)

testing&training

predict the data using SVM algorithm

plotting

Component

**server**

**datasets**

**matplotlib**

**numpy**

**sklearn**

### SAMPLE CODE EXPLINATIONS

Numpy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays.

The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant)  created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software)

* a powerful N-dimensional array object
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities.

# HISTORY

## 7. HISTORY OF FACE RECOGNITION

It may seem to some that facial recognition came out of nowhere. But in truth, this technology has been in the works for some time. Take a look at the history of face recognition in order to shed light on how this transforming tech came to be, and how it has evolved over time.

Here are some key events in the history of facial recognition:

### Manual Measurements by Bledsoe (1960s):-

Many would say that the father of facial recognition was Woodrow Wilson Bledsoe. Working in the 1960s, Bledsoe developed a system that could classify photos of faces by hand using what’s known as a RAND tablet, a device that people could use to input horizontal and vertical coordinates on a grid using a stylus that emitted electromagnetic pulses. The system could be used to manually record the coordinate locations of various facial features including the eyes, nose, hairline and mouth.

These metrics could then be inserted in a database. Then, when the system was given a new photograph of an individual, it was able to retrieve the image from the database that most closely resembled that individual. At the time, face recognition was unfortunately limited severely by the technology of the era and computer processing power. However, it was an important first step in proving that face recognition was a viable biometric.

**Increased Accuracy with 21 Facial Markers (1970s):-**

In the 1970s, Goldstein, Harmon, and Lesk were able to add increased accuracy to a manual facial recognition system. They used 21 specific subjective markers including lip thickness and hair color in order to identify faces automatically. As with Bledsoe’s system, the actual biometrics had to still be manually computed.

### Eigenfaces (Late 1980s-Early 1990s):-

In 1988, Sirovich and Kirby began applying linear algebra to the problem of facial recognition. What became known as the Eigenface approach started as a search for a low-dimensional representation of facial images. Sirovich and Kriby were able to show that feature analysis on a collection of facial images could form a set of basic features. They were also able to show that less than one hundred values were required in order to accurately code a normalized face image.

In 1991, Turk and Pentland expanded upon the Eigenface approach by discovering how to detect faces within images. This led to the first instances of automatic face recognition. Their approach was constrained by technological and environmental factors, but it was a significant breakthrough in proving the feasibility of automatic facial recognition.

### FERET Program (1993-2000s):-

The Defense Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology rolled out the Face Recognition Technology (FERET) program beginning in the 1990s in order to encourage the commercial face recognition market. The project involved creating a database of facial images. The database was updated in 2003 to include high-resolution 24-bit color versions of images. Included in the test set were 2,413 still facial images representing 856 people. The hope was that a large database of test images for facial recognition would be able to inspire innovation, that might result in more powerful facial recognition technology.

#### Super Bowl XXXV (2002):-

At the 2002 Super Bowl, law enforcement officials used facial recognition in a major test of the technology. While officials reported that several “petty criminals” were detected, overall the test was seen as a failure. False positives and backlash from critics proved that face recognition wasn’t quite ready for prime time. One of the big technological limitations at the time was that face recognition did not yet work well in large crowds, functionality that is essential to using face recognition for event security.

#### Face Recognition Vendor Tests (2000s):-

The National Institute of Standards and Technology (NIST) began Face Recognition Vendor Tests (FRVT) in the early 2000s. Building on FERET, FRVTs were designed to provide independent government evaluations of facial recognition systems that were commercially available, as well as prototype technologies. These evaluations were designed to provide law enforcement agencies and the U.S. government with information necessary to determine the best ways to deploy facial recognition technology.

#### Law Enforcement Forensic Database (2009):-

In 2009, the Pinellas County Sherriff’s Office created a forensic database that allowed officers to tap into the photo archives of the state’s Department of Highway Safety and Motor Vehicles (DHSMV). By 2011, about 170 deputies had been outfitted with cameras that let them take pictures of suspects that could be crosschecked against the the database. This resulted in more arrests and criminal investigations than would have otherwise been possible.

#### Social Media (2010-Present):-

Beginning in 2010, Facebook began implementing facial recognition functionality that helped identify people whose faces may be featured in the photos that Facebook users update daily. While the feature was instantly controversial with the news media, sparking a slew of privacy-related articles, Facebook users at large did not seem to mind. Having no apparent negative impact on the website’s usage or popularity, more than 350 million photos are uploaded and tagged using face recognition each day.

#### First Major Installation of Face Recognition in an Airport (2011):-

In 2011, the government of Panama, partnering with then-U.S. Secretary of Homeland Security Janet Napolitano, authorized a pilot program of FaceFirst’s facial recognition platform in order to cut down on illicit activity in Panama’s Tocumen airport (known as a hub for drug smuggling and organized crime).

