**TABLE OF CONTENTS**

**TITLE PAGE NO**

ABSTRACT i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

LIST OF ACRONYMS iv

1. [INTRODUCTION 1](#_TOC_250015)
   1. [PROJECT OVERVIEW 1](#_TOC_250014)
   2. [PROJECT OBJECTIVES 3](#_TOC_250013)
   3. [SCOPE OF THE PROJECT 3](#_TOC_250012)
   4. [ORGANIZATION OF CHAPTERS 4](#_TOC_250011)
      1. [INTRODUCTION 4](#_TOC_250010)
      2. [LITERATURE SURVEY 4](#_TOC_250009)
      3. SOFTWARE & HARDWARE REQUIREMENTS 4
      4. [SOFTWARE DEVELOPMENT ANALYSIS 4](#_TOC_250008)
      5. [PROJECT SYSTEM DESIGN 5](#_TOC_250007)
      6. [PROJECT CODING 5](#_TOC_250006)
      7. [PROJECT TESTING 5](#_TOC_250005)
      8. [OUTPUT SCREENS 5](#_TOC_250004)
      9. CONCLUSION 5
2. [LITERATURE SURVEY 6](#_TOC_250003)
   1. [SURVEY ON BACKGROUND 6](#_TOC_250002)
   2. CONCLUSIONS ON SURVEY 8
3. SOFTWARE & HARDWARE REQUIREMENTS 9
   1. [SOFTWARE REQUIREMENTS 9](#_TOC_250001)
   2. [HARDWARE REQUIREMENTS 13](#_TOC_250000)

|  |  |
| --- | --- |
| 4. SOFTWARE DEVELOPMENT ANALYSIS | 14 |
| 4.1 OVERVIEW OF PROBLEM | 14 |
| 4.2 DEFINE THE PROBLEM | 15 |
| 4.3 MODULES OVERVIEW | 16 |
| 4.4 DEFINE THE MODULES | 16 |
| 4.5 MODULE FUNCTIONALITY | 17 |
| 5. PROJECT SYSTEM DESIGN | 18 |
| 5.1 ARCHITECTURE DIAGRAM | 18 |
| 5.2 UML DIAGRAMS | 18 |
| 6. PROJECT CODING | 26 |
| 6.1 CODING TEMPLATES | 26 |
| 6.2 OUTLINE FOR VARIOUS FILES | 42 |
| 7. PROJECT TESTING | 44 |
| 7.1 VARIOUS TEST CASES | 44 |
| 7.2 BLACK BOX TESTING | 44 |
| 7.3 WHITE BOX TESTING | 46 |
| 8. OUTPUT SCREENS | 50 |
| 8.1 INPUT SCREENS | 49 |
| 8.2 OUTPUT SCREEN | 52 |
| 9. EXPERIMENTAL RESULTS | 54 |
| 10. CONCLUSION & FUTURE ENHANCEMENT | 55 |
| 11. REFERENCES | 56 |

**ABSTRACT**

Recently, the popularity of automated and unmanned restaurants has increased. Due to the absence of staff, there is no direct perception of the customers' impressions in order to find out what their experiences with the restaurant concept are like. For this purpose, this paper presents a rating system based on facial expression recognition with pre-trained convolutional neural network (CNN) models. For interactive human and computer interface (HCI) it is important that the computer understand facial expressions of human. With HCI the gap between computers and humans will reduce. The computers can interact in more appropriate way with humans by judging their expressions. There are various techniques for facial expression recognition which focuses on getting good results of human expressions and then the food is supposed to be rated. Currently, three expressions (satisfied, neutral and disappointed) are provided by the scoring system

**FIGURE NO TITLE PAGE NO**

**1.1.1 EVOLUTION ON FER 2**

**4.1.1 CASCADE STRUCTURE OF NETWORK 14**

**5.1.1 ARCHITECTURE DIAGRAM 18**

* + - 1. **USE CASE DIAGRAM FOR USER 20**
      2. **USE CASE DIAGRAM FOR ADMIN 21**
      3. **SEQUENCE DIAGRAM 22**
      4. **COMPONENT DIAGRAM 23**
      5. **ACTIVITY DIAGRAM 24**
      6. **CLASS DIAGRAM 25**
    1. **BLACK BOX STRUCTURE 45**
    2. **PROCESSING OF INPUT IMAGE 54**

**FIGURE NO TITLE PAGE NO**

* + 1. **USER UPLOADING RATING 50**
    2. **USER UPLOADING IMAGE 50**
    3. **ADMIN LOGIN PAGE 51**
    4. **HOME PAGE 52**
    5. **USER OUTPUT RATING PAGE 52**
    6. **ADMIN WELCOME PAGE 53**
    7. **ADMIN ACCESSING REVIEWS 53**

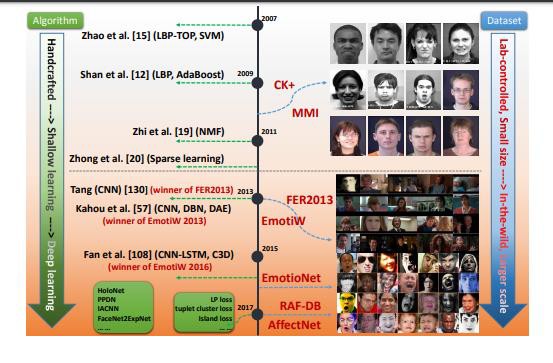
|  |  |
| --- | --- |
| <CNN> | Convolutional Neural Network |
| <HCI> | Human and Computer Interface |
| <FER> | Facial Expression Recognition |
| <FACS> | Facial Action Coding System |
| <LBP> | Local Binary Patterns |
| <NMF> | Non-negative Matrix Factorization |

# INTRODUCTION

## Project Overview

FACIAL expression is one of the most powerful, natural and universal signals for human beings to convey their emotional states and intentions. Numerous studies have been conducted on automatic facial expression analysis because of its practical importance in sociable robotics, medical treatment, driver fatigue surveillance, and many other human- computer interaction systems. In the field of computer vision and machine learning, various facial expression recognition (FER) systems have been explored to encode expression information from facial representations. As early as the twentieth century, Ekman and Friesen defined six basic emotions based on cross-culture study, which indicated that humans perceive certain basic emotions in the same way regardless of culture.

These prototypical facial expressions are anger, disgust, fear, happiness, sadness, and surprise. Contempt was subsequently added as one of the basic emotions. Recently, advanced research on neuroscience and psychology argued that the model of six basic emotions are culture-specific and not universal. Although the affect model based on basic emotions is limited in the ability to represent the complexity and subtlety of our daily affective displays and other emotion description models, such as the Facial Action Coding System (FACS) and the continuous model using affect dimensions, are considered to represent a wider range of emotions, the categorical model that describes emotions in terms of discrete basic emotions is still the most popular perspective for FER, due to its pioneering investigations along with the direct and intuitive definition of facial expressions. And in this survey, we will limit our discussion on FER based on the categorical model. FER systems can be divided into two main categories according to the feature representations: static image FER and dynamic sequence FER. In static-based methods, the feature representation is encoded with only spatial information from the current single image, whereas dynamic-based methods consider the temporal relation among contiguous frames in the input facial expression sequence. Based on these two vision based methods, other modalities, such as audio and physiological channels, have

also been used in multimodal systems to assist the recognition of expression. Most of the traditional methods have used handcrafted features or shallow learning (e.g., local binary patterns (LBP), LBP on three orthogonal planes (LBP-TOP) , non-negative matrix factorization (NMF) and sparse learning ) for FER. However, since 2013, emotion recognition competitions such as FER2013 and Emotion Recognition in the Wild (EmotiW) have collected relatively sufficient training data from challenging real-world scenarios, which implicitly promote the transition of FER from lab-controlled to in-the- wild settings. In the meanwhile, due to the dramatically increased chip processing abilities (e.g., GPU units) and well-designed network architecture, studies in various fields have begun to transfer to deep learning methods, which have achieved the state-of- the-art recognition accuracy and exceeded previous results by a large margin. Likewise, given with more effective training data of facial expression, deep learning techniques have increasingly been implemented to handle the challenging factors for emotion recognition in the wild. Figure 1 illustrates this evolution on FER in the aspect of algorithms and datasets.

**Figure 1.1.1 Evolution on FER in the aspect of algorithms and datasets**

Exhaustive surveys on automatic expression analysis have been published in recent years. These surveys have established a set of standard algorithmic pipelines for FER. However, they focus on traditional methods, and deep learning has rarely been reviewed. Very recently, FER based on deep learning has been surveyed in which is a brief review

without introductions on FER datasets and technical details on deep FER. Therefore, in this paper, we make a systematic research on deep learning for FER tasks based on both static images and videos (image sequences). We aim to give a newcomer to this filed an overview of the systematic framework and prime skills for deep FER.

Despite the powerful feature learning ability of deep learning, problems remain when applied to FER. First, deep neural networks require a large amount of training data to avoid overfitting. However, the existing facial expression databases are not sufficient to train the well-known neural network with deep architecture that achieved the most promising results in object recognition tasks. Additionally, high inter-subject variations exist due to different personal attributes, such as age, gender, ethnic backgrounds and level of expressiveness. In addition to subject identity bias, variations in pose, illumination and occlusions are common in unconstrained facial expression scenarios. These factors are nonlinearly coupled with facial expressions and therefore strengthen the requirement of deep networks to address the large intra-class variability and to learn effective expression-specific representations.

## Project Objectives

The Internet has opened the new doors for information exchange and the growth of social media has created unprecedented opportunities for citizens to publicly raise their opinions, but it has serious bottlenecks when it comes to doing analysis of these opinions. So the main objectives of our project are:

1. Avoids errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. User-friendly screens for the data entry to handle large volume of data.
3. Unmanned task can be done by just detecting the Facial emotion of the person.

