A Project Report on

REAL TIME HUMAN EMOTION RECOGNITION BASED ON FACIAL EXPRESSIONS

Submitted in Partial Fulfillment Of the requirement for the Award of the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE& ENGINEERING

Submitted by

P.SAILAJA	193H1A0534
S. SHAFIYA	193H1A0546
G.KEERTANA	193H1A0512
S.ANITHA	193H1A0550

Under the extreme guidance of

Smt. SURAM SWETHA, M.Tech.,

Assistant Professor, Dept of CSE.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

K.L.M COLLEGE OF ENGINEERING FOR WOMEN

(APPROVED BY AICTE AND AFFILIATED TO J.N.T.U. ANANTAPURAMU)

KADAPA-516 003

2019-2023

K.L.M COLLEGE OF ENGINEERING FOR WOMEN
(Approved by AICTE and Affiliated to J.N.T. U, Anantapuramu)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"Real Time Human Emotion Recognition based on Facial Expression"**that is being submitted by **P.SAILAJA(Reg.No.193H1A0534), S.SHAFIYA(Reg.No.193H1A0546), G.KEERTANA(Reg.No.193H1A0512), S.ANITHA (Reg.No.193H1A0550)** in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology in COMPUTER SCIENCE & ENGINEERING** to the **Jawaharlal Nehru Technological University Anantapur** is a record of bonafide work carried out by under my guidance and supervision. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree.

Internal Guide	**Head of the Department**
S.SWETHA, M.Tech, **Assistant Professor,** **Department of CSE.**	**D.SRINIVASULU, M.Tech, (Ph.D).** **HOD & Assistant Professor,** **Department of CSE.**

Submitted for university Examination (Viva Voice) held on ----------------------

Internal Examiner	**External Examiner**

ACKNOWLEDGMENT

We are highly indebted to our superiors for the encouragement and help given for the completion of our project.

We wish to express our gratitude to **Dr.K.CHANDRA OBUL REDDY**, **Managing Director, Kandula Group of Institutions, and Dr. M.V. RATHNAMMA, Principal of KLM College of Engineering for Women, Kadapa** for their valuable suggestions and advice throughout the course.

At the same time, we express a deep sense of reverence in profound gratitude to our **Head of the Department and Assistant Professor, D. SRINIVASULU,** Department of **COMPUTER SCIENCE & ENGINEERING** and our **Project Guide Smt. S. SWETHA, Assistant Professor,** Department of **COMPUTER SCIENCE & ENGINEERING** for their constant motivation, valuable help throughout the project work and for arranging all the facilities, and resources needed for our project. We also extend our thanks to other faculties for their cooperation during our course. Not to forget, our teaching and non-teaching staff and family who had, directly and indirectly, helped and supported us in completing our project.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 ABOUT THE PROJECT

Facial Detection and recognition research has been widely studied in recent years. The facial recognition applications plays an important role in many areas such as security, camera surveillance, identity verification in modern electronic devices, criminal investigations, database management systems and smart card applications etc.This work presents deep learning algorithms used in facial recognition for accurate identification and detection. The main objective of facial recognition is to authenticate and identify the facial features. However, the facial features are captured in real time and processed using haar cascade detection. The sequential process of the work is defined in three different phases where in the first phase human face is detected from the camera and in the second phase, the captured input is analyzed based on the features and database used with support of keras convolutional neural network model. In the last phase human face is authenticated to classify the emotions of human as happy, neutral, angry, sad, disgust and surprise. The proposed work presented is simplified in three objectives as face detection, recognition and emotion classification. In support of this work Open CV library, dataset and python programming is used for computer vision techniques involved. In order to prove real time efficacy, an experiment was conducted for multiple students to identify their inner emotions and find physiological changes for each face. The results of the experiments demonstrates the perfections in face analysis system. Finally, the performance of automatic face detection and recognition is measured with Accuracy

1.2EXISTING SYSTEM

Human computer interaction is a common trend and innate ability to distinguish among multiple faces. Until recent past computer vision problems were quite challenging but advent of modern technologies has trivially improved from the problems of varying light, changed by age, hair and other accessories. However, face recognition applications are used improve access to identify Human computer interaction is a common trend and innate ability to distinguish among and verify the people by their face features. Hence interpreting the facial features and their actions is much required. As these features and expressions helps in classify the emotions of human face. Recent advances in technology has resulted in the use of Artificial intelligence system as these systems are capable to understand and realize the emotion recognition through facial features. Hence this is an attempt to prove the existence of latest technological developments for human-computer interaction using deep learning or Convolution neural network models. To recognize and classify the human face various methods are required but deep learning technique outperforms other methods by its large capabilities of different datasets and fast computation capabilities. Usually the process of face recognition and classification involves various steps such as preprocessing, detection, orientation, extraction of features and classification of emotion.

1.2.1 Disadvantages :

- Huge Storage Requirements
- Less accuracy
- Privacy Issue

1.3 PROPOSED SYSTEM

The proposed work carries out in three sequential steps as Face Detection, Face Recognition and Face Classification. In the first step a video camera is used to capture the human face and detect the exact location of face by a bounding box coordinates for the face detected in real-time. This step involves face detection using Haar cascade detection with open CV library. Viola jones algorithm and haar cascade features are combined to detect human face. The images detected have shapes, objects and landscapes etc. In this phase human face is detected and face features are extracted and stored in the database for face recognition. The CNN model as shown in figure 4 uses VGG 16 to match the face from the database and recognize with the name for the face detected. Faces are recognized from the database and are compared to identify or detect the face through embedding vectors. The distribution platform use Anaconda and python 3.5 software in processing face detection, recognition and classification. The image features in the database dlib and other libraries. First face is detected and then recognized with the database features and matching using CNN model training and testing database. Finally the recognized human face is classified based on the expression in real time as Angry, fear, disgust, happy, neutral and surprise. The network architecture VGG 16 is built with CNN model for large database recognition and classification. The designed network model has honey comb 3 x 3 layers where the two connected layers have 4096 nodes with Softmax classification. The local binary model histogram is used as open CV library for detecting the human faces. The image pixels are identified by setting a threshold and the end result is represented in form of a binary number. In order to perform this LBPH uses 4 parameters like radio, neighbors, Grid X and Y. Here the radio button used defines the local radius in circular manner for the binary model. The circular local binary model represents the number of sample points as neighbors. The X and Y dimensions of grid defines the number of horizontal and vertical direction cells. The circular binary model is formed to the marked areas with appropriate label to receive the test data from machine. To perform the procedural result of face detection first create a folder and name it and write two python files create_data.py face_recognize.py then haarcascade_frontalface_default.xml library set and run complete files from Integrated development environment of python.

1.3.1 Advantages :

- Better Security
- High Accuracy
- Automated Identification.

1.4 MODULES OF PROJECT

1.4.1 Upload image from dataset :

Using this module we will upload image where we had given in the previous dataset.

1.4.2. Pre-process & detect face in image :

Using this module it will pre-process the image and mentioned no.of human faces and if face detected then only it will predict facial expression**.**

1.4.3. Detect facial expression :

Using this module it will recognize the image from the dataset and shows the emotion with emoji based on CNN algorithm.

1.4.4. Detect facial expression from webcam :

Finally it shows the facial expression with the use of live webcam including the name of facial emotion.

2. SOFTWARE METHODOLOGIES

2.1 INTRODUCTION

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

2.2 SDLC

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

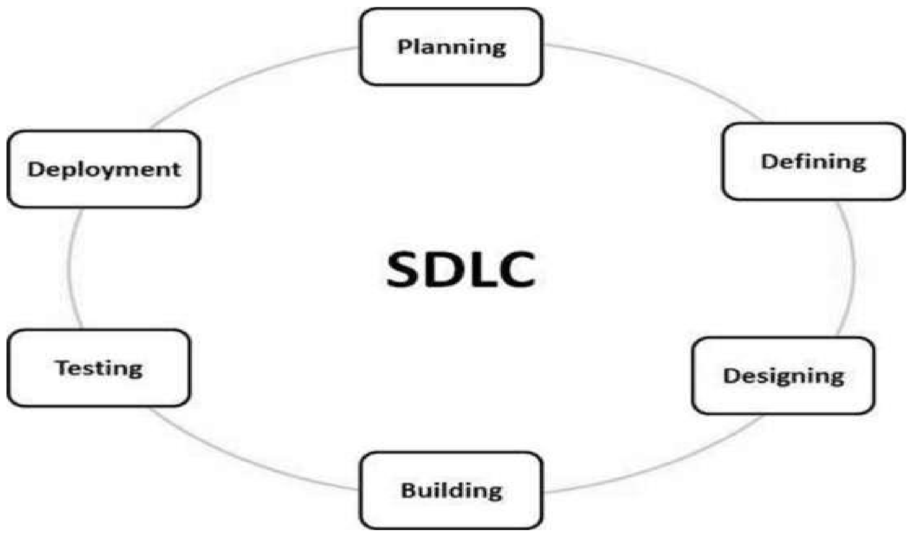


Fig.2.1: SDLC Architecture

A typical Software Development Life Cycle consists of the following stages −

2.2.1 Planning and Requirement Analysis :

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

2.2.2 Defining Requirements :

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

2.2.3 Designing the Product Architecture :

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

2.2.4 Building or Developing the Product :

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

2.2.5 Testing the Product :

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

2.2.6 Deployment in the Market and Maintenance :

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

2.3 SDLC MODELS

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry &miuns;

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

2.4 WATERFALL MODEL :

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

2.4.1 Waterfall Model – Design :

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.

