A Project Report on

# BMI ESTIMATION USING FACIAL FEATURES

*is submitted in partial fulfillment of the requirement for the award of the Degree of*

***BACHELOR OF TECHNOLOGY***

*to*

**G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY**
**(Autonomous)**
Approved by AICTE | NAAC Accreditation with 'A' Grade
Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA

**by**


| | |
|---|---|
| **BOYA SOMANAIDU** | **(19AT1A05E3)** |
| **KURUVA SREENU** | **(19AT1A05F2)** |
| **GADIVEMULA SRIVATHSA** | **(19AT1A05F8)** |
| **KAKANURU UMESH CHANDRA REDDY** | **(19AT1A05G9)** |


**Under the Guidance of**

**Mr. D. JAYANARAYANA REDDY** M.Tech.,(Ph.D)

**Assistant Professor**

GPCET
Pioneering Innovative Education


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY**
**(Autonomous)**
Approved by AICTE | NAAC Accreditation with 'A' Grade
Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA
Nandikotkur Road, Venkayapalli (V), Kurnool - 518452, Andhra Pradesh


**2019-2023**

# G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

**(Autonomous)**

(Approved by AICTE | NAAC Accreditation with 'A' Grade
Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

GPCET
Pioneering Innovative Education

## CERTIFICATE

This is to certify that the project report entitled **"BMI ESTIMATION USING FACIAL FEATURES"** being submitted by **BOYA SOMANAIDU (19AT1A05E3), KURUVA SREENU (19AT1A05F2), GADIVEMULA SRIVATHSA (19AT1A05F8), KAKANURU UMESH CHANDRA REDDY (19AT1A05G9)** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of G. Pullaiah College of Engineering and Technology, Kurnool is a record of bonafide work carried out by them 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 or Diploma.

**Mr. D. JAYANARAYANA REDDY** M.Tech.,(Ph.D)
Project Supervisor

**Mrs. M. SRI LAKSHMI** M.Tech.,(Ph.D)
Head of the Department

Date of Viva-Voce_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

# ACKNOWLEDGEMENT

We thank our project supervisor, **Mr. D. JAYANARAYANA REDDY**M.Tech.,(Ph.D) for his guidance, valuable suggestions and support in the completion of the project.

We would like to express our deep sense of gratitude and our sincere thanks to HOD **Mrs. M. SRI LAKSHMI**M.Tech.,(Ph.D). Department of Computer Science and Engineering, G. Pullaiah College of Engineering and Technology, Kurnool for providing the necessary facilities and encouragement towards the project work.

We owe indebtedness to our principal **Dr. C. SREENIVASA RAO**M.Tech., Ph.D. G. Pullaiah College of Engineering and Technology, Kurnool for providing us the required facilities.

We are extremely grateful to Chairman **Mr. G. V. M. MOHAN KUMAR** of G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh for their good blessings.

We gratefully acknowledge and express our thanks to teaching and non teaching staff of CSE Department.

## Project Associates

**BOYA SOMANAIDU (19AT1A05E3)**
**KURUVA SREENU (19AT1A05F2)**
**GADIVEMULA SRIVATHSA (19AT1A05F8)**
**KAKANURU UMESH CHANDRA REDDY (19AT1A05G9)**

# ABSTRACT

Body Mass Index (BMI) is a measure of how healthy a person is with respect to their body weight. BMI has shown a correlation with various factors like physical health, mental health, popularity. BMI calculation often requires accurate height and weight, which would take manual work to measure. Largescale automation of BMI calculation can be utilized for analyzing various aspects of society and can be used by governments and companies to make effective decisions. Previous works have used only geometric facial features discarding other information, or a data-driven deep learning-based approach in which the amount of data becomes a bottleneck. In this project we are using python CNN (convolution neural networks) algorithm to predict BMI by analyzing facial features. CNN will take image as input and then extract facial features from image and based on facial features BMI will be predicted.