Shortly after implementation, the system resulted in the apprehension of multiple Interpol suspects. Pleased with the success of the initial deployment,

FaceFirst expanded into the facility’s north terminal. The FaceFirst implementation at Tocumen remains the largest biometrics installation at an airport to date.

#### Law Enforcement Agencies Adopt Mobile Face Recognition (2014):-

Beginning in 2014, The Automated Regional Justice Information System (ARJIS), began supplying partner agencies with FaceFirst’s mobile platform supporting face recognition for law enforcement. ARJIS, a complex criminal justice enterprise network that promotes information and data sharing among local, state and federal law enforcement agencies, wanted to solve a critical problem: instant identification for people who had no ID or did not want to be identified. Some of the agencies that started using mobile face recognition to identify suspects in the field include San Diego police, DOJ, FBI, DEA, CBP and U.S. Marshalls.

**Face Recognition “Inevitable” for Retail (2017):-**  As face recognition is adopted by retail faster than any other industry, experts are taking note. In a recent webinar, D&D Daily Publisher and Editor Gus Downing stated that face recognition is on an “inevitable path to retail adoption.” Downing, considered one of the foremost loss prevention thought leaders, is just one expert that now sees massive advantages for retailers who use a face recognition system.

#### iPhone X (2017):-

Apple released the iPhone X in 2017, advertising face recognition as one of it’s primary new features. The face recognition system in the phone is used for device security. The new model of iPhone sold out almost instantly, proving that consumers now accept facial recognition as the new gold standard for security.

#### Watchlist as a Service (2017):-

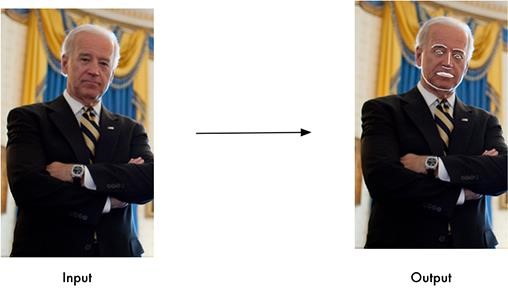
It’s becoming easier than ever for organizations to benefit from facial recognition technology. This year, FaceFirst introduced WatchList as a Service (WaaS) at the NRF Protect conference. WaaS is a new face recognition data platform designed to help prevent shoplifting and violent crime. WatchList includes a managed database of known criminals that pose a safety, theft or violent crime risk.

## DESIGN

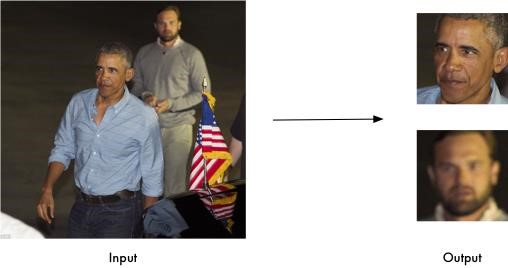
On a grand, algorithmic scale. Where we see a face, recognition technology sees data. That data can be stored and accessed. For instance, half of all Indian adults have their images stored in one or more facial-recognition databases that law enforcement agencies can search, according to a University study. Though technologies may vary, this is how face recognition works..

**Step 1**. A picture of your face is captured from a photo or video. Your face might appear alone or in a crowd. Your image may show you looking straight ahead or nearly in profile.

**Step 2**. Facial recognition software reads the geometry of your face. Key factors include the distance between your eyes and the distance from forehead to chin. The software identifies facial landmarks — one system identifies 68 of them — that are key to distinguishing your face. The result: your facial signature.



**Step 3**. Your facial signature — a mathematical formula — is compared to a database of known faces. And consider this: at least 117 million Indians have images of their faces in one or more police databases. According to a May 2018 report, the FBI has had access to 412 million facial images for searches.



**Step 4**. A determination is made. Your faceprint may match that of an image in a facial recognition system database.

In general, that’s how facial recognition works.

### CONCLUSION

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable.The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCA transform.

This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate. The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area. The fully automated face detection and recognition system was not robust enough to achieve a high recognition accuracy. The only reason for this was the face recognition subsystem did not display even a slight degree of in variance to scale, rotation or shift errors of the segmented face image.

However, if some sort of further processing, such as an eye detection technique, was implemented to further normalize the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system. Implementing an eye detection technique would be a minor extension to the implemented system and would not require a great dealof additional research.

All other implemented systems displayed commendable results and reflect well on the deformable template and Principal Component Analysis strategies.The most suitable real-world applications for face detection and recognition systemsare for mugshot matching and surveillance.