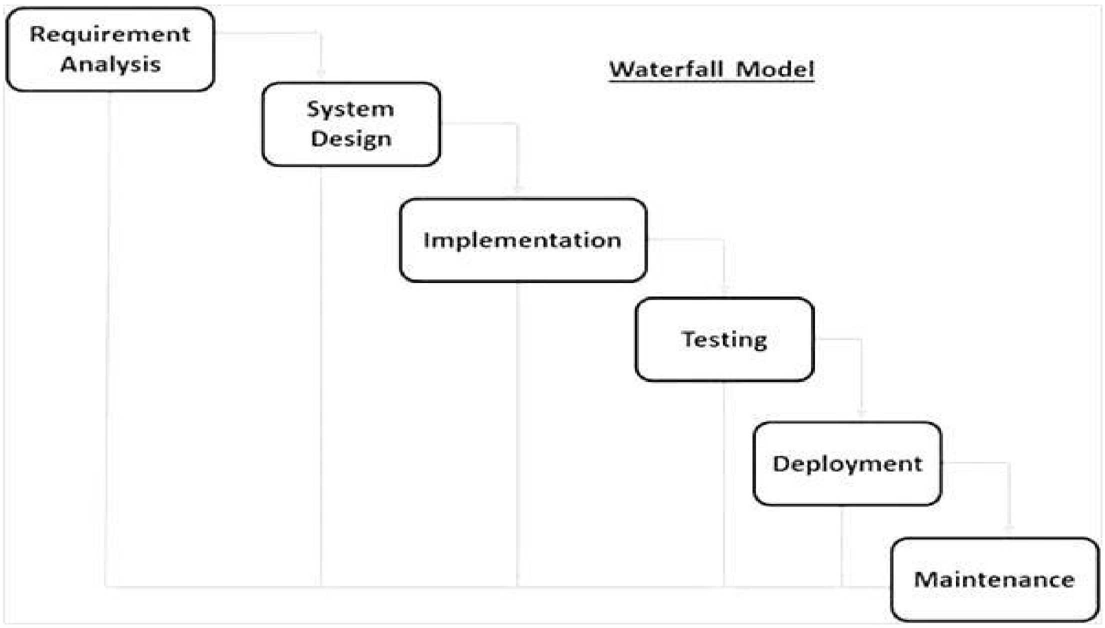


Fig.2.2: Water Fall Model

2.4.2 Stages in Waterfall Model :

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

2.4.3 Waterfall Model – Application :

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

2.4.4 Advantages :

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time

- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

2.4.5 Disadvantages :

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.

3.SOFTWARE ENVIRONMENT:

3.1. PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

3.2. DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

4. SYSTEM ANALYSIS

4.1. REQUIREMENT SPECIFICATION:

Use this Requirements Specification template to document the requirements for your product or service, including priority and approval. Tailor the specification to suit your project, organizing the applicable sections in a way that works best, and use the checklist to record the decisions about what is applicable and what isn't. The format of the requirements depends on what works best for your project. This document contains instructions and examples which are for the benefit of the person writing the document and should be removed before the document is finalized.

To regenerate the TOC, select all (CTL-A) and press F9.

Executive Summary

Project Overview :

Describe this project or product and its intended audience, or provide a link or reference to the project charter.

Purpose and Scope of this Specification :

Describe the purpose of this specification and its intended audience. Include a description of what is within the scope what is outside of the scope of these specifications. For example:

In scope :

This document addresses requirements related to phase 2 of Project A:

modification of Classification Processing to meet legislative mandate ABC.

Modification of Labor Relations Processing to meet legislative mandate ABC.

Out of Scope

The following items in phase 3 of Project A are out of scope:

modification of Classification Processing to meet legislative mandate XYZ.

modification of Labor Relations Processing to meet legislative mandate XYZ.

(Phase 3 will be considered in the development of the requirements for Phase 2, but the Phase 3 requirements will be documented separately.)

Product/Service Description

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements (provide the reasons why certain specific requirements are later specified).

Product Context :

How does this product relate to other products? Is it independent and self-contained? Does it interface with a variety of related systems? Describe these relationships or use a diagram to show the major components of the larger system, interconnections, and external interfaces.

User Characteristics :

Create general customer profiles for each type of user who will be using the product. Profiles should include:

Student/faculty/staff/other

experience

technical expertise

other general characteristics that may influence the product

Assumptions :

List any assumptions that affect the requirements, for example, equipment availability, user expertise, etc. For example, a specific operating system is assumed to be available; if the operating system is not available, the Requirements Specification would then have to change accordingly.

Constraints :

Describe any items that will constrain the design options, including

parallel operation with an old system

audit functions (audit trail, log files, etc.)

access, management and security

criticality of the application

system resource constraints (e.g., limits on disk space or other hardware limitations)

other design constraints (e.g., design or other standards, such as programming language or framework)

Dependencies :

List dependencies that affect the requirements. Examples:

This new product will require a daily download of data from X,

Module X needs to be completed before this module can be built.

Requirements

Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.

Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.

Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)

Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

Priority Definitions :

The following definitions are intended as a guideline to prioritize requirements.

Priority 1 – The requirement is a “must have” as outlined by policy/law

Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits

Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

A good requirement is:

Correct

Unambiguous (all statements have exactly one interpretation)

Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)

Consistent

Ranked for importance and/or stability

Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)

Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)

Does not specify any particular design

Traceable (cross-reference with source documents and spawned documents).

4.2 FEASIBILITY STUDY :

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Social feasibility

Economical feasibility :

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical feasibility :

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social feasibility :

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.3 FUNCTIONAL REQUIREMENTS:

The Functional Requirements Specification documents the operations and activities that a system must be able to perform.

Functional Requirements should include:

- ➢ Descriptions of data to be entered into the system
- ➢ Descriptions of operations performed by each screen
- ➢ Descriptions of work-flows performed by the system
- ➢ Descriptions of system reports or other outputs
- ➢ Who can enter the data into the system
- ➢ How the system meets applicable regulatory requirements

The Functional Requirements Specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

Examples of Functional Requirements :

Functional requirements should include functions performed by specific screens, outlines of work-flows performed by the system, and other business or compliance requirements the system must meet.

Interface requirements :

- ➢ User name accepts numeric/character data entry.
- ➢ Date Filed only accepts dates before the current date.
- ➢ Email can accept email format only.

Business Requirements :

- ➢ Data must be entered before a request can be approved.

- Clicking the Approve button moves the request to the Approval Workflow.
- All personnel using the system will be trained according to internal SOP AA-101.

Regulatory/Compliance Requirements :

- The database will have a functional audit trail.
- The system will limit access to authorized users.
- The spreadsheet can secure data with electronic signatures.

Security Requirements :

- Members of the Data Entry group can enter requests but cannot approve or delete requests.
- Members of the Managers group can enter or approve a request but cannot delete requests.
- Members of the Administrators group cannot enter but can approve requests and can delete requests.

Depending on the system being described, different categories of requirements are appropriate. System Owners, Key End-Users, Developers, Engineers, and Quality Assurance should all participate in the requirement gathering process, as appropriate to the system.

Requirements outlined in the Functional Requirements Specification are usually tested in the Operational Qualification.