**Keywords:** geometric facial features, data-driven approach, deep learning, CNN.

# LIST OF FIGURES AND TABLES

# CHAPTER 1

# INTRODUCTION

# CHAPTER 1
# INTRODUCTION

## 1.1 Introduction:

The BMI (Body Mass Index) of any person is a crucial indicator of health. It checks if the person is underweight, normal, overweight, or obese. In the current scenario, health is one of the most neglected factors. Technology which has more benefits also has some drawbacks. It has made humans lazy and thus reduced their physical activity leading to a sedentary lifestyle and a rise in BMI which adversely affects their health and increases the risk of chronic diseases. The more the BMI, the more is the chance of developing cardiovascular and other harmful diseases. On the other side of the coin, some people have problems like malnutrition and deficiencies. So, BMI can help a person to keep a track record of their health. According to [1], on average, one out of every three adults is obese, which is about 36% of the population, and by the year 2030, an estimated 20% of the global population would be obese. Human faces carry a significant amount of information about a person. Recent studies have shown a strong correlation between the human face and the BMI of the person. The people with skinny faces have chances of less BMI and vice versa. Generally, obese people tend to have the middle and lower part of the face wider. It is difficult for the person to calculate BMI if they do not have a measuring tape and weighing machine. Recently there have been many advancements in deep learning where models can extract meaningful features from the images. By utilizing these methods, we can predict the BMI from human faces. So, In this paper, we have proposed a technique to predict BMI from human faces. This system could help health insurance companies to maintain the health records of their customers. Also, the government could track the health records of a particular region and devise policies accordingly.

**Keywords**: BMI, Cardiovascular diseases.

# CHAPTER 2

# LITERATURE REVIEW

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 Literature survey:

### 2.1.1 Title: Face-to-BMI: Using Computer Vision to Infer Body Mass Index on social media

https://www.researchgate.net/publication/314433619_Face-to-BMI_Using_Computer_Vision_to_Infer_Body_Mass_Index_on_Social_Media

A person's weight status can have profound implications on their life, ranging from mental health, to longevity, to financial income. At the societal level, "fat shaming" and other forms of "sizeism" are a growing concern, while increasing obesity rates are linked to ever raising healthcare costs. For these reasons, researchers from a variety of backgrounds are interested in studying obesity from all angles. To obtain data, traditionally, a person would have to accurately self-report their body-mass index (BMI) or would have to see a doctor to have it measured. In this paper, we show how computer vision can be used to infer a person's BMI from social media images. We hope that our tool, which we release, helps to advance the study of social aspects related to body weight.

### 2.1.2 Title: Facial Image Analysis for Body Mass Index, Makeup, and Identity

https://researchrepository.wvu.edu/cgi/viewcontent.cgi?article=7979&context=etd

**ABSTRACT:** The principal aim of facial image analysis in computer vision is to extract valuable information (e.g., age, gender, ethnicity, and identity) by interpreting perceived electronic signals from face images. In this dissertation, we develop facial image analysis systems for body mass index (BMI) prediction, makeup detection, as well as facial identity with makeup changes and BMI variations. BMI is a commonly used measure of body fatness. In the first part of this thesis, we study BMI related topics. At first, we develop a computational method to predict BMI information from face images automatically. We formulate the BMI prediction from facial features as a machine vision problem. Three regression methods, including least square estimation, Gaussian processes for regression, and support vector regression are employed to predict the BMI value. Our preliminary results show that it is feasible to develop a computational system for BMI prediction from face images. Secondly, we address the influence of BMI changes on face identity. Both synthesized and real face images are assembled as the databases to facilitate our study. Empirically, we found that large BMI alterations