## Scope of the Project

The project scope statement is a key document that provides all stakeholders with a clear understanding of why the project was initiated and defines its key goals. Our goal is to predict the Restaurant reviews, to get User opinions by analysing reviews and to know the performance of our trained CNN Model. We experiment with different linguistically

motivated models of sentiment expression, again using the results to improve the performance of our classifier. Last but not the least, human expressive behaviors in realistic applications involve encoding from different perspectives, and the facial expression is only one modality. Although pure expression recognition based on visible face images can achieve promising results, incorporating with other models into a high- level framework can provide complementary information and further enhance the robustness. For example, participants in the EmotiW challenges and Audio Video Emotion Challenges (AVEC) considered the audio model to be the second most important element and employed various fusion techniques for multimodal affect recognition. Additionally, the fusion of other modalities, such as infrared images, depth information from 3D face models and physiological data, is becoming a promising research direction due to the large complementarity for facial expressions.

## Organization of Chapters

### Introduction

### Literature Survey

This chapter deal with the study of the existing applications. The pros and cons of the existing application are carefully examined. The user expectations from the system, their experiences and the overall perception of the system is stated. The developer’s perceptions are also recorded.

**Software and Hardware Requirements**

The scope of our application is determined and software requirement specifications are hence concluded in this chapter.

### Software Development Analysis

The existing systems are given a deeper study from the developers view. The infrastructure, the logic and the implementation methods are analyzed. The drawbacks and the problems faced by the developers in developing were deeply studied

### Project System Design

The proposed system details are stated in this chapter. The infrastructure and the workflow of the application were discussed. The algorithms to be used are also stated here with their pros and cons.

### Project Coding

The implementation details are explained in this chapter. The system design is converted into code by developing the required features.

### Project Testing

The developed application is vigorously tested. The metrics that are used for testing the application are recorded. The expected and the actual behavior of the application is reported in this chapter.

### Output Screens

This chapter contains the information of the deployment of the application in the form of screenshots and navigation is explained.

**1.4.8 Conclusion**

This chapter contains the final conclusions which are drawn after deployment of the application.

# LITERATURE SURVEY

## Survey on Background

**P. Ekman and W. V. Friesen, "Constants across societies in the face and feeling." Journal of character and social brain research, vol. 17, no. 2, pp. 124–129, 1971**

Explored the subject of whether any outward appearances of feeling are widespread. Ongoing investigations indicating that individuals from educated societies connected a similar feeling ideas with a similar facial practices couldn't exhibit that probably some outward appearances of feeling are all inclusive; the way of life contrasted had all been uncovered with a portion of a similar broad communications introductions of outward appearance, and these may have shown the individuals in each culture to perceive the remarkable outward appearances of different societies. To show that individuals from a preliterate culture who had negligible introduction to proficient societies would connect a similar feeling ideas with indistinguishable facial practices from do individuals from Western and Eastern educated societies, information were accumulated in New Guinea by recounting to 342 Ss a story, demonstrating them a lot of 3 faces, and requesting that they select the face which indicated the feeling proper to the story. Ss were individuals from the Fore phonetic social gathering, which up until 12 yr. prior was a confined, Neolithic, material culture. Results give proof on the side of the theory.

**R. E. Jack, O. G. Garrod, H. Yu, R. Caldara, and P. G. Schyns, "Outward appearances of feeling are not socially all inclusive," Proceedings of the National Academy of Sciences, vol. 109, no. 19, pp. 7241–7244, 2012.**

Since Darwin's original works, the all-inclusiveness of outward appearances of feeling has stayed one of the longest standing discussions in the organic and sociologies. Quickly expressed, the comprehensiveness speculation guarantees that all people impart six fundamental inward enthusiastic states (glad, shock, dread, disturb, outrage, and tragic) utilizing a similar facial developments by uprightness of their organic and

transformative starting points [Susskind JM, et al. (2008) Nat Neurosci 11:843–850]. Here, we disprove this expected all-inclusiveness. Utilizing a one of a kind PC designs stage that consolidates generative syntaxes [Chomsky N (1965) MIT Press, Cambridge, MA] with visual discernment, we got to the imagination of 30 Western and Eastern culture people and remade their psychological portrayals of the six fundamental outward appearances of feeling. Multifaceted examinations of the psychological portrayals challenge all-inclusiveness on two separate checks. To begin with, while Westerners speak to every one of the six fundamental feelings with an unmistakable arrangement of facial developments normal to the gathering, Easterners don't. By discrediting the long- standing all-inclusiveness speculation, our information feature the ground-breaking impact of culture on molding essential practices once considered naturally designed. Therefore, our information open an extraordinary nature–sustain banter across expansive fields from transformative brain science and social neuroscience to long range interpersonal communication by means of advanced symbols.

**Y.-I. Tian, T. Kanade, and J. F. Cohn, “Recognizing action units for facial expression analysis,” IEEE Transactions on pattern analysis and machine intelligence, vol. 23, no. 2, pp. 97–115, 2001.**

Ekman and Friesen built up the Facial Action Coding System (FACS) for portraying outward appearances by activity units (AUs). Of 44 FACS AUs that they characterized, 30 AUs are anatomically identified with the withdrawals of explicit facial muscles: 12 are for upper face, and 18 are for lower face. AUs can happen either separately or in blend. At the point when AUs happen in blend they might be added substance, in which the mix doesn't change the presence of the constituent AUs, or no additive, in which the presence of the constituents changes. Even though the quantity of nuclear activity units is moderately little, more than 7,000 diverse AU mixes have been watched [30] . FACS gives the elucidating power important to portray the subtleties of outward appearance. Usually happening AUs and a portion of the added substance and no additive AU blends are appeared in Tables 1 and 2. For instance of a no additive impact, AU 4 shows up contrastingly relying upon whether it happens alone or in mix with AU 1 (as in AU 1 ‡

4). At the point when AU 4 happens alone, the foreheads are drawn together and brought down. In AU 1 ‡ 4, the foreheads are drawn together however are raised because of the activity of AU 1. AU 1 ‡ 2 is another case of no additive mixes. At the point when AU 2 happens alone, it raises the external forehead, yet in addition regularly pulls up the internal temple which brings about a fundamentally the same as appearance to AU 1 ‡ 2. These impacts of the no additive AU mixes increment the challenges of AU acknowledgment.

* + - * 1. **Conclusion on Survey**

For interactive human and computer interface (HCI) it is important that the computer understand facial expressions of human. With HCI the gap between computers and humans will reduce. The computers can interact in more appropriate way with humans by judging their expressions. There are various techniques for facial expression recognition which focuses on getting good results of human expressions in order to get reviews in the restaurants. We decided to present a rating system based on facial expression recognition with pre-trained convolutional neural network (CNN) models.

* + - 1. **SOFTWARE AND HARDWARE REQUIREMENTS**

## Software Requirements

* Operating System: Windows family
* Technology: Python 3.6 or Higher
* IDE: PyCharm

**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.

**What can python do?**

Python can be used on a server to create web applications.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

**Good to know**

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

**Python Syntax compared to other programming languages**

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Purpose**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of

typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**Functional Requirements**

Graphical User interface with the User.

**Non-functional Requirements**

* **Maintainability:** Maintainability is used to make future maintenance easier, meet new requirements. Our project can support expansion.
* **Robustness:** Robustness is the quality of being able to withstand stress, pressures or changes in procedure or circumstance. Our project also provides it.
* **Reliability:** Reliability is an ability of a person or system to perform and maintain its functions in circumstances. Our project also provides it.
* **Size:** The size of a particular application plays a major role, if the size is less then efficiency will be high. The size of database we have developed is 5.05 MB.
* **Speed:** If the speed is high then it is good. Since the no of lines in our code is less, hence the speed is high.
* **Power Consumption:** In battery-powered systems, power consumption is very important. In the requirement stage, power can be specified in terms of battery life. However, the allowable wattage can’t be defined by the customer. Since the no of lines of code is less CPU uses less time to execute hence power usage will be less.

**Input and Output Design 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**

* 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.
* 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.
* 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