4.4 NON FUNCTIONAL REQUIREMENS:

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. **Non-functional requirements** add tremendous value to business analysis. It is commonly misunderstood by a lot of people. It is important for business stakeholders, and Clients to clearly explain the requirements and their expectations in measurable terms. If the non-functional requirements are not measurable then they should be revised or rewritten to gain better clarity. For example, User stories help in mitigating the gap between developers and the user community in Agile Methodology.

Usability:

Prioritize the important functions of the system based on usage patterns. **Frequently used functions should be tested for usability**, as should complex and critical functions. Be sure to create a requirement for this.

Reliability :

Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period.

The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software.

Your goal should be a long MTBF (mean time between failures). It is defined as the average period of time the system runs before failing.

Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.

It's a good idea to also include requirements that make it easier to monitor system performance.

Performance :

What should system response times be, as measured from any point, under which circumstances?
Are there specific peak times when the load on the system will be unusually high?

Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

Supportability :

The system needs to be **cost-effective to maintain**.

Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.

5.HARDWARE & SOFTWARE REQUIREMENTS

5.1HARDWARE REQUIREMENTS:

• **System processor**	**:**	Pentium IV 2.4 GHz.
• **Hard Disk**	**:**	40 GB(min)
• **Floppy Drive**	**:**	1.44 Mb.
• **Monitor**	:	14’ Colour Monitor.
• **Mouse**	**:**	Optical Mouse.
• **Ram**	**:**	512 Mb(min)

5.2 SOFTWARE_REQUIREMENTS:

• **Operating system**	**:**	Windows 7 Ultimate or above.
• **Coding Language**	**:**	Python.
• **Front-End**	**:**	Python.
• **Designing**	**:**	Html, css, javascript.
• **Data Base**	**:**	MySQL.

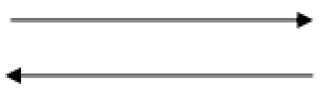
6. SYSTEM DESIGN

6.1 DATA DESIGN

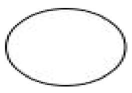
A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input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

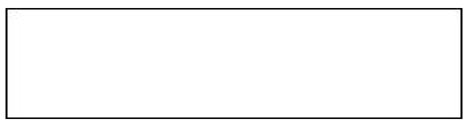
1. **Dataflow:** Data move in a specific direction from an origin to a destination.



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.



Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering. Database Designing is a part of the development process. In the linear development cycle, it is used during the system requirements phase to construct the data components of the analysis model. This model represents the major data objects and the relationship between them. It should not be confused with data analysis, which takes place in the system design phase. As in a DFD, a model of data consists of a number of symbols joined up according to certain conventions. System designers describe these conceptual modeling using symbols from a modeling method known as entity relationship analysis.

Entity Relationship Diagram :

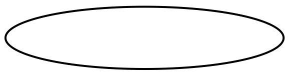
Entity relationship analysis uses three major abstractions to describe data. These are

1. Entities, which are distinct things in the enterprise.
2. Relationships, which are meaningful interactions between the objects.
3. Attributes, which are the properties of the entities and relationship.

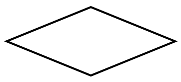
The relative simplicity and pictorial clarity of this diagramming technique may well account in large part for the widespread use of ER model. Such a diagram consists of the following major components.

E-R Diagram Components :

Rectangles, which represent the entity set.

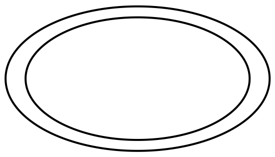


Ellipse, which represent attributes.



Diamonds, which represent relationship sets.

Lines, which link attributes to entity sets and entity sets to relationships.



Double Ellipse, which represents multi valued attributes.

Double lines, which indicates total participation of an entity in a relationship set.

Entity :

- An entity is an object that exists and is distinguishable from other objects.
- An entity may be concrete or abstract.
- An entity is a set of entities of the same type.

- Entity sets need not be disjoint.
- An entity is represented by a set of attributes.

Mapping Constraints :

An E-R diagram may define certain constraints which the contents of a database must conform.

Mapping Cardinalities :

It expresses the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of the following:

One-to-One – An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
One-to-many -An entity in A is associated with any number in B. An entity in B is associated with any number in A.
Many-to-many – Entities in A and B are associated with any number from each other.
Cardinality: It indicates that which type relationship the business rule follows is called cardinality.
Connectivity: It specifies that which type of notation the entities are connected in both sides that one side or many side.

6.2 DATA DICTIONARY

The logical characteristics of current systems data stores, including name, description, aliases, contents, and organization, identifies processes where the data are used and where immediate access to information required, Serves as the basis for identifying database requirements during system design.

Uses of Data Dictionary :

- To manage the details in large systems.
- To communicate a common meaning for all system elements.
- To Document the features of the system.
- To facilitate analysis of the details in order to evaluate characteristics and determine where system changes should be made.
- To locate errors and omissions in the system.

6.3 UML DIAGRAMS

It is a language to specifying, visualizing and constructing the artifacts of software system as well as for business models. UML was originally motivated by the desire to standardize the disparate notational system and approaches to software design developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1994-95. The UML notation is useful for graphically depicting Object Oriented Analysis and Object Oriented Design modules. The unified modeling language is a standard language for specifying, Visualizing, Constructing and documenting the software system and its components. It is a graphical language that provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain and control information about the systems.

An Overview of UML

The Unified Modeling Language is a language for

- Visualizing.
- Specifying.
- Constructing.
- Documenting.

UML Models

User model view :

- This view represents the system from the user's perspective.
- The analysis representation describes a usage scenario from the end-users perspective.

Structural model view :

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structures.

Behavioral model view :

It represents the dynamic of Behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation model view :

In this the structural and behavioral as parts of the system are represented as they are to be built.

Environmental model view :

In this the structural and Behavioral aspects of the environment in which the system is to be implemented are represented. UML is specifically constructed through two different domains they are

- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the Bahavioural modeling, implementation modeling and environmental model views.

A Conceptual model of UML :

- The three major elements of UML are
- The UML's basic building blocks.
- The rules that dictate how those building blocks may be put together.
- Some common mechanisms that apply throughout the UML.

Basic building blocks of the UML :

The vocabulary of UML encompasses three kinds of building blocks

- Things.
- Relationships.
- Diagrams.

Things :

Things are the abstractions that are first-class citizens in a model. Relationships tie these things together. Diagrams group the interesting collection of things. There are four kinds of things in the UML

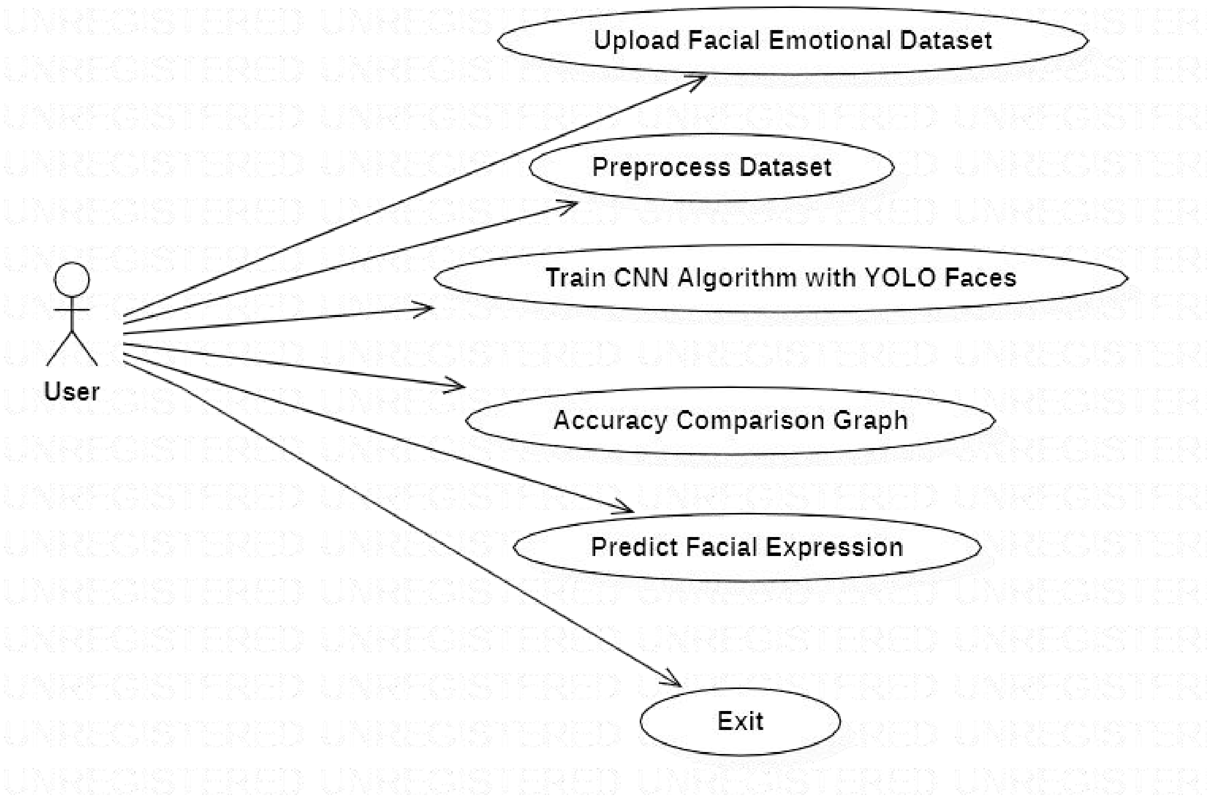
- Structural things
- Behavioral things
- Grouping things
- Annotational things

Structural Things :

Structural things are the nouns of the UML models. These are mostly static parts of the model, representing elements that are either conceptual or physical. In all, there are seven kinds of Structural things.