can significantly reduce the matching accuracy of the face recognition system. Then we study if the influence of BMI changes can be reduced to improve the face recognition performance. The partial least squares (PLS) method is applied for this purpose. Experimental results show the feasibility to develop algorithms to address the influence of facial adiposity variations on face recognition, caused by BMI changes. Makeup can affect facial appearance obviously. In the second part of this thesis, we deal with makeup influence on face identity. It is principal to perform makeup detection at first to address makeup influence. Four categories of features are proposed to characterize facial makeup cues in our study, including skin color tone, skin smoothness, texture, and highlight. A patch selection scheme and discriminative mapping are presented to enhance the performance of makeup detection. Secondly, we study dual attributes from makeup and non-makeup faces separately to reflect facial appearance changes caused by makeup in a semantic level. Cross-makeup attribute classification and accuracy change analysis is operated to divide dual attributes into four categories according to different makeup effects. To develop a face recognition system that is robust to facial makeup, PLS method is proposed on features extracted from local patches. We also propose a dual-attributes based method for face verification. Shared dual attributes can be used to measure facial similarity, rather than a direct matching with low- level features. Experimental results demonstrate the feasibility to eliminate the influence of makeup on face recognition.

**2.1.3 Title: BMI Prediction From Face Images.**

https://www.researchgate.net/publication/335364971_BMI_Prediction_From_Face_Images

**ABSTRACT:** Body-mass index (BMI) is the amount of mass per area of a person, and it is an important indicator of the weight status. From health industry till the social media applications, there are many areas where BMI data is used. Various machine learning techniques are developed for BMI prediction only from a face image without any information about the weight and height of a person.Making this kind of predictions is a regression problem. In this study, a deep network based BMI predictor tool is designed and its performance is compared with the existing methods from previous studies. Additionally, a new data set for validation purposes is introduced.

**2.1.4 Title: A Framework for Healthcare Everywhere: BMI Prediction Using Kinect and Data Mining Techniques on Mobiles**

https://www.researchgate.net/publication/308817784_A_Framework_for_Healthcare_Everywhere_BMI_Prediction_Using_Kinect_and_Data_Mining_Techniques_on_Mobiles

## BMI estimation using facial features

**ABSTRACT:** Recently, health-care has become a popular issue. Having a good physique is also commonly regarded as important for being healthy. For evaluating our body status, the Body Mass Index (BMI) is a widely used indicator. However, calculating BMI is inconvenient and requires the physical measuring of people's weights and heights. In this paper, we are interested in building a mobile-based BMI prediction system using Kinect and data mining techniques so that everybody can easily monitor their BMI everywhere by taking a snapshot of their face. The rationale behind this is the intuition that there is a correlation between the shape of one's face and one's BMI values, which people often act on when noticing a friend has either gained or lost weight. Through the evaluations of 50 volunteers, we show that the rules for training BMI prediction match with people's common intuitions.

**2.1.5 Title: Al-based BMI Inference from Facial Images: An Application to Weight Monitoring**

https://arxiv.org/pdf/2010.07442.pdf

**ABSTRACT:** Self-diagnostic image-based methods for healthy weight monitoring is gaining increased interest following the alarming trend of obesity. Only a handful of academic studies exist that investigate AI-based methods for Body Mass Index (BMI) inference from facial images as a solution to healthy weight monitoring and management. To promote further research and development in this area, we evaluate and compare the performance of five different deep- learning based Convolutional Neural Network (CNN) architectures i.e., VGG19, ResNet50, DenseNet, MobileNet, and lightCNN for BMI inference from facial images. Experimental results on the three publicly available BMI annotated facial image datasets assembled from social media, namely, VisualBMI, VIP-Attributes, and Bollywood datasets, suggest the efficacy of the deep learning methods in BMI inference from face images with minimum Mean Absolute Error (MAE) of 1.04 obtained using ResNet50.

# CHAPTER 3

# EXISTING MODEL AND PROPOSED MODEL

# CHAPTER 3

# EXISTING MODEL AND PROPOSED MODEL

## 3.1 Existing system:

A person's body mass index (BMI) is a gauge of how healthy they are in relation to their weight. Numerous aspects, including physical health, mental health, and popularity, have been linked to BMI. BMI calculations frequently call for precise measurements of height and weight, which entail labor-intensive manual labor. Governments and businesses can employ large-scale automation of BMI calculation to analyze different facets of society and to make smart decisions. Previous approaches have exclusively employed geometric facial traits, disregarding additional information, or have used a data-driven deep learning approach where the volume of data becomes a difficult.