**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.

1. Select methods for presenting information.
2. 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.

## Hardware Requirements

* Processer: Intel i3 or Higher
* Ram: Min 4 GB
* Hard Disk: Min 100 GB

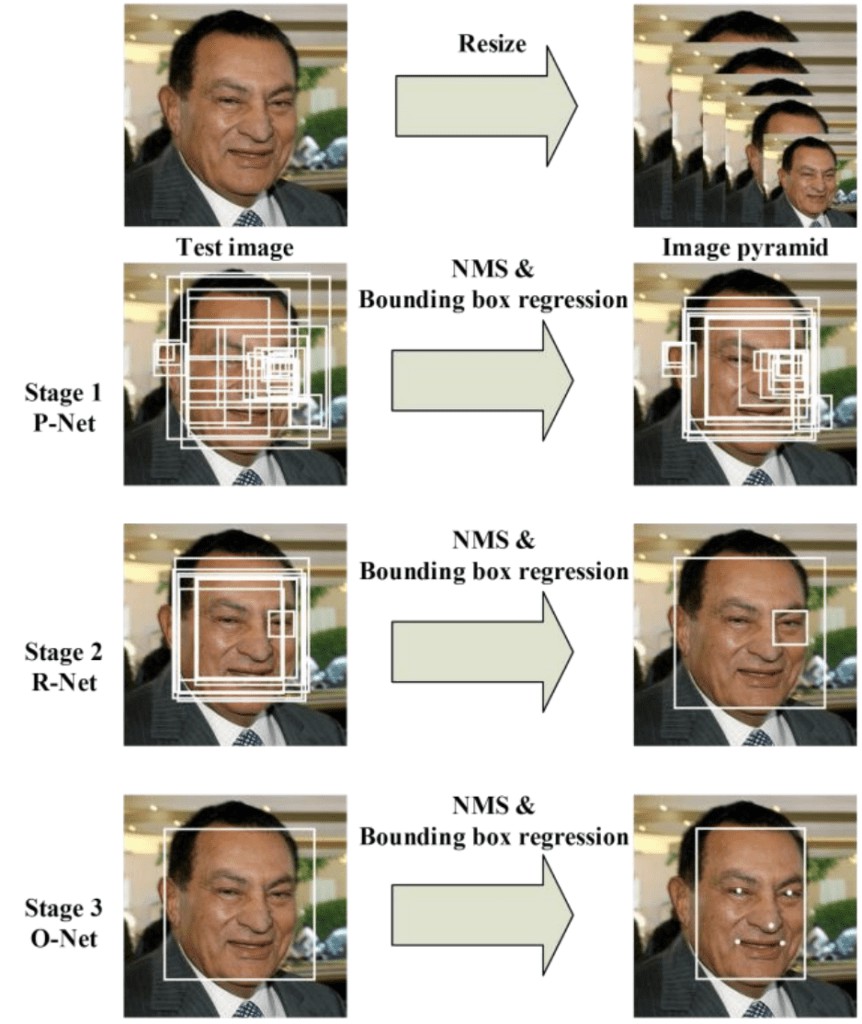
# SOFTWARE DEVELOPMENT ANALYSIS

* + - * 1. **Overview of Problem:**

The purpose of this analysis is to build a prediction model to detect the facial expression in order to predict the review on a restaurant’s food/service. To do so, we will work on various Facial Expression datasets, we will load it into Viola Jones Object detection model which makes use of Haar based classifiers. In the end, we hope to find a "best" model for predicting the customer reviews based on facial expressions.

To build the CNN model to predict the review based on customer facial expressions, following steps are performed.

The network uses a cascade structure with three networks:



**Figure 4.1.1. Cascade Structure of network**

* First the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions.
* The second model (Refine Network or R-Net) filters the bounding boxes
* The third model (Output Network or O-Net) proposes facial landmarks.

**Existing System**

As there is no staff available in unmanned restaurants, it is difficult for the restaurant management to estimate how the concept and the food is experienced by the customers. Existing rating systems, such as Google and TripAdvisor, only partially solve this problem, as they only cover a part of the customer’s opinions.

**Disadvantages of Existing System**

1. These rating systems are only used by a subset of the customers who rate the restaurant on independent rating platforms on their own initiative.
2. This applies mainly to customers who experience their visit as very positive or negative.
3. Anonymous reviews posted on independent rating platforms will often lead to a mixed perception on the restaurants making it hard to decide its genuinely.
   * + - 1. **Define the Problem:**

Let's define the problem in order to think about the solution so as to get an optimal solution. Currently there are many restaurants which are operating all around the world. But hardly restaurants know about what their customers think of the restaurant's service. Restaurants aren't aware of the reasons for their thrusts and troughs, facial recognition analysis helps them for the reason for fluctuation of their financial condition. We collect facial reviews from various customers and perform face detection followed by facial expression recognition using Viola Jones Object Detection Algorithm which is based on Haar based classifiers.

**Proposed System**

In order to solve the above problem, all customers must be motivated to give a rating. The proposed system introduces an approach for a restaurant rating system that asks every customer for a rating after their visit to increase the number of ratings as much as possible. This system can be used unmanned restaurants; the scoring system is based on facial expression detection using pre- trained convolutional neural network (CNN) models. It allows the customer to rate the food by taking or capturing a picture of his face that reflects the corresponding feelings.

**Advantages of Proposed System**

1. Compared to text-based rating system, there is much less information and no individual experience reports collected.
2. It is a simple, fast and playful rating system.
3. It gives a wider range of opinions about the experiences of the customers with the restaurant concept.
4. There is no need for any independent rating platforms.
5. Can know every person’s review who visited the restaurants.
   * + - 1. **Modules Overview:**

In Development and analysis of the data set, there are three modules involved. They are:

1. Face Detection
2. Facial Expression Recognition Classification
3. Convolutional Neural Network (CNN)
   * + - 1. **Define the Modules**

**Face Detection**

Face detection or localization is an important step for image classification since only the principal component of face such as nose, eyes, mouth are needed for classification. Face detection algorithms can be broadly classified into feature, knowledge, template and appearance

based methods. Our proposed system uses Viola Jones object detection algorithm for face localization which comes under feature based classification. Viola Jones object detection algorithm uses Haar feature based cascade classifiers. The Haar Cascade classifier is extremely important element of the face detection. The presence of the features in any of the input image is determined by the Haar features.

**Facial Expression Recognition classification:**

After learning the deep features, the final step of FER (Facial Expression Recognition) is to classify the given face into one of the basic emotion categories. Unlike the traditional methods, where the feature extraction step and the feature classification step are independent, deep networks can perform FER in an end-to-end way. Specifically, a loss layer is added to the end of the network to regulate the back-propagation error; then, the prediction probability of each sample can be directly output by the network.

**Convolutional neural network (CNN):**

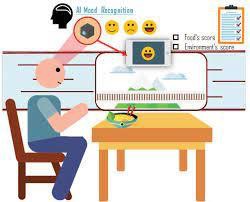
In CNN, SoftMax loss is the most common used function that minimizes the cross- entropy between the estimated class probabilities and the ground truth distribution. CNN has been extensively used in diverse computer vision applications, including FER. At the beginning of the 21st century, several studies in the FER literature found that the CNN is robust to face location changes and scale variations and behaves better than the multilayer perceptron (MLP) in the case of previously unseen face pose variations, employed the CNN to address the problems of subject independence as well as translation, rotation, and scale invariance in the recognition of facial expressions.

**4.4 Module Functionality:**

The CNN model collects the reviews from a .xml file which contains various facial expressions based on which the emotion label is detected, and the results are subjected to Viola Jones Object Detection Algorithm and Haar based classifiers to obtain the results.

* + - 1. **PROJECT SYSTEM DESIGN**
         1. **Architecture Diagram:**

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.



**Figure 5.1.1 Architecture Diagram**

* + - * 1. **UML Diagrams:**

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object -oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard

language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

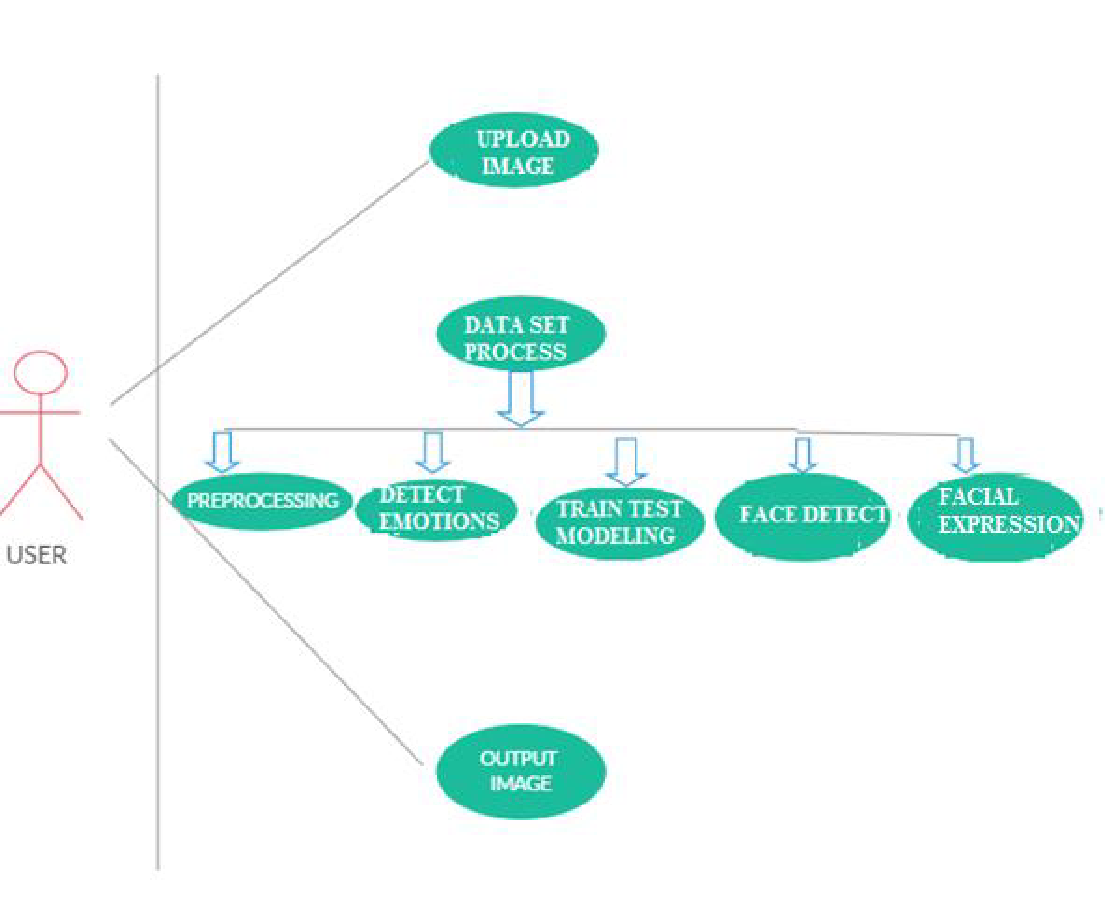
The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The Primary goals in the design of the UML are as follows:

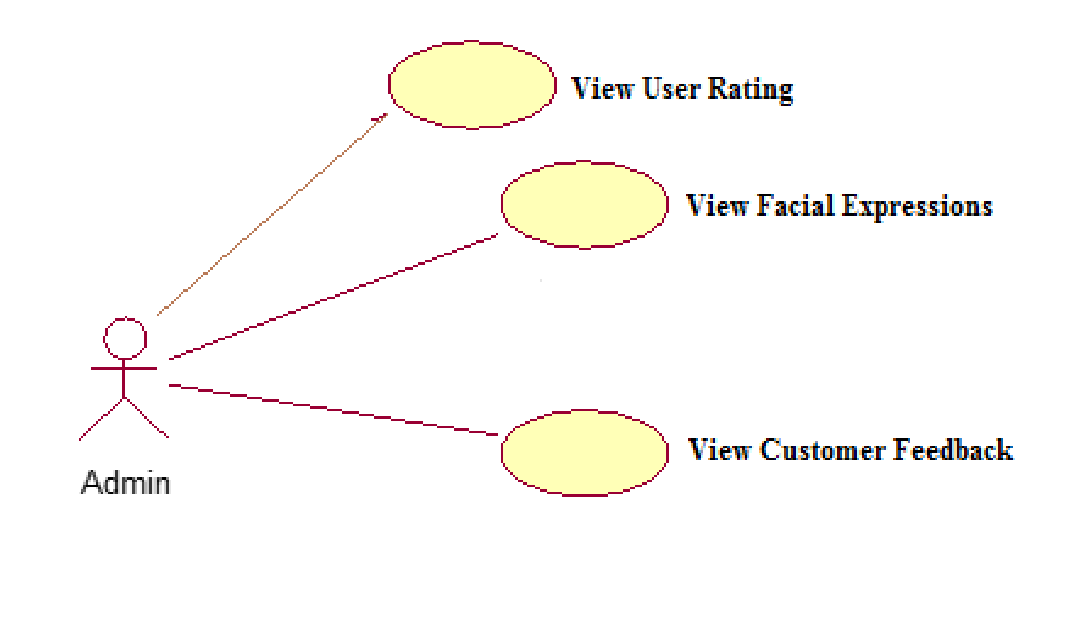
1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**Usecase 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.



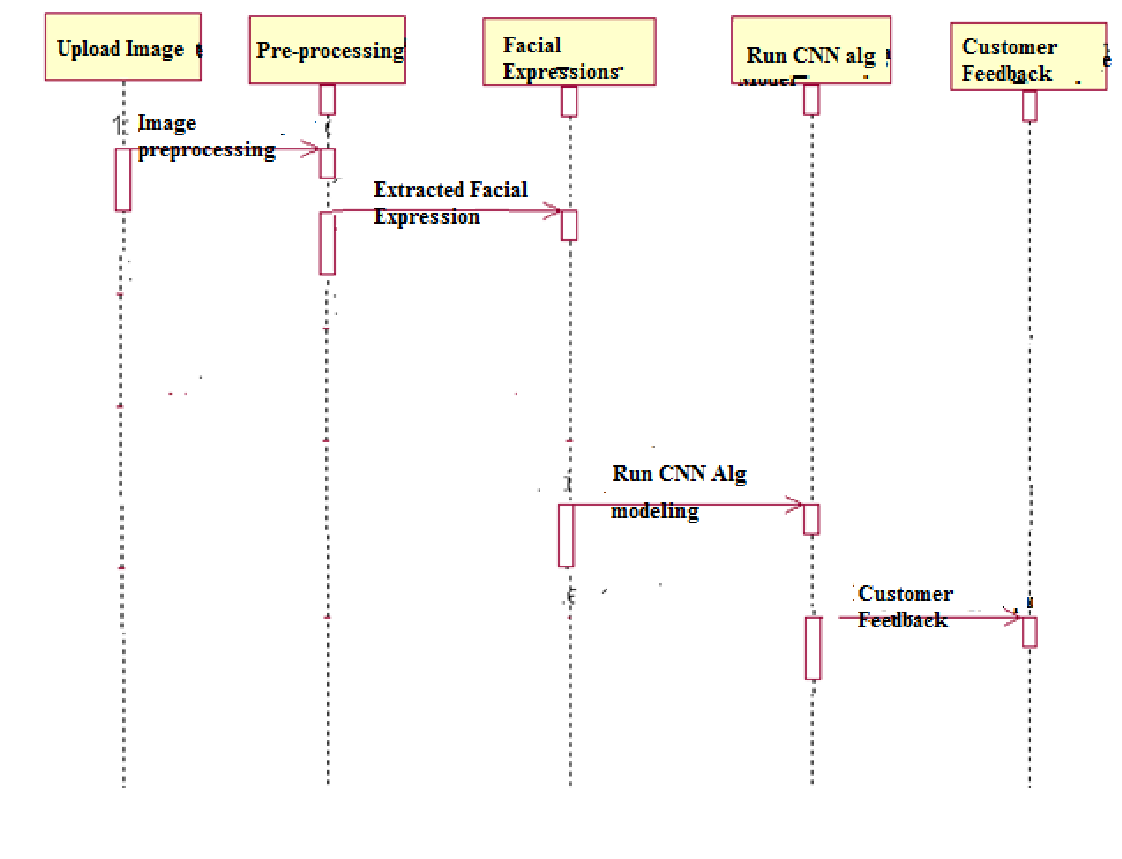
**Figure 5.2.1.1. Usecase diagram of User**



**Figure 5.2.1.2 Usecase Diagram of Admin**

**Sequence Diagram :**

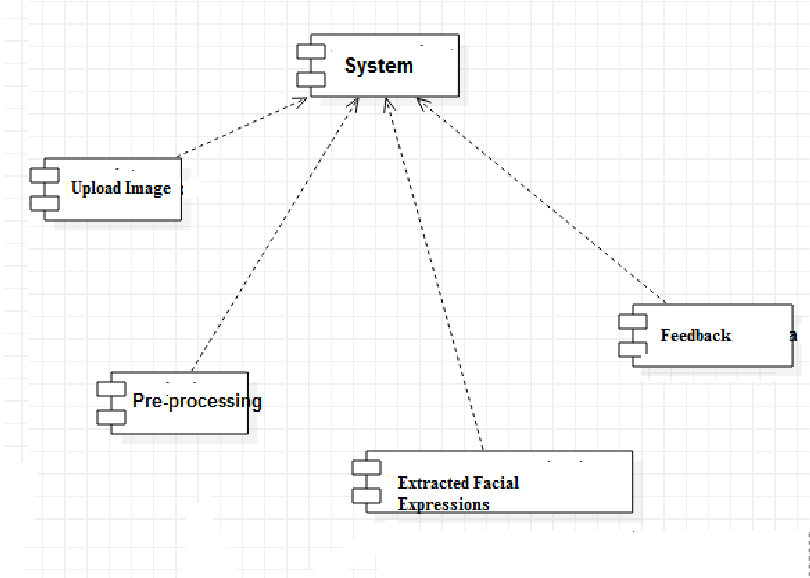
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagram.



**Figure 5.2.2.1 Sequence Diagram**

**Component Diagram:**

The component diagram extends the information given in a component notation element. One way of illustrating the provided and required interfaces by the specified component is in the form of a rectangular compartment attached to the component element.[2] Another accepted way of presenting the interfaces is to use the ball-and-socket graphic convention. A provided dependency from a component to an interface is illustrated with a solid line to the component using the interface from a "lollipop", or ball, labelled with the

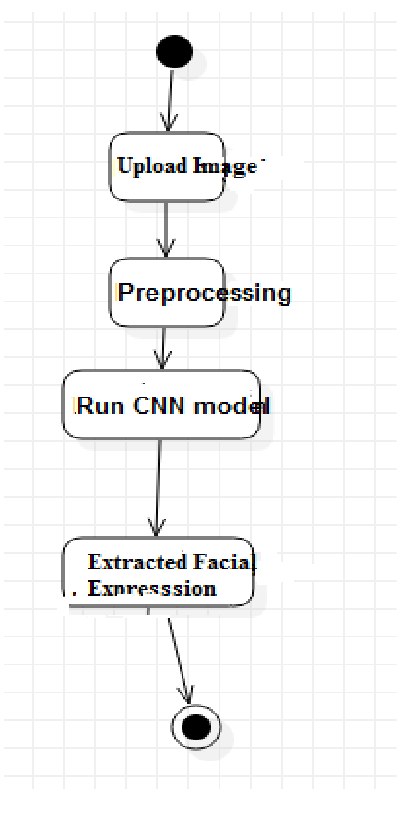
name of the interface. A required usage dependency from a component to an interface is illustrated by a half-circle, or socket, labelled with the name of the interface, attached by a solid line to the component that requires this interface. Inherited interfaces may be shown with a lollipop, preceding the name label with a caret symbol. To illustrate dependencies between the two, use a solid line with a plain arrowhead joining the socket to the lollipop.

**Figure 5.2.3.1 Component Diagram**

**Activity diagrams:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions[1] with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores. 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 are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control,

they can also include elements showing the flow of data between activities through one or more data stores.



**Fig 5.2.4.1 Activity Diagram**

**Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object- oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML

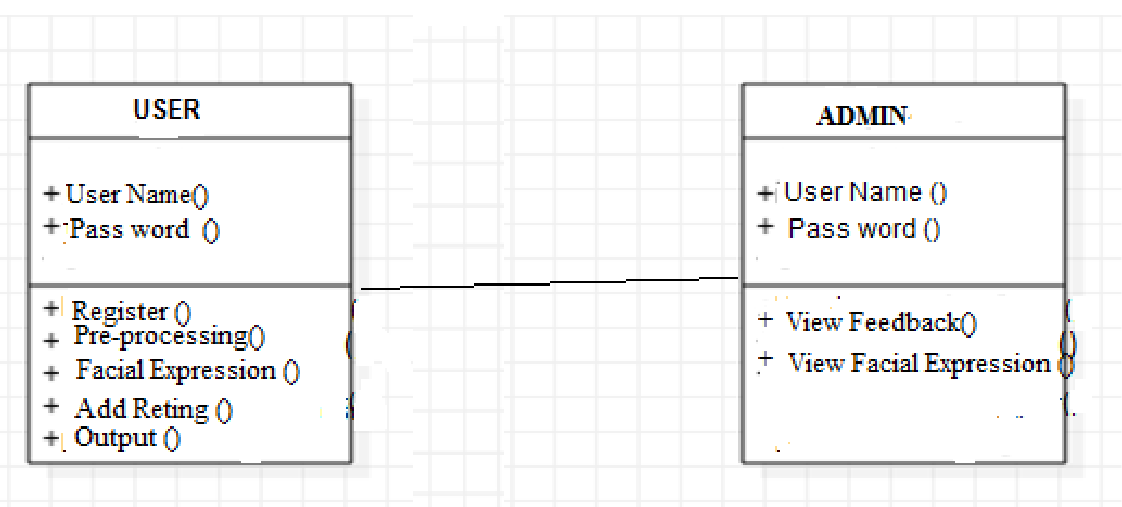


diagram in the coder community.