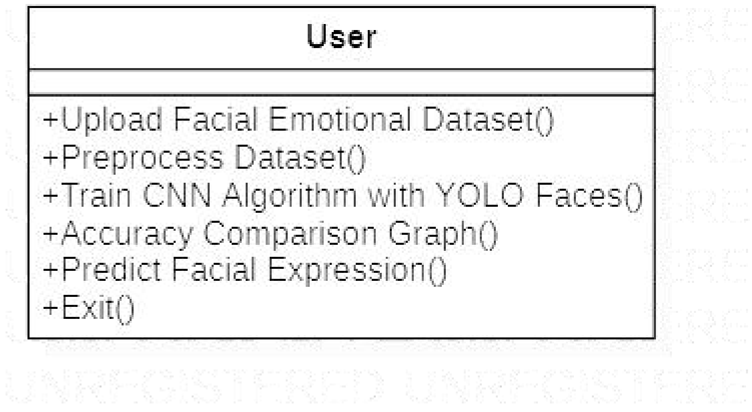
6.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.


Upload Facial Emotional Dataset
Preprocess Dataset
Train CNN Algorithm with YOLO Faces
User
Accuracy Comparison Graph
Predict Facial Expression
Exit

6.3.2 CLASS DIAGRAM:

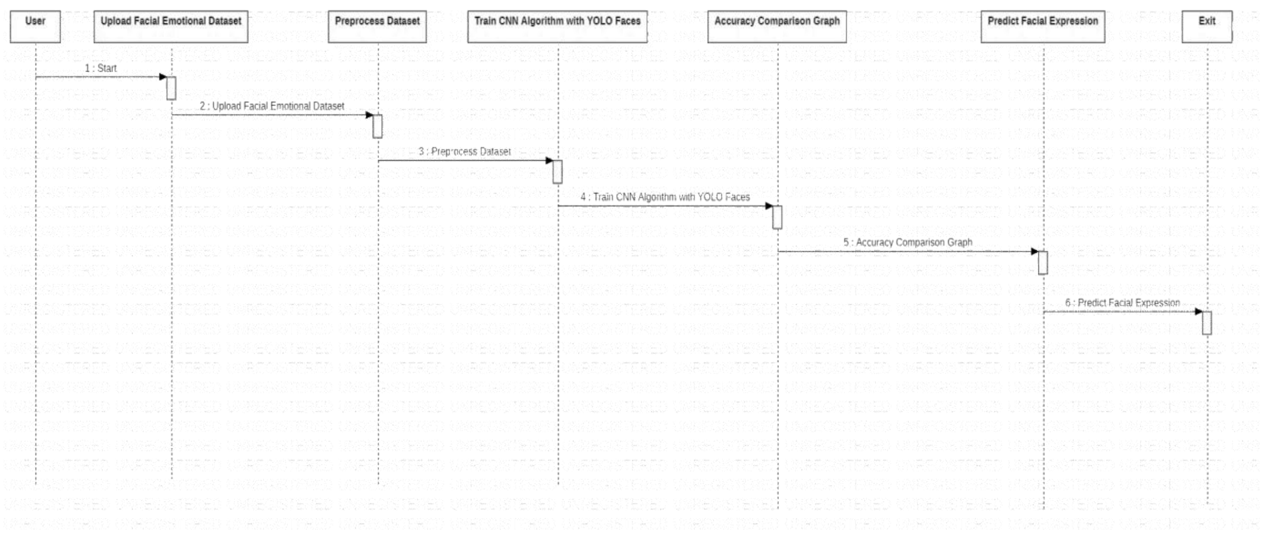
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.


User
+Upload Facial Emotional Dataset()
+Preprocess Dataset()
+Train CNN Algorithm with YOLO Faces()
+Accuracy Comparison Graph()
+Predict Facial Expression()
+Exit()

6.3.3 SEQUENCE DIAGRAM:

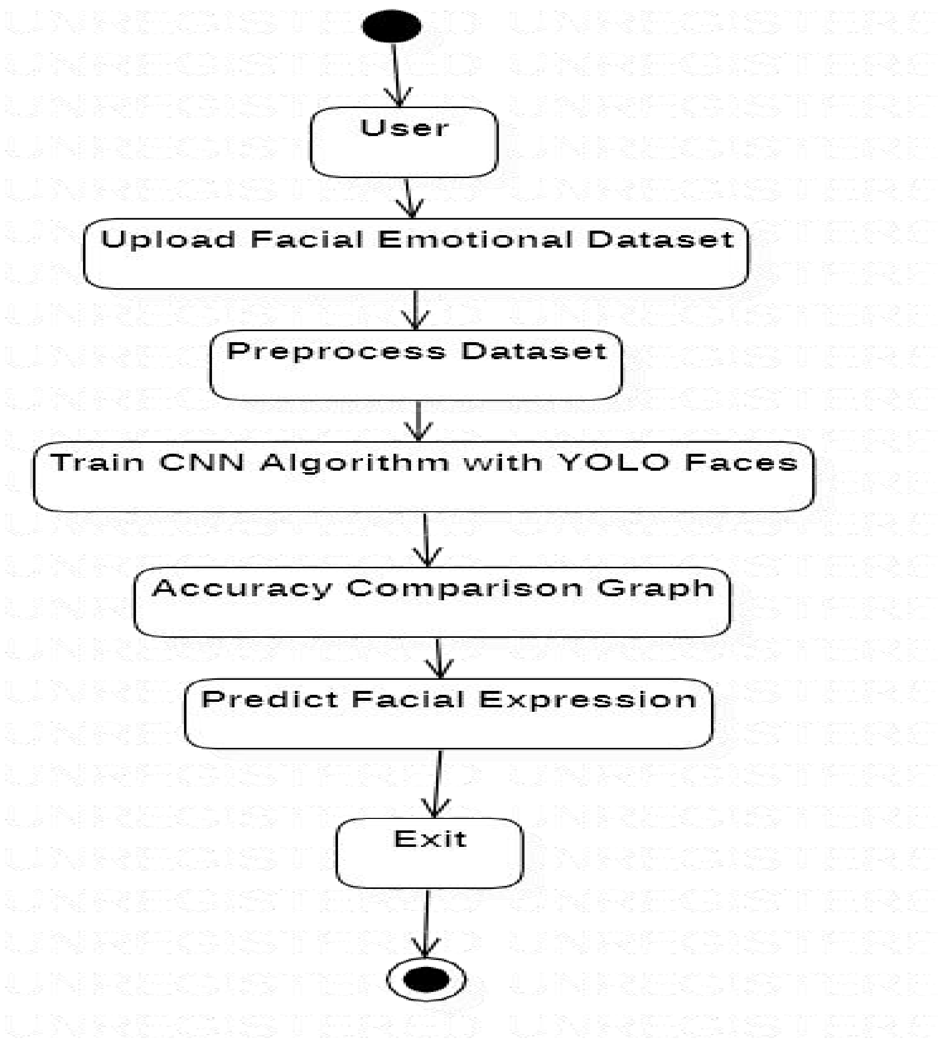
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event

scenarios, and timing diagrams.