### 3.1.1 Problems of Existing System

1. the volume of data becomes a difficult
2. Recently there have been many advancements in deep learning where models can extract meaningful features from the images but cannot predict BMI

## 3.2 Proposed system:

In this project, we are analyzing facial features to estimate BMI using the Python CNN (convolution neural networks) algorithm. CNN will use a picture as its input, extracting facial traits from it before predicting BMI based on those features.

**Advantages of proposed system:**

1. Far preprocessing for best prediction
2. High prediction rate

## 3.3 Functional Requirements

1.Data Collection

2.Data Preprocessing

3.Training and Testing

4.Modiling

5.Predicting

## 3.4 Non-Functional Requirements

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

Fig 3.1: Block Diagram

## 3.5 System Design

### 3.5.1 System Architecture:

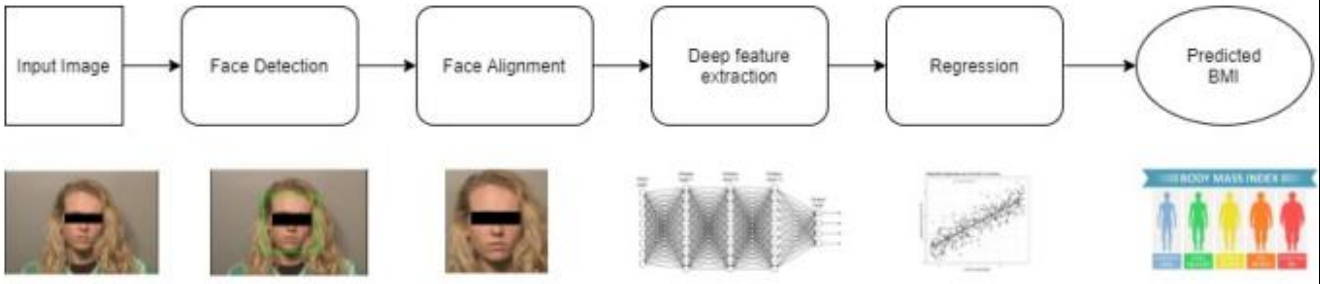



**Fig. 3.5.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

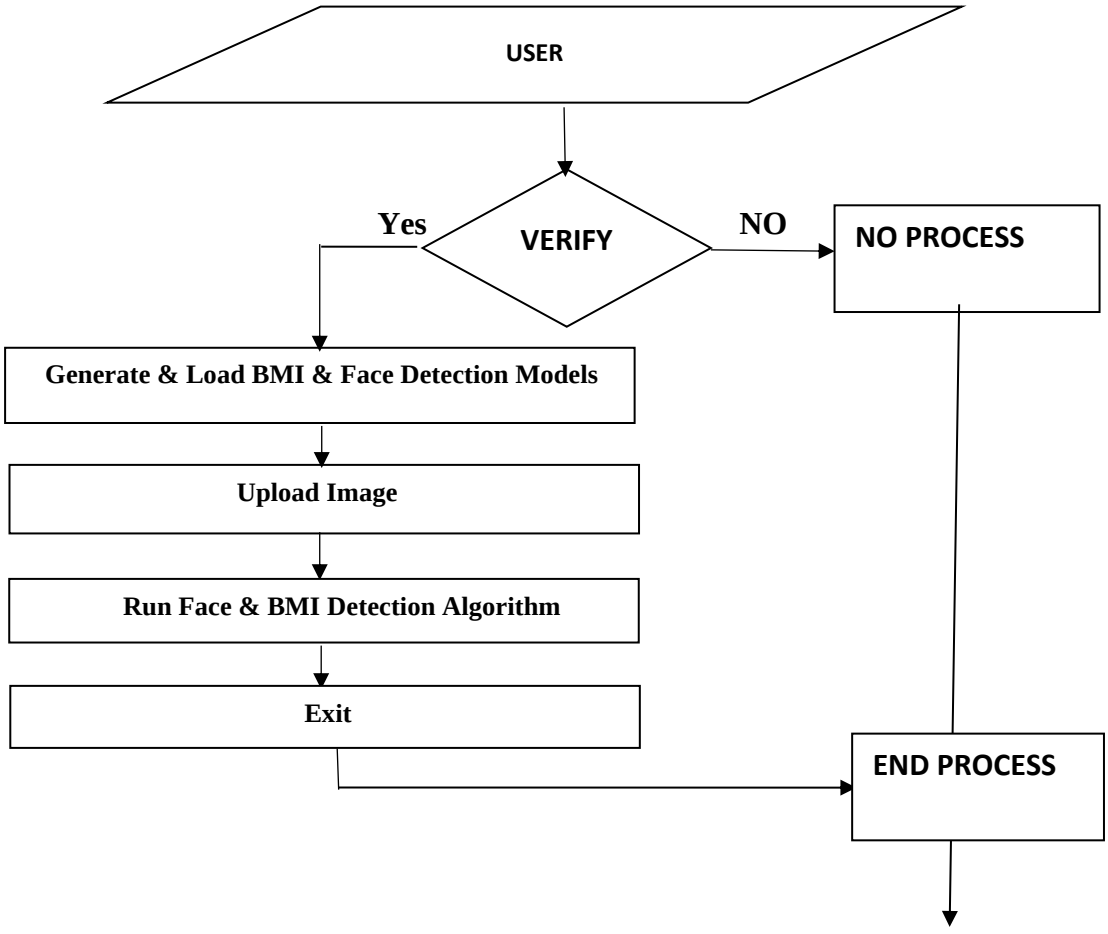



**Fig.3.5.2 Dataflow diagram**

## 3.5.2 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 is comprised of 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 system, 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.

**GOALS:**

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 process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**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.