The purpose of the class diagram can be summarized as −

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

**Figure 5.2.5.1 Class Diagram**

* + - 1. **PROJECT CODING**
         1. **Code Templates**

**FacialApp\views.py**

from django.shortcuts import render

from django.template import RequestContext import pymysql

from django.http import HttpResponse from django.conf import settings

from django.core.files.storage import FileSystemStorage import datetime

import cv2

from keras.models import load\_model

from keras.preprocessing.image import img\_to\_array import numpy as np

def Index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def User(request):

if request.method == 'GET':

return render(request, 'User.html', {})

def Admin(request):

if request.method == 'GET':

return render(request, 'Admin.html', {})

def AdminLogin(request):

if request.method == 'POST':

username = request.POST.get('t1', False) password = request.POST.get('t2', False)

if username == 'admin' and password == 'admin':

context= {'data':'welcome '+username}

return render(request, 'AdminScreen.html', context) else:

context= {'data':'login failed'}

return render(request, 'Admin.html', context)

def ViewRating(request):

if request.method == 'GET':

strdata = '<table border=1 align=center width=100%><tr><th>Customer Name</th><th>Rating</th><th>Facial Expression</th><th>Photo</th> <th>Date & Time</th></tr><tr>'

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'facial',charset='utf8')

with con:

cur = con.cursor() cur.execute("select \* FROM rating") rows = cur.fetchall()

for row in rows:

strdata+='<td>'+row[0]+'</td><td>'+str(row[1])+'</td><td>'+row[2]+'</td><td><img src=/static/photo/'+row[0]+'.png width=200

height=200></img></td><td>'+str(row[4])+'</td></tr>' context= {'data':strdata}

return render(request, 'ViewRatings.html', context)

def Rating(request):

if request.method == 'POST' and request.FILES['t3']:

output = ''

myfile = request.FILES['t3']

name = request.POST.get('t1', False) rating = request.POST.get('t2', False) fs = FileSystemStorage()

filename = fs.save('C:/Python/Facial/Facial/FacialApp/static/photo/'+name+'.png', myfile)

now = datetime.datetime.now()

current\_time = now.strftime("%Y-%m-%d %H:%M:%S")

detection\_model\_path =

'C:/Python/Facial/Facial/FacialApp/haarcascade\_frontalface\_default.xml' emotion\_model\_path = 'C:/Python/Facial/Facial/FacialApp/\_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", "surprised","neutral"]

orig\_frame =

cv2.imread('C:/Python/Facial/Facial/FacialApp/static/photo/'+name+'.png') orig\_frame = cv2.resize(orig\_frame, (48, 48))

frame = cv2.imread(filename,0)

faces =

face\_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbors=5,minSize=(30,30

),flags=cv2.CASCADE\_SCALE\_IMAGE)

print("==================="+str(len(faces))) print(emotion\_classifier)

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()]

if label == 'happy':

output = 'Satisfied' if label == 'neutral':

output = 'Neutral'

if label == 'angry' or label == 'sad' or label == 'disgust' or label == 'scared' or label

== 'surprised':

output = 'Disappointed' print("==================="+output)

db\_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root', database = 'facial',charset='utf8')

db\_cursor = db\_connection.cursor() query = "INSERT INTO

rating(customer\_name,rating,facial\_expression,photo\_path,rating\_date) VALUES('"+name+"','"+rating+"','"+output+"','"+name+'.png'+"','"+current\_time+"')"

db\_cursor.execute(query) db\_connection.commit() print(db\_cursor.rowcount, "Record Inserted") if db\_cursor.rowcount == 1:

context= {'data':'Your Rating is : '+rating+' and Facial Expression : '+output} return render(request, 'User.html', context)

else:

context= {'data':'Error in request process'} return render(request, 'User.html', context)

**FacialApp\urls.py**

from django.urls import path from . import views

urlpatterns = [path("index.html", views.Index, name="Index"), path("User.html", views.User, name="User"), path("Rating", views.Rating, name="Rating"), path("Admin.html", views.Admin, name="Admin"),

path("AdminLogin", views.AdminLogin, name="AdminLogin"), path("ViewRating", views.ViewRating, name="ViewRating"),

]

**Facial\settings.py**

"""

Django settings for Facial project.

Generated by 'django-admin startproject' using Django 2.2.7.

For more information on this file, see https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see https://docs.djangoproject.com/en/2.2/ref/settings/ """

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath( file )))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret! SECRET\_KEY = 'x9at8+ndi2w522k3r&54&8gv6zc^#pv4ol\_t^1sl#8c&fjv0hr'

# SECURITY WARNING: don't run with debug turned on in production! DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'FacialApp'

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'Facial.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [

os.path.join('C:/Python/Facial/Facial/FacialApp', 'templates'),

],

'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [ 'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth', 'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'Facial.wsgi.application' # Database

# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql', 'NAME': 'chatbot',

'HOST': '127.0.0.1',

'PORT': '3306',

'USER': 'root',

'PASSWORD': 'root', 'OPTIONS': {

'autocommit': True,

},

}

}

# Password validation

# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME':

'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC\_URL = '/static/'

**Facial\urls.py**

"""Facial URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see: https://docs.djangoproject.com/en/2.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home') Class-based views
3. Add an import: from other\_app.views import Home
4. Add a URL to urlpatterns: path('', Home.as\_view(), name='home') Including another URLconf
5. Import the include() function: from django.urls import include, path
6. Add a URL to urlpatterns: path('blog/', include('blog.urls')) """

from django.contrib import admin from django.urls import path, include.

urlpatterns = [

path('admin/', admin.site.urls), path('', include('FacialApp.urls')),

]

**Facial\wsgi.py**

"""

WSGI config for Facial project.

It exposes the WSGI callable as a module-level variable named ``application``. For more information on this file, see https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Facial.settings') application = get\_wsgi\_application()

**templates\index.html**

{% load static %}

<html>

<head>

<title>Facial Expression Recognition</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="{% static 'style.css' %}" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="main">

<div class="main\_resize">

<div class="header">

<div class="logo">

<h1><span><center>A Deep Learning Facial Expression Recognition Based Scoring

<br/><center>System for Restaurants</center></span><small></small></h1>

</div>

</div>

<div class="content">

<div class="content\_bg">

<div class="menu\_nav">

<ul>

<li><a href="{% url 'Index' %}">Home</a></li>

<li><a href="{% url 'User' %}">User</a></li>

<li><a href="{% url 'Admin' %}">Administrator</a></li>

</ul>

</div>

<div class="hbg"><img src="{% static 'images/header\_images.jpg' %}" width="915" height="286" alt="" /></div>

<p align="justify"><font size="3" style="font-family: Comic Sans MS" color="black">Recently, the popularity of automated and unmanned restaurants has increased. Due to the absence of staff, there is no direct perception of the customers' impressions in order to find out what their experiences with the restaurant

concepts are like. For this purpose, this paper presents a rating system based on facial expression recognition with pre-trained convolutional neural network (CNN) models. It is composed of a web application, a web server, and a pre-trained AIserver. Both the food and the environment are supposed to be rated. Currently, three expressions (satisfied, neutral and disappointed) are provided by the scoring system.</p>

</body>

</html>

**templates\user.html**

{% load static %}

<html>

<head>

<title>Facial Expression Recognition</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="{% static 'style.css' %}" rel="stylesheet" type="text/css" />

<script language="javascript"> function validate(formObj)

{

if(formObj.t1.value.length==0)

{

alert("Please enter customer name"); formObj.t1.focus();

return false;

}

if(formObj.t2.value.length==0)

{

alert("Please enter rating"); formObj.t2.focus();

return false;

}

if(formObj.t3.value.length==0)

{

alert("Please upload photo"); formObj.t3.focus();

return false;

}

formObj.actionUpdateData.value="update"; return true;

}

</script>

</head>

<body>

<div class="main">

<div class="main\_resize">

<div class="header">

<div class="logo">

<h1><span><center>A Deep Learning Facial Expression Recognition Based Scoring

<br/><center>System for Restaurants</center></span><small></small></h1>

</div>

</div>

<div class="content">

<div class="content\_bg">

<div class="menu\_nav">

<ul>

<li><a href="{% url 'Index' %}">Home</a></li>

<li><a href="{% url 'User' %}">User</a></li>

<li><a href="{% url 'Admin' %}">Administrator</a></li>

</ul>

</div>

<div class="hbg"><img src="{% static 'images/header\_images.jpg' %}" width="915" height="286" alt="" /></div>

<center>

<form name="f1" method="post" action="{% url 'Rating' %}" enctype="multipart/form- data" onsubmit="return validate(this);">

{% csrf\_token %}

<br/>

<h2><b>User Rating Screen</b></h2>

<font size="" color="red"><center>{{ data }}</center></font>

<table align="center" width="40" >

<tr><td><b>Customer&nbsp;Name</b></td><td><input type="text" name="t1" style="font-family: Comic Sans MS" size="20"/></td></tr>

<tr><td><b>Rating</b></td><td><input type="text" name="t2" style="font-family: Comic Sans MS" size="10"/></td></tr>

<tr><td><b>Upload&nbsp;Photo</b></td><td><input type="file" name="t3" style="font-family: Comic Sans MS" size="30"/></td></tr>

<tr><td></td><td><input type="submit" value="Submit"></td>

</table>

</div>

</div>

</body>

</html>

**templates\adminscreen.html**

{% load static %}

<html>

<head>

<title>Facial Expression Recognition</title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="{% static 'style.css' %}" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="main">

<div class="main\_resize">

<div class="header">

<div class="logo">

<h1><span><center>A Deep Learning Facial Expression Recognition Based Scoring

<br/><center>System for Restaurants</center></span><small></small></h1>

</div>

</div>

<div class="content">

<div class="content\_bg">

<div class="menu\_nav">

<ul>

<li><a href="{% url 'ViewRating' %}">View Users Rating</a></li>

<li><a href="{% url 'Index' %}">Logout</a></li>

</ul>

</div>

<div class="hbg"><img src="{% static 'images/header\_images.jpg' %}" width="915" height="286" alt="" /></div>

<p align="justify"><font size="3" style="font-family: Comic Sans MS" color="black">

<font size="" color="red"><center>{{ data }}</center></font></p>

</body>

</html>

**manage.py**

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks.""" import os

import sys def main():

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Facial.settings') try:

from django.core.management import execute\_from\_command\_line except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?"