User
Upload Facial Emotional Dataset
Preprocess Dataset
Train CNN Algorithm with YOLO Faces
Accuracy Comparison Graph
Predict Facial Expression
Exit
1 : Start
2 : Upload Facial Emotional Dataset
3 : Preprocess Dataset
4 : Train CNN Algorithm with YOLO Faces
5 : Accuracy Comparison Graph
6 : Predict Facial Expression

6.3.4 ACTIVITY DIAGRAM:

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
Upload Facial Emotional Dataset
Preprocess Dataset
Train CNN Algorithm with YOLO Faces
Accuracy Comparison Graph
Predict Facial Expression
Exit

7. IMPLEMENTATION & RESULTS

7.1 INTRODUCTION:

7.1.1 PYTHON

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.

7.1.2 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Researc0h 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.

7.1.3 Python Features

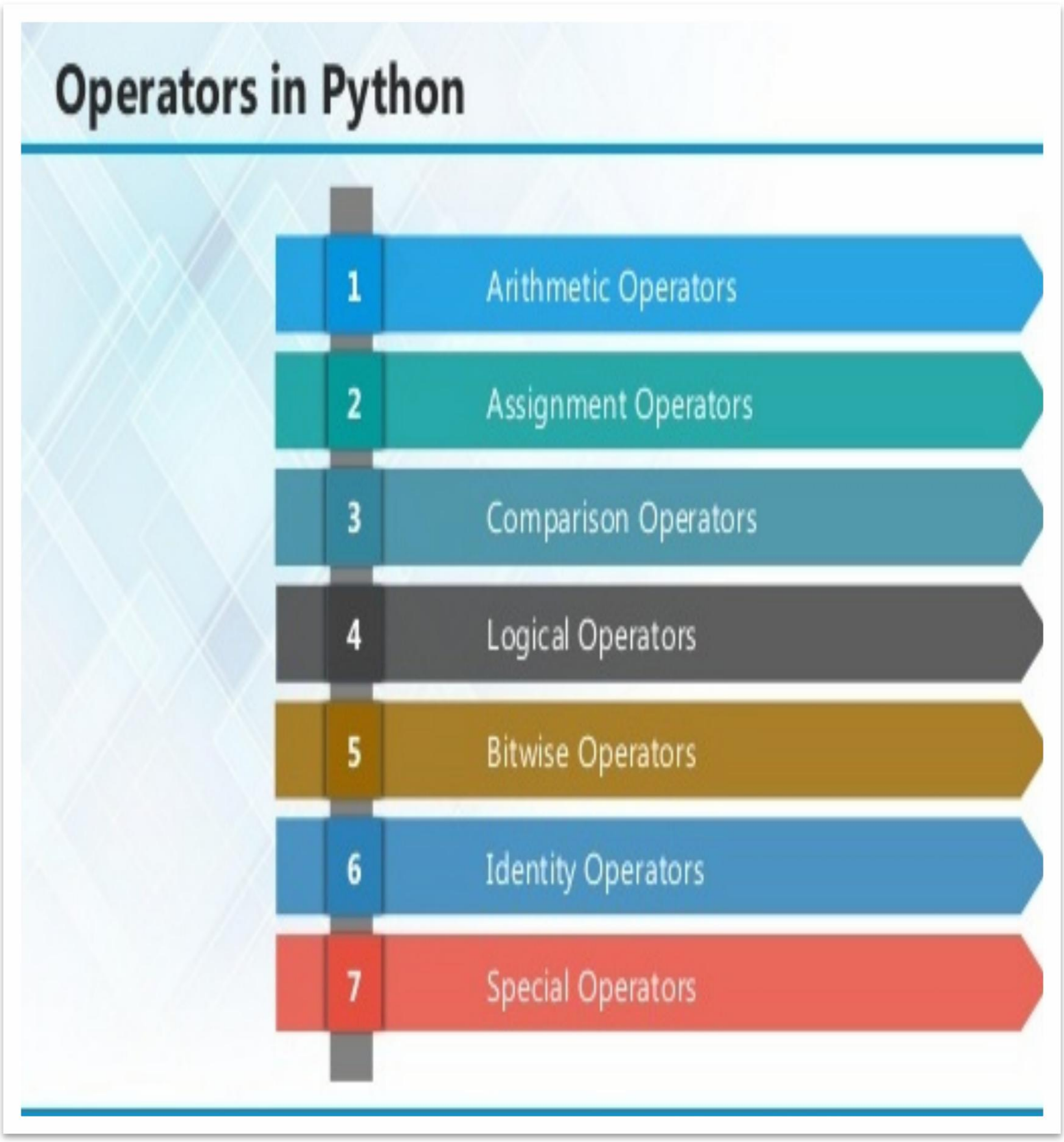
Python's features include:

- **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.

Python has a big list of good features:

- 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, C++, COM, ActiveX, CORBA, and Java.



Operators in Python

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Identity Operators
7. Special Operators

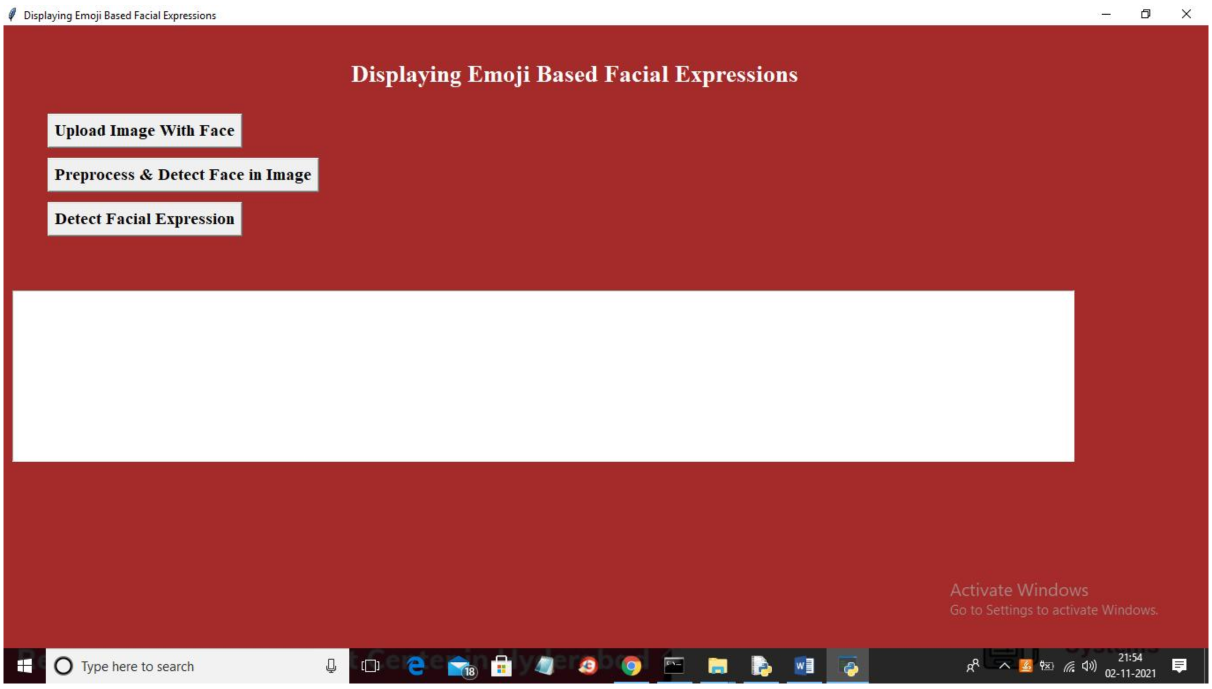
7.2 SAMPLE SOURCE CODE:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from imutils import paths
import numpy as np
from collections import defaultdict
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import imutils
import cv2
import numpy as np
import sys
from tkinter import ttk
import os


main = tkinter.Tk()
main.title("Displaying Emoji Based Facial Expressions")
main.geometry("1200x1200")

global value
global filename
global faces
global frame
detection_model_path =
'models/haarcascade_frontalface_default.xml'
emotion_model_path = 'models/_mini_XCEPTION.106-0.65.hdf5'
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry","disgust","scared", "happy", "sad",
"surprise","neutral"]
global songslist

def upload():
    global filename
    global value
    filename = askopenfilename(initialdir = "images")
    pathlabel.config(text=filename)


def preprocess():
    global filename
    global frame
    global faces
    text.delete('1.0', END)
    orig_frame = cv2.imread(filename)
    orig_frame = cv2.resize(orig_frame, (48, 48))
    frame = cv2.imread(filename,0)
    faces =
face_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbor
s=5,minSize=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)
```