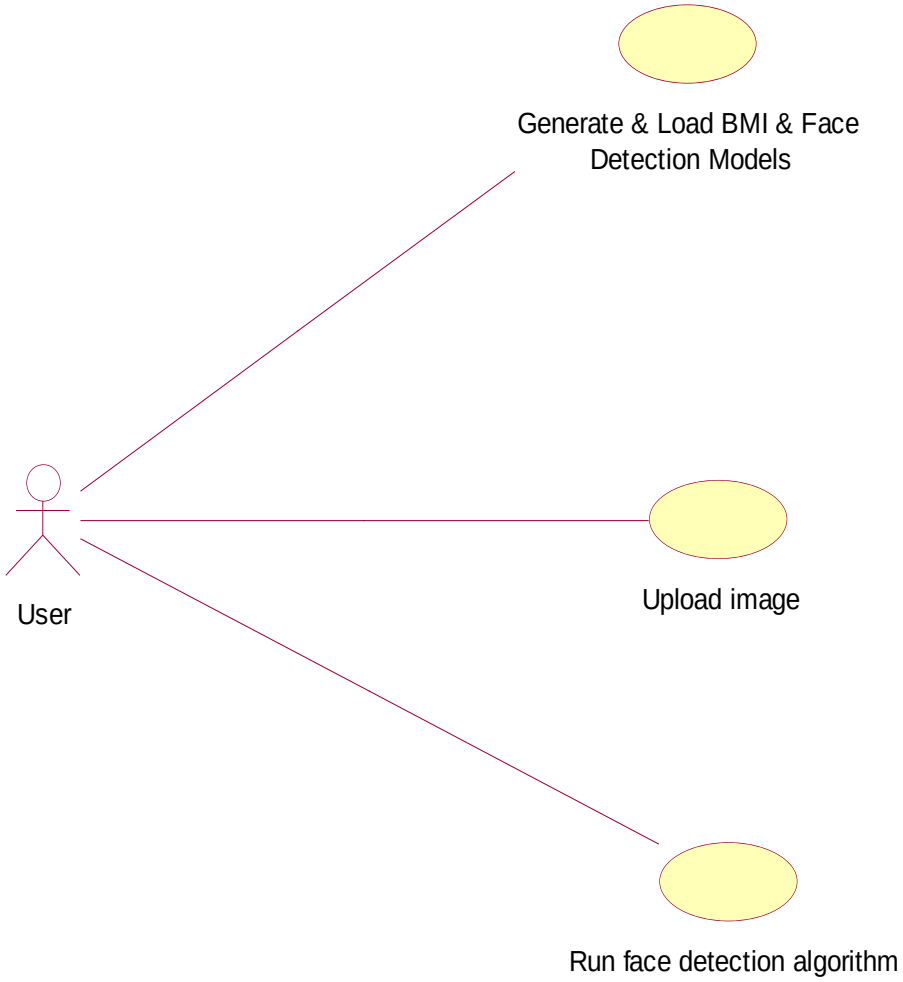



**Fig.3.5.3 Usecase diagram**

**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

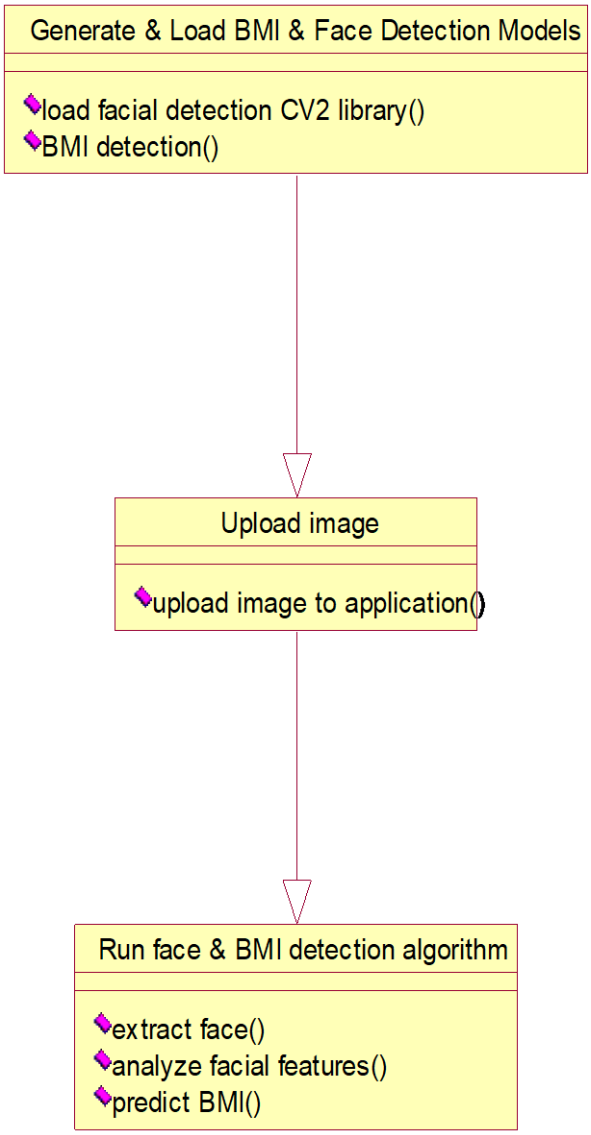



**Fig.3.5.4 Class diagram**

**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

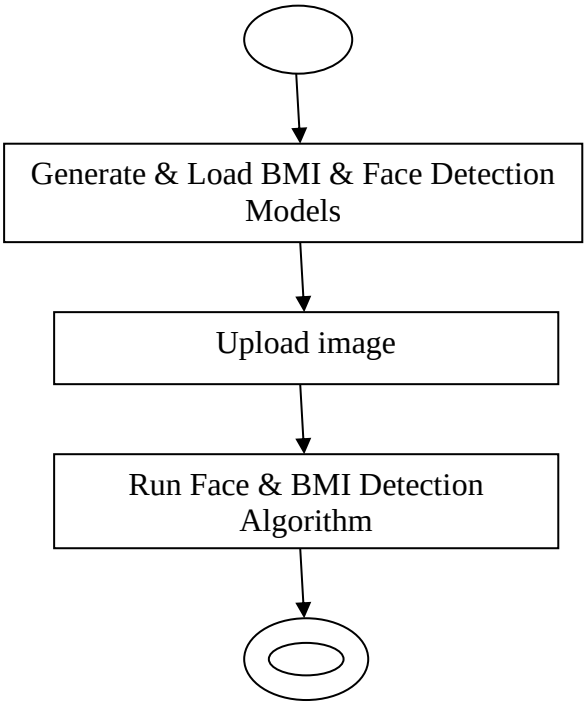



**Fig.3.5.5 Activity diagram**

**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

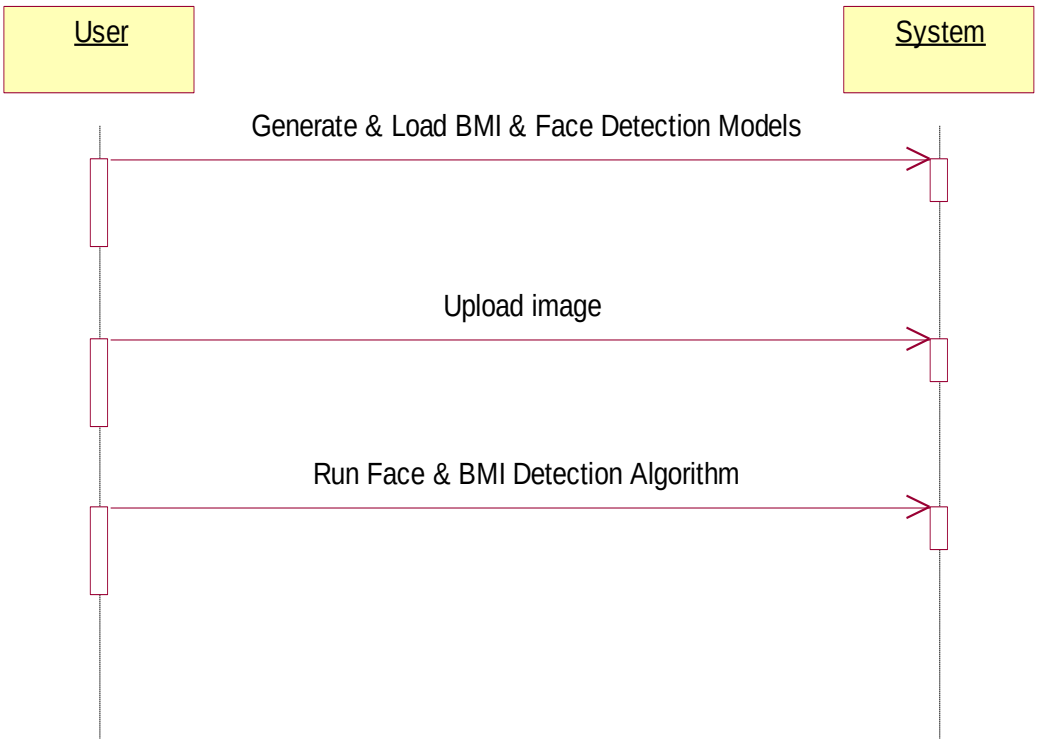