) from exc execute\_from\_command\_line(sys.argv)

if name == ' main ': main()

* + - * 1. **Outline for Various Files**

**app1\views.py**

This is the important file of our project. First the required modules are imported. Then the facial expressions from the .xml haar classifier file are read. We need to pre-process our image by removing any light effects or unwanted effects to enhance the image. Next, we resize the image into blocks of 48\*48 to detect the face in image. Once this process is done, we will have all the faces detected in the image. Then the face with the highest accuracy is processed further for classification. In the next step, the trained CNN model detects the emotion label of the face detected and provides us the result.

**urls.py**

Every page on the Internet needs its own URL. This way your application knows what it should show to a user who opens that URL. In Django, we use something called URLconf (URL configuration). URLconf is a set of patterns that Django will try to match the requested URL to find the correct view. This happens in urls.py file.

**settings.py**

settings.py is a core file in Django projects. It holds all the configuration values that your web app needs to work; database settings, logging configuration, where to find static files, API keys if you work with external APIs, and a bunch of other stuff.

**wsgi.py**

Django's primary deployment platform is WSGI, the Python standard for web servers and applications. Django's startproject management command sets up a minimal default WSGI configuration for you, which you can tweak as needed for your project, and direct any WSGI-compliant application server to use.

**index.html , user.html and adminscreen.html**

An HTML file contains Hypertext Markup Language (HTML), which is used to format the structure of a webpage. It is stored in a standard text format and contains tags that define the page layout and content of the webpage, including the text, tables, images, and hyperlinks displayed on the webpage.

**manage.py**

A command-line utility that lets you interact with this **Django** project in various ways. The **manage.py** script is used to create applications, work with databases, and start the development web server.

* + - 1. **PROJECT TESTING**

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.

* + - * 1. **Various Testcases**

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of a test oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

* + - * 1. **Block Box Testing**

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing. The main focus of black box testing is on the validation of your functional requirements. Black box testing gives abstraction from code and focuses on testing effort on the software system behavior. Black box testing facilitates testing communication amongst modules

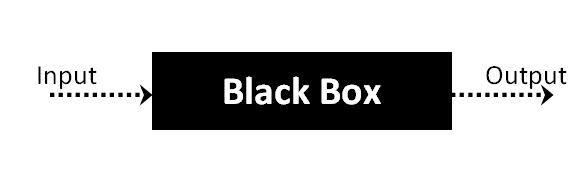


Fig 7.2.1 Black Box Structure

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially, the requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

**Types of Black Box Testing**

There are many types of Black Box Testing but the following are the prominent ones -

**Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.

**Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.

**Regression testing** - Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

**Tools used for Black Box Testing:**

Tools used for Black box testing largely depends on the type of black box testing you are doing.

For Functional/ Regression Tests you can use - QTP, Selenium.

For Non-Functional Tests, you can use - LoadRunner, Jmeter.

**Black Box Testing Techniques**

Following are the prominent Test Strategy amongst the many used in Black box Testing

**Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.

**Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.

**Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

* + - * 1. **White Box Testing**

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type

perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "Whitebox" was used because of the see-through box concept. The clear box or Whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end- user experience can be tested. White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of Whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

**Steps in White Box Testing:**

we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

**Step 1) Understand the Source Code**

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive

users who might inject malicious code into the application either knowingly or unknowingly.

**Step 2) Create Test Cases and Execute**

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

**White Box Testing Techniques**

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product. There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques a box tester can use:

**Statement Coverage**:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

**Branch Coverage -** This technique checks every possible path (if-else and other conditional loops) of a software application.

Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.

Following are important Whitebox Testing Techniques:

Statement Coverage

Decision Coverage

Branch Coverage

Condition Coverage

Multiple Condition Coverage

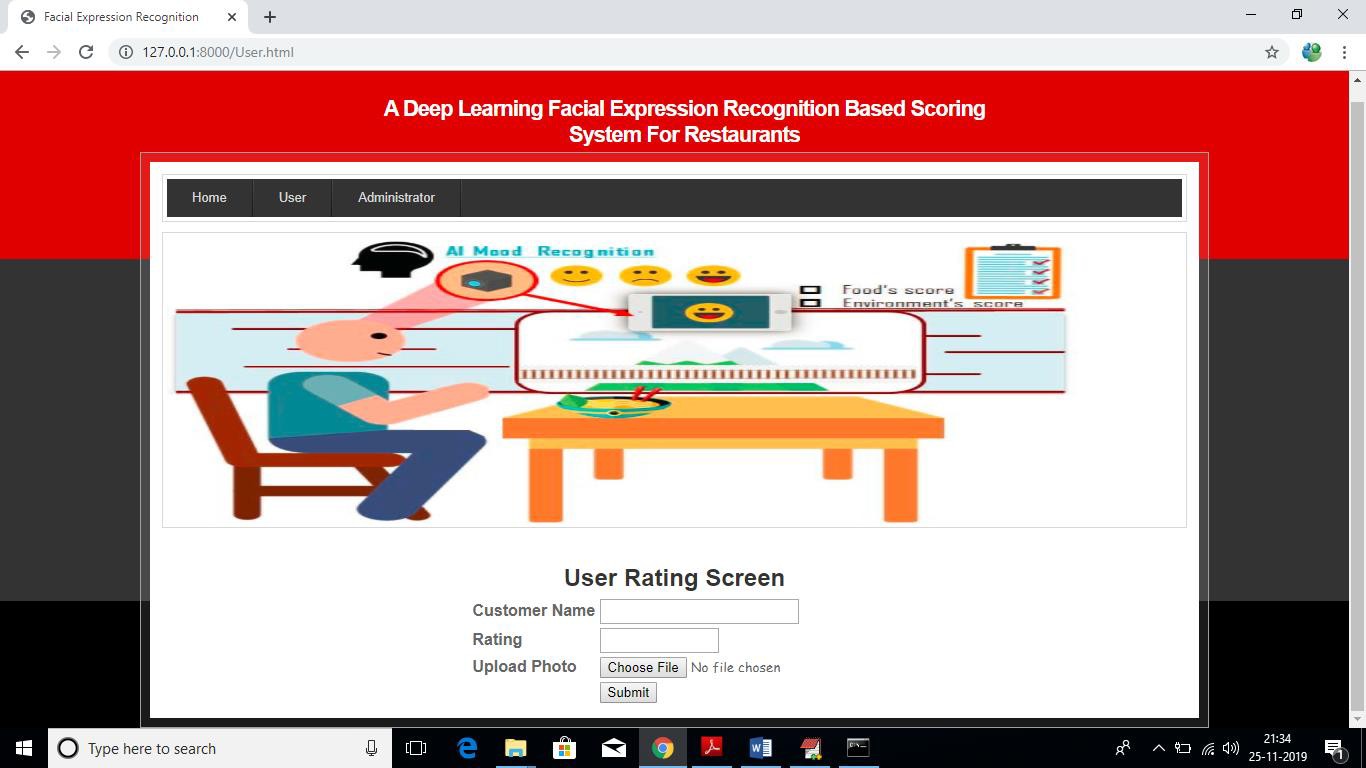
Finite State Machine Coverage

Path Coverage

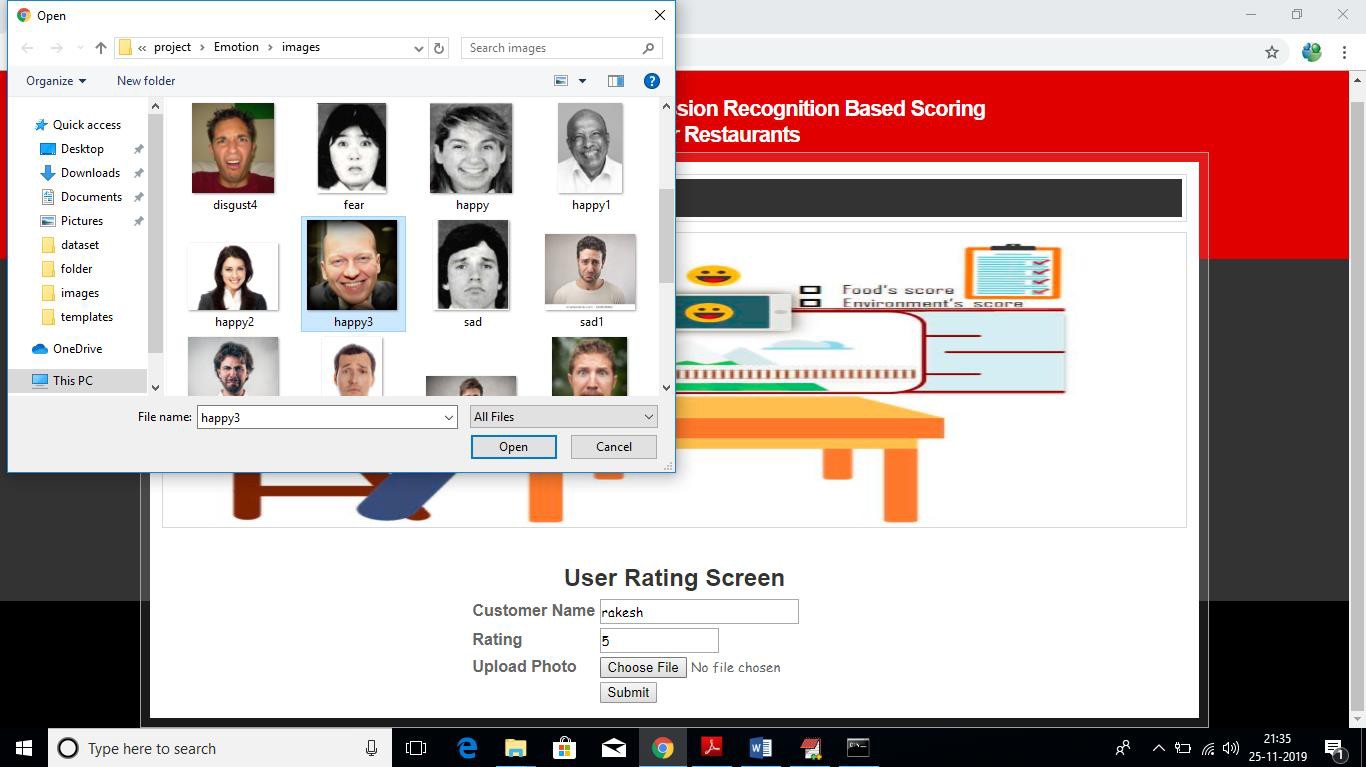
Control flow testing

Data flow testing

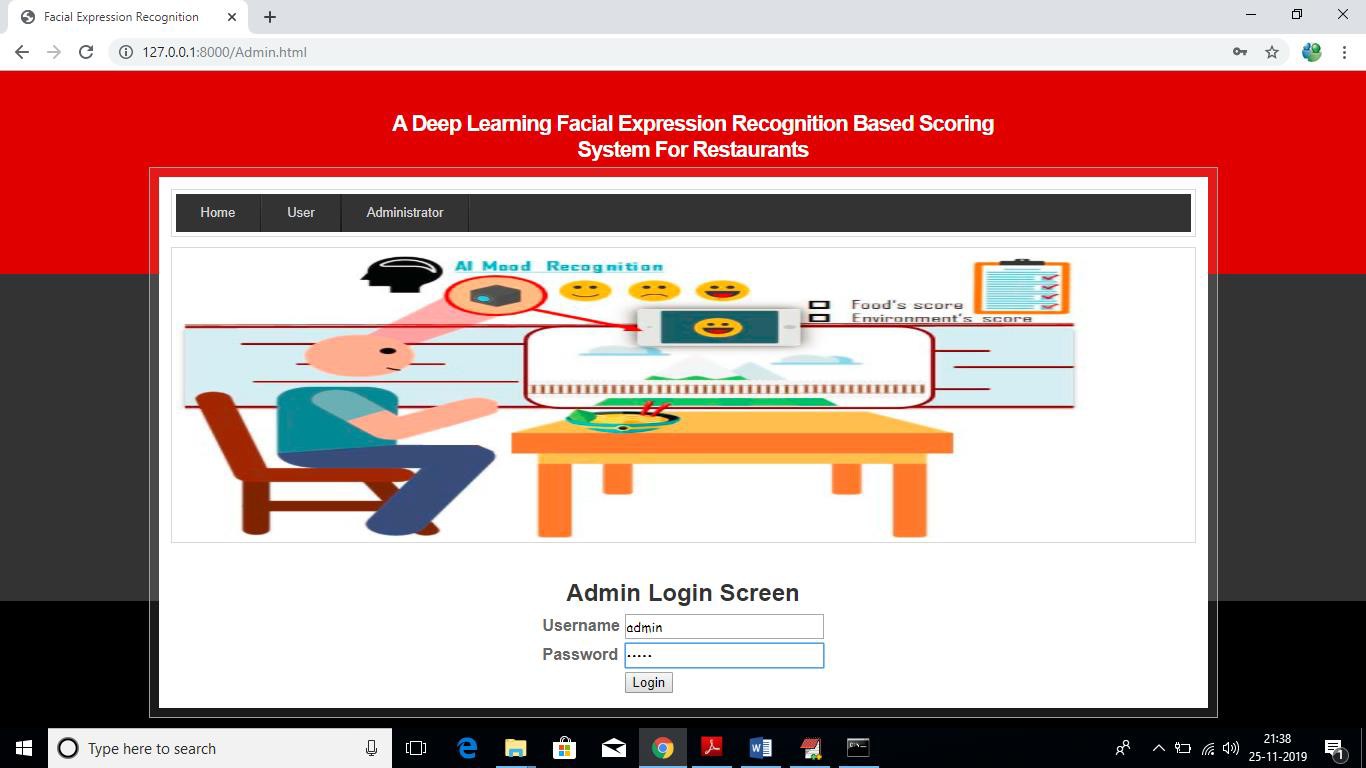
* 1. **Input Screens:-**
     + 1. **OUTPUT SCREENS**