```
    text.insert(END,"Total number of faces detected : 
"+str(len(faces)))

def detectExpression():
    global faces
    if len(faces) > 0:
       faces = sorted(faces, reverse=True,key=lambda x: (x[2] - 
x[0]) * (x[3] - x[1]))[0]
       (fX, fY, fW, fH) = faces
       roi = frame[fY:fY + fH, fX:fX + fW]
       roi = cv2.resize(roi, (48, 48))
       roi = roi.astype("float") / 255.0
       roi = img_to_array(roi)
       roi = np.expand_dims(roi, axis=0)
       preds = emotion_classifier.predict(roi)[0]
       emotion_probability = np.max(preds)
       label = EMOTIONS[preds.argmax()]
       img = cv2.imread('Emoji/'+label+".png")
       img = cv2.resize(img, (600,400))
       cv2.putText(img, "Facial Expression Detected As : "+label, 
(10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
       cv2.imshow("Facial Expression Detected As : "+label, img)
       cv2.waitKey(0)
    else:
       messagebox.showinfo("Facial Expression Prediction 
Screen","No face detceted in uploaded image")

def detectfromvideo(image):
    result = 'none'
    temp = image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_detection.detectMultiScale(gray,scaleFactor=1.1, 
minNeighbors=5, minSize=(30, 30), flags = cv2.CASCADE_SCALE_IMAGE)
    print("Found {0} faces!".format(len (faces)))
    output = "none"
    if len(faces) > 0:
        faces = sorted(faces, reverse=True,key=lambda x: (x[2] - 
x[0]) * (x[3] - x[1]))[0]
        (fX, fY, fW, fH) = faces
        roi = temp[fY:fY + fH, fX:fX + fW]
        roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        roi = cv2.resize(roi, (48, 48))
        roi = roi.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)
        preds = emotion_classifier.predict(roi)[0]
        emotion_probability = np.max(preds)
        label = EMOTIONS[preds.argmax()]
        output = label
    return output

def detectWebcamExpression():
    cap = cv2.VideoCapture(0)
    while True:
        _, img = cap.read()
        height, width, channels = img.shape
        result = detectfromvideo(img)
        if result != 'none':
            print(result)
            img1 = cv2.imread('Emoji/'+result+".png")
            img1 = cv2.resize(img1, (width,height))
            cv2.putText(img1, "Facial Expression Detected As :
"+result, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)
            cv2.imshow("Emoji Output",img1)
        cv2.putText(img, "Facial Expression Detected As :
"+result, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)
        cv2.imshow("Facial Expression Output", img)
        if cv2.waitKey(650) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()

font = ('times', 20, 'bold')
title = Label(main, text='Displaying Emoji Based Facial
Expressions')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=80)
title.place(x=5,y=5)

font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload Image With Face",
command=upload)
upload.place(x=50,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=300,y=100)

preprocessbutton = Button(main, text="Preprocess & Detect Face in
Image", command=preprocess)
preprocessbutton.place(x=50,y=150)
preprocessbutton.config(font=font1)

emotion = Button(main, text="Detect Facial Expression",
command=detectExpression)
emotion.place(x=50,y=200)
emotion.config(font=font1)

emotion = Button(main, text="Detect Facial Expression from
WebCam", command=detectWebcamExpression)
emotion.place(x=50,y=250)
emotion.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=10,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=300)
text.config(font=font1)