**Fig.3.5.6 Sequence diagram**

**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

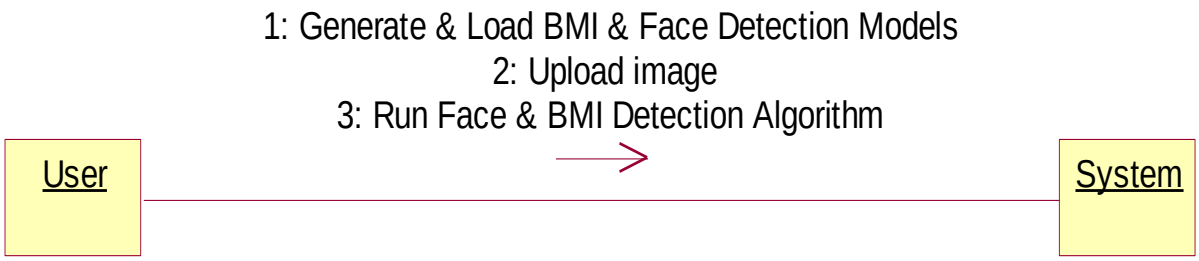



**Fig.3.5.7 Collaboration diagram**

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

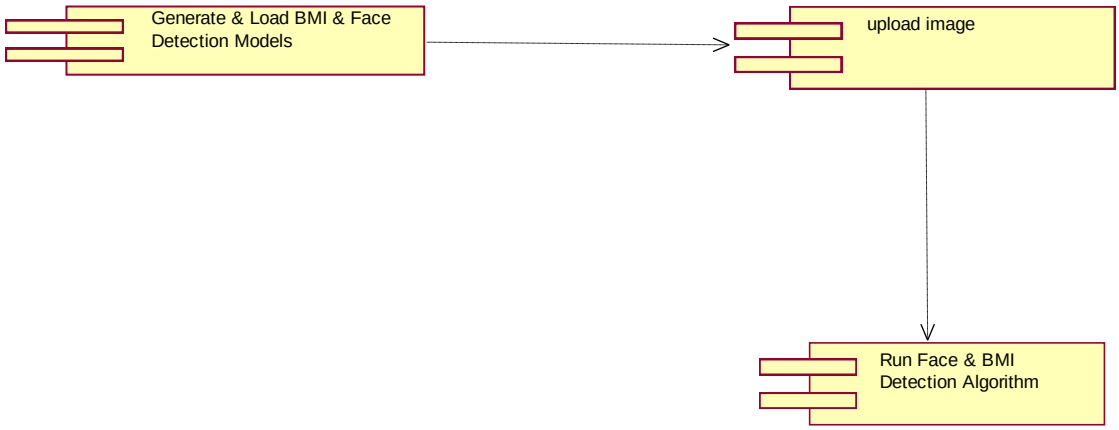



**Fig.3.5.8 Component diagram**

**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

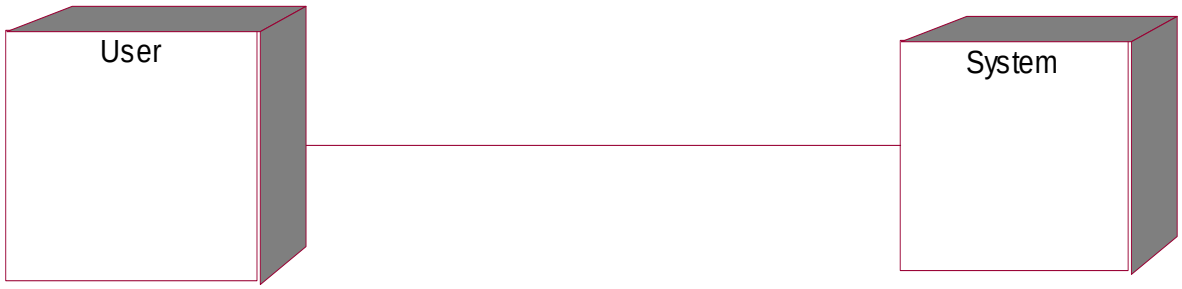



**Fig.3.5.9 Deployment diagram**

## 3.6 Modules:

In this project we are using python CNN (convolution neural networks) algorithm to predict BMI by analyzing facial features. CNN will take image as input and then extract facial features from image

and based on facial features BMI will be predicted. To implement this project we have designed following modules.

1) Generate & Load BMI & Face Detection Models: Using this module we will load facial detection CV2 library and BMI detection CNN model. Facial detection library help us to detect human face from uploaded image and then facial features will input to CNN model to predict BMI
2) Upload Image: using this module we will upload image to application
3) Run Face & BMI Detection Algorithm: This model extract face from given input image and then facial features will be analyze to predict BMI. Based on predicted BMI insurance policy will be quoted to users.

## 3.7 Algorithms:

## CNN:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

## 3.8 TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts

system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