**Screenshot 8.1.1 User uploading rating**



**Screenshot 8.1.2 User uploading image**

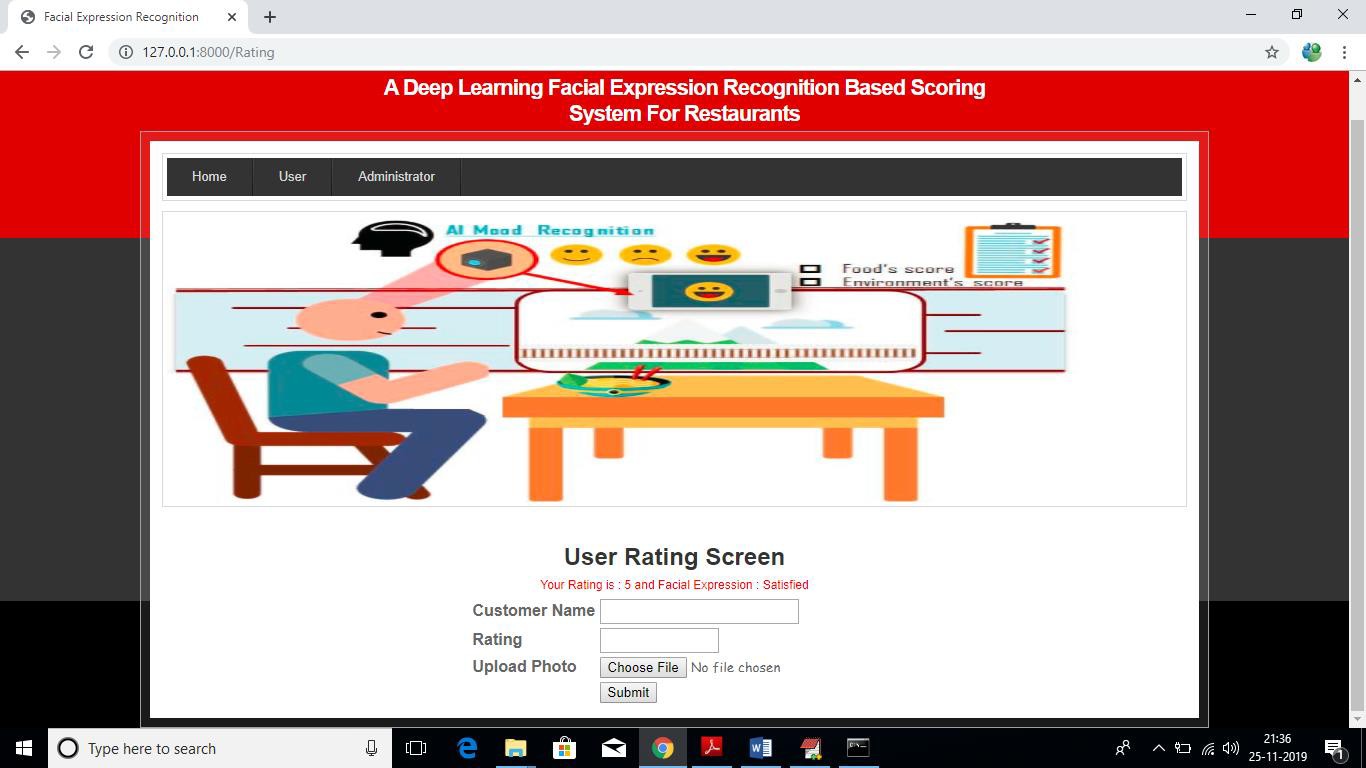


**Screenshot 8.1.3 Admin login**

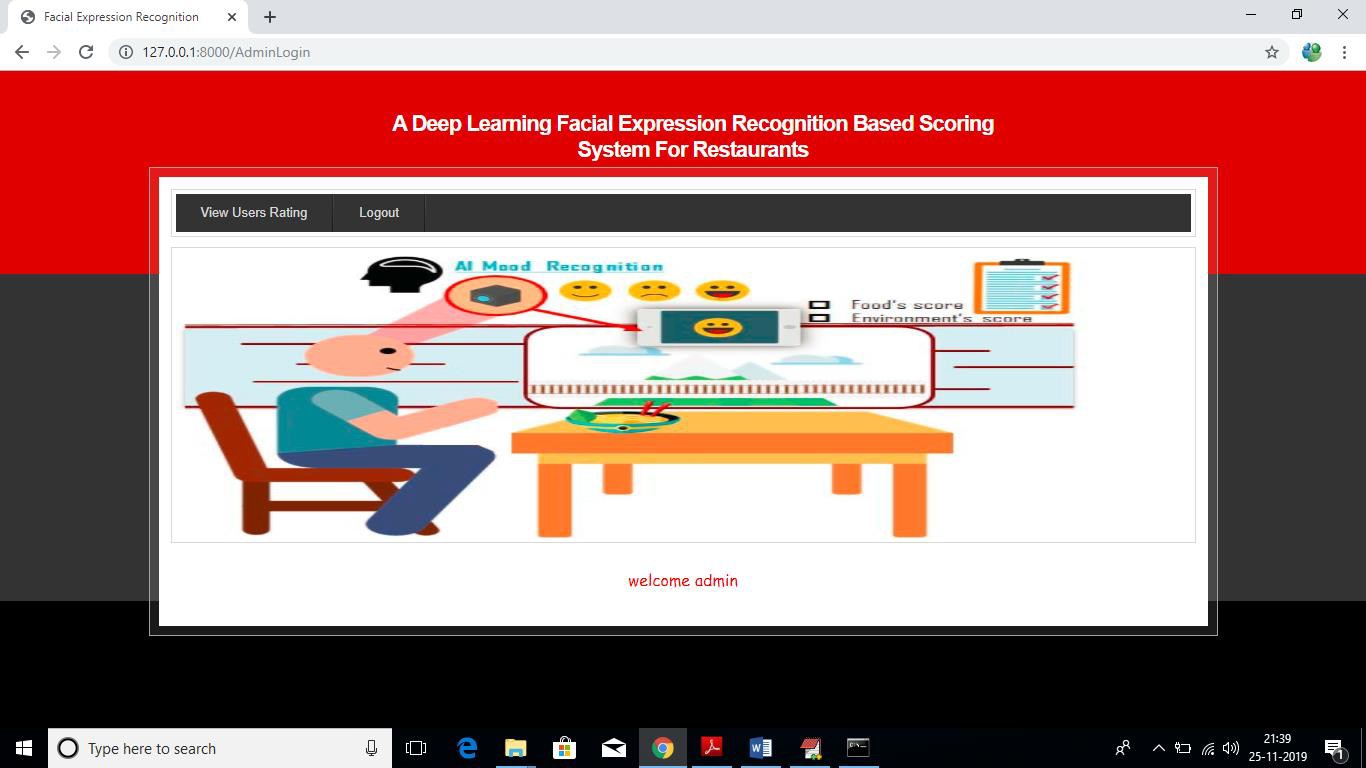
* 1. **Output Screens:-**



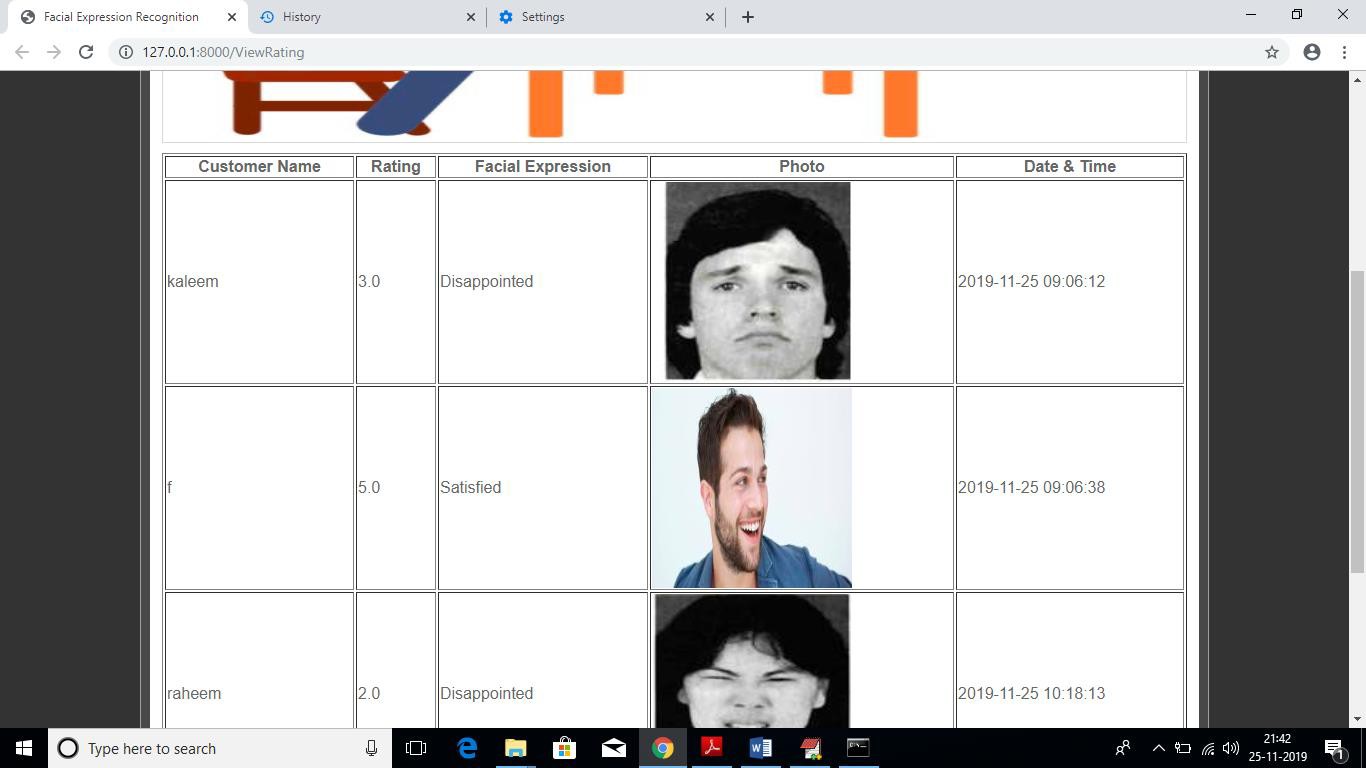
**Screenshot 8.2.1 Home Page**



**Screenshot 8.2.2 User rating output**

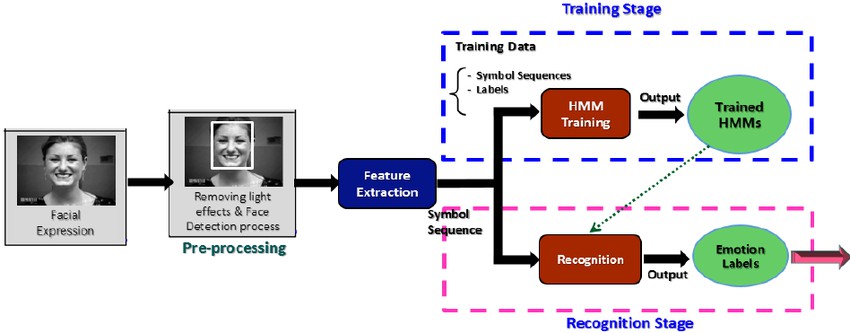


**Screenshot 8.2.3 Admin welcome page**



**Screenshot 8.2.4 Admin accessing reviews with emotion label results**

* + - 1. **EXPERIMENTAL RESULTS**
         1. **Analyzing the result with the help of input image:**



**Fig 9.1.1 Processing of input image**

From the above flow, we can analyze the input image and predict the emotion label just by viewing the image. The results from the experimental results can thus be said to match with the actual generated results.

* + - 1. **CONCLUSION AND FUTURE ENHANCEMENT**
         1. **Conclusion**

We proposed a deep learning facial expression based scoring system for restaurants for the purpose of classification of customer reviews. We filtered input images, performed face detection and finally generated the emotion label using the CNN Model. We achieved an accuracy of 85% with Haar based classifiers.

* + - * 1. **Future Enhancement**

1. A further development could lead to a system where the customer can rate touch-less in the restaurant. For this, it must be ensured that the accuracy of the facial expression recognition is high enough.
2. It is also an idea to extend the image-based rating system with a speech recognition feature. The customer could express his opinion and impressions verbally or make suggestions for improvement like it is already done with Google ratings.
3. It is planned to extend the system with a web application that will enable the restaurant management to get a quick graphical overview and easy insights into the statistics.
   * + 1. **REFERENCES**
4. Hussain Saeed, Ali Shouman,Malis Elfar,Mostafa Shabka,Shikharesh Majumdar, and Chung Horng-Lung, “Near-field communication sensors and cloud based smart restaurant management system,” in Proceedings of the 2016 IEEattern Recognition(CVPRE 3rd World Forum on Internet of Things(WTIOT),pp.686-691,2016
5. Florian Schroff,Dmitry Kalenichenko,and James Philbin,”FaceNet:a unified embedding for face recognition and clustering” n Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition(CVPR),pp.815-823,2015
6. Andrew G. Howard,Menglong Zhu, Bo Chen,Dmitry Kalenichenko, Weijun Wang, Tobias sWeyand,Macro Andreeto,and Hartwig Adam,”MobileNets:efficient convolutional neural networks for mobile vision applications,”
7. Janne Tommala,Pedram Ghazi,Bishwo Adhikara,and Hekki Huttunen,”Real time system for facial analysis,” in Proceedings of the 7 th European Workshop on Visual Information Processing.
8. Muscles of The Face and Their Functions Facial Expression Recognition (Face recognition Techniques) Part 1 M Anatomy "Muscles of the Face and Their Functions facial Expression Recognition (Face Recognition Techniques) Part 1."
9. FERA 2015-Second Facial Expression Recognition and Analysis challenge - IEEE Conference publication
10. Dr. Chetana Tukkoji et.al “ITM-CLD: Intelligent traffic management to handling cloudlets of the large data”, published in CSOC 2018-Cybernetics and Algorithms in Intelligent Systems, book series AISC- volume 765, 17 May 2018.