main.config(bg='brown')
main.mainloop()
```

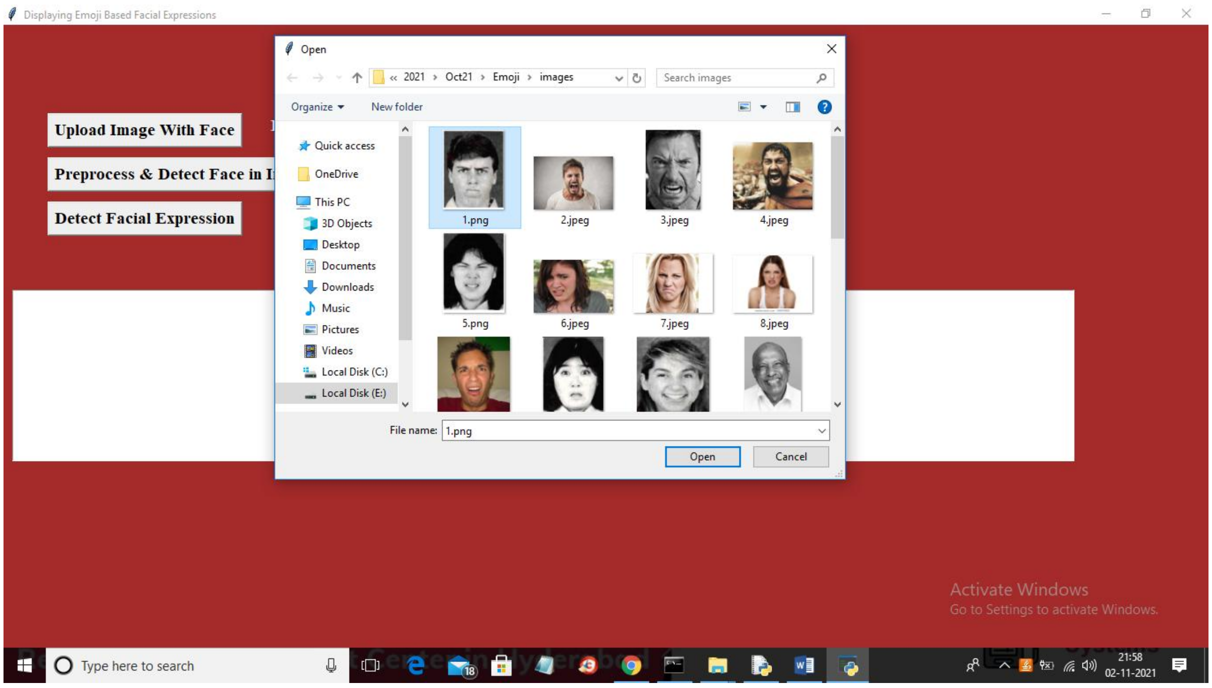
7.3 SCREENSHOTS:

To run project double click on 'run.bat' file to get below screen.


Displaying Emoji Based Facial Expressions
Displaying Emoji Based Facial Expressions
Upload Image With Face
Preprocess & Detect Face in Image
Detect Facial Expression
Activate Windows
Go to Settings to activate Windows.
Type here to search
21:54
02-11-2021

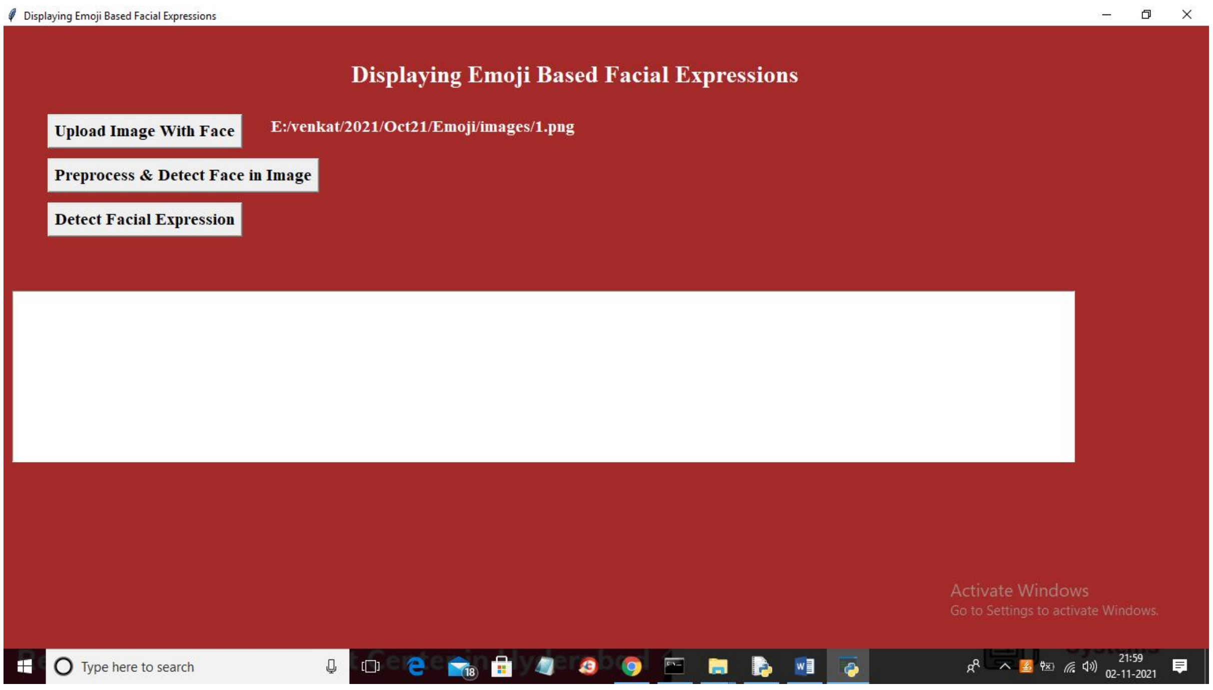
Screen 1 : Home Page

In above screen click on 'Upload Image with Face' button to upload image



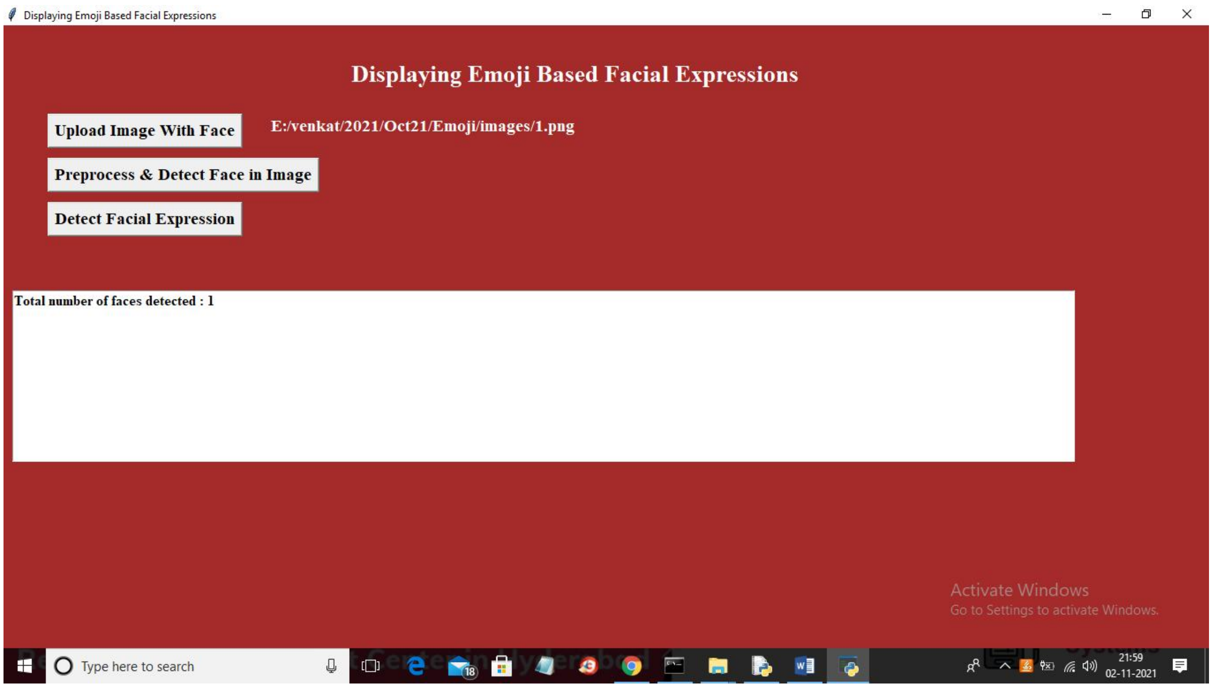
Screen 2 : Uploading Image

In above screen selecting and uploading '1.png' file and then click on 'Open' button to load image and to get below output


Displaying Emoji Based Facial Expressions
Displaying Emoji Based Facial Expressions
Upload Image With Face
E:/venkat/2021/Oct21/Emoji/images/1.png
Preprocess & Detect Face in Image
Detect Facial Expression
Activate Windows
Go to Settings to activate Windows.
Type here to search
21:59
02-11-2021

Screen 3 : Preprocess & Detect Face in Image

In above screen image is loaded and now click on ‘Preprocess & Detect Face in Image’ button to check whether image contains human face or not


Displaying Emoji Based Facial Expressions
Upload Image With Face
Preprocess & Detect Face in Image
Detect Facial Expression
Activate Windows
Go to Settings to activate Windows.
Type here to search
21:59
02-11-2021

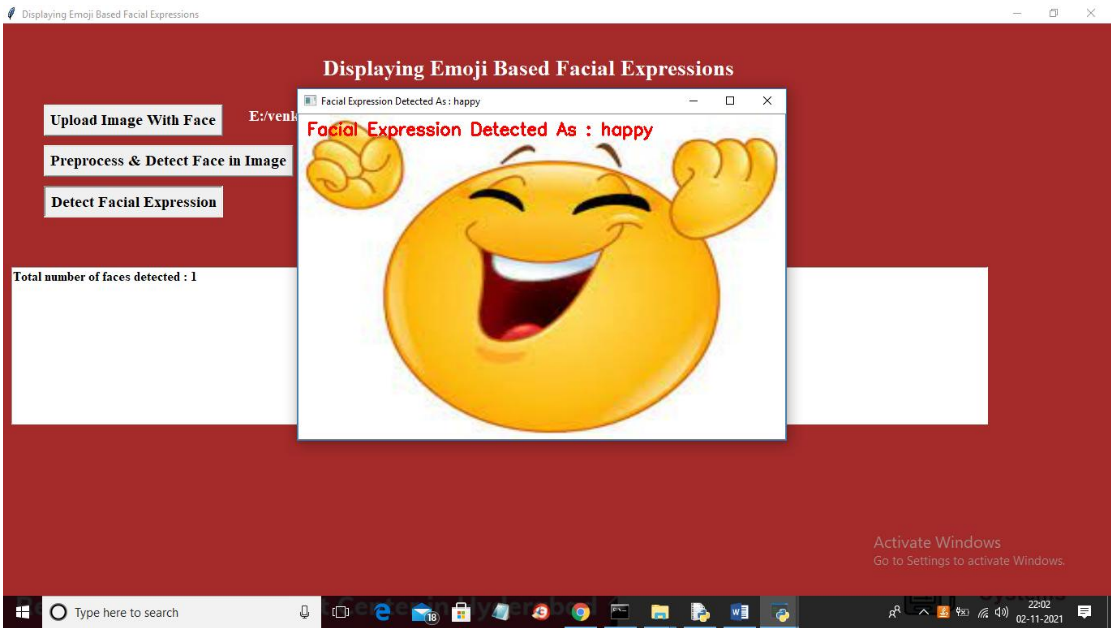
Displaying Emoji Based Facial Expressions

E:/venkat/2021/Oct21/Emoji/images/1.png

Total number of faces detected : 1

Screen 4 : Detect Facial Expression

In above screen in text are we can see one face is detected and now click on 'Detect Facial Expression' button to detect facial expression and to get below output.


Displaying Emoji Based Facial Expressions
Displaying Emoji Based Facial Expressions
Upload Image With Face
Preprocess & Detect Face in Image
Detect Facial Expression
E:/venk
Facial Expression Detected As : happy
Type here to search
Activate Windows
Go to Settings to activate Windows.
22:02
02-11-2021

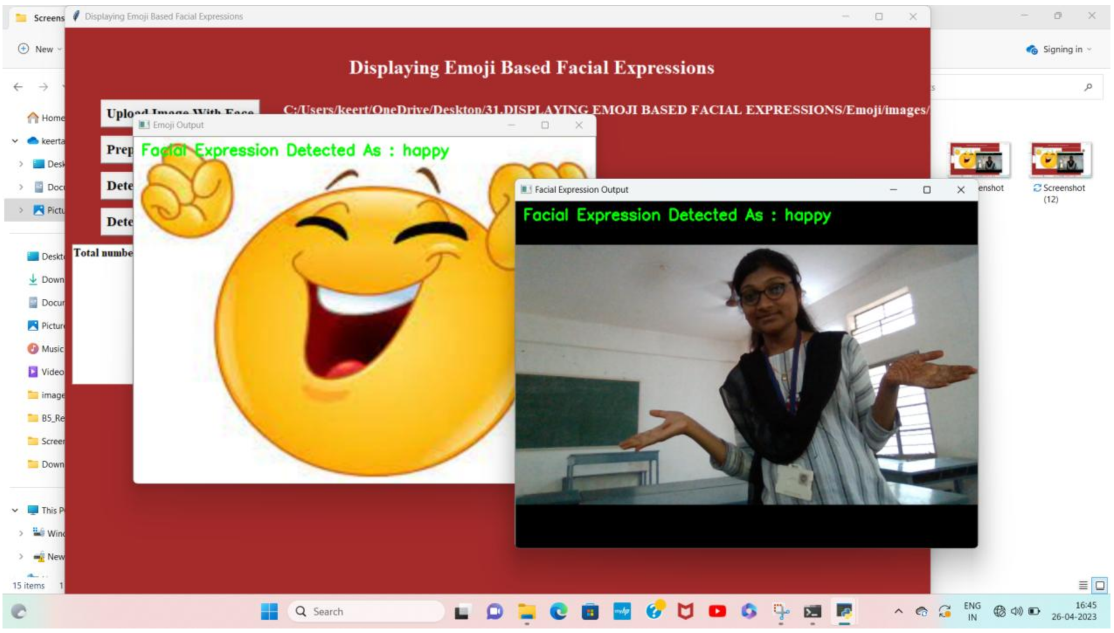
Facial Expression Detected As : happy

Total number of faces detected : 1

Screen 5 : Displaying Emoji

In above screen in uploaded image expression detected as HAPPY and then HAPPY Emoji is displayed and below is the other output

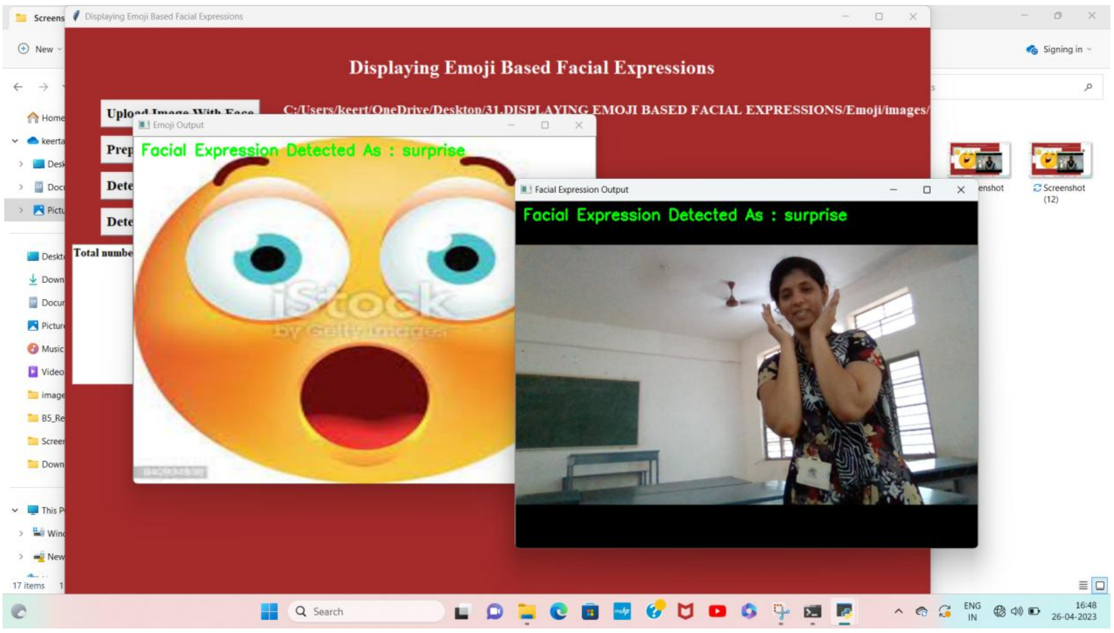
In the below screen we will observe the final **detecting the HAPPY EMOJI with real time example of Human face**


Displaying Emoji Based Facial Expressions
Upload Image With Face
C:/Users/keert/OneDrive/Desktop/31.DISPLAYING EMOJI BASED FACIAL EXPRESSIONS/Emoji/images/
Emoji Output
Facial Expression Detected As : happy
Facial Expression Output
Facial Expression Detected As : happy
Total numbe
16:45
26-04-2023

Screen 6 : Displaying Emoji with Human face

In above screen facial expression detected as HAPPY and same HAPPY EMOJI is displayed.

In the below screen we will observe the final **detecting the SURPRISE EMOJI with real time example of Human face**


Displaying Emoji Based Facial Expressions
C:/Users/keert/OneDrive/Desktop/31.DISPLAYING EMOJI BASED FACIAL EXPRESSIONS/Emoji/images/
Emoji Output
Facial Expression Detected As : surprise
Facial Expression Output
Facial Expression Detected As : surprise

Screen 7 : Displaying Human Face with live webcam

8. SYSTEM TESTING

8.1 INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 TYPES OF TESTING

8.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

8.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.2.3 Functional Test: Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.2.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.2.7 Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.2.8 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

8.2.9 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

9. INPUT &OUTPUT DESIGN

9.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

9.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

10. CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION:

This paper proposed a low cost and functionality method for real time classification seven different emotions by facial expression based on LeNet CNN architecture. In this study, facial expression pictures, which can be said has a small number, were successfully trained in CNN and achieved high classification accuracy. Using the Haar Cascade library, the effect of unimportant pixels which is outside facial expressions was reduced. In addition, single-depth placement of the pixels in the pictures to networks did not only result in loss of success rate, but also reduced training time and number of networks. Using a custom database has provided higher validation and test accuracy than training in existing databases. The real-time test model has the functionality to query each image that occurs in every second. Emotion estimation from facial expressions is the area of interest of many researchers in the literature. It is hoped that this study will be a source of studies that will help in the early detection of diseases from facial expressions and also studies of consumer behavior analysis.

10.2 FUTURE ENHANCEMENT:

The project is still continuing and is expected to produce successful outcomes in the area of emotion recognition. We expect to make the system and the source code available for free. In addition to that, it is our intention to extend the system to recognize emotions in video sequences. However, the results of classifications and the evaluation of the system are left out from the paper since the results are still being tested. We hope to make the results announced with enough time so that the final system will be available for this paper's intended audience.

11. REFERENCES

[1] M. Cabanac, "What is emotion?," Behavioural processes, vol. 60, pp. 69-83, 2019.

[2] R. Roberts, "What an Emotion Is: a Sketch," The Philosophical Review, vol. 97, 2020.

[3] E. Shouse, "Feeling, emotion, affect," M/c journal, vol. 8, no. 6, p. 26, 2020.

[4] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1D & 2D CNN LSTM networks," Biomedical Signal Processing and Control, vol. 47, pp. 312-323, 2019.

[5] J. M. B. Fugate, A. J. O'Hare, and W. S. Emmanuel, "Emotion words: Facing change," Journal of Experimental Social Psychology, vol. 79, pp. 264-274, 2018.

[6] J. P. Powers and K. S. LaBar, "Regulating emotion through distancing: A taxonomy, neurocognitive model, and supporting meta-analysis," Neuroscience & Biobehavioral Reviews, vol. 96, pp. 155-173, 2021.

[7] R. B. Lopez and B. T. Denny, "Negative affect mediates the relationship between use of emotion regulation strategies and general health in college-aged students," Personality and Individual Differences, vol. 151, p. 109529, 2019.

[8] S. Albanie, A. Nagrani, A. Vedaldi, and A. J. a. p. a. Zisserman, "Emotion recognition in speech using cross-modal transfer in the wild," pp. 292-301, 2020.

[9] K.-Y. Huang, C.-H. Wu, Q.-B. Hong, M.-H. Su, and Y.-H. Chen, "Speech Emotion Recognition Using Deep Neural Network Considering Verbal and Nonverbal Speech Sounds," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 5866-5870: IEEE.

[10] M. Degirmenci, M. A. Ozdemir, R. Sadighzadeh, and A. Akan, "Emotion Recognition from EEG Signals by Using Empirical Mode Decomposition," in 2018 Medical Technologies National Congress (TIPTEKNO), 2018, pp. 1-4.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, pp. 2278-2324, 1998.

[12] T. Chang, G. Wen, Y. Hu, and J. Ma, "Facial Expression Recognition Based on Complexity Perception Classification Algorithm," arXiv e-prints, Accessed on: February 01, 2020

https://ui.adsabs.harvard.edu/abs/2020arXiv180300185C

[13] K. Clawson, L. Delicato, and C. Bowerman, "Human Centric Facial Expression

Recognition" 2021.

12. ANNEXURE

1. Install Python 3.7.0

2. Install Opencv

3. Open the file,run the program

4. It will display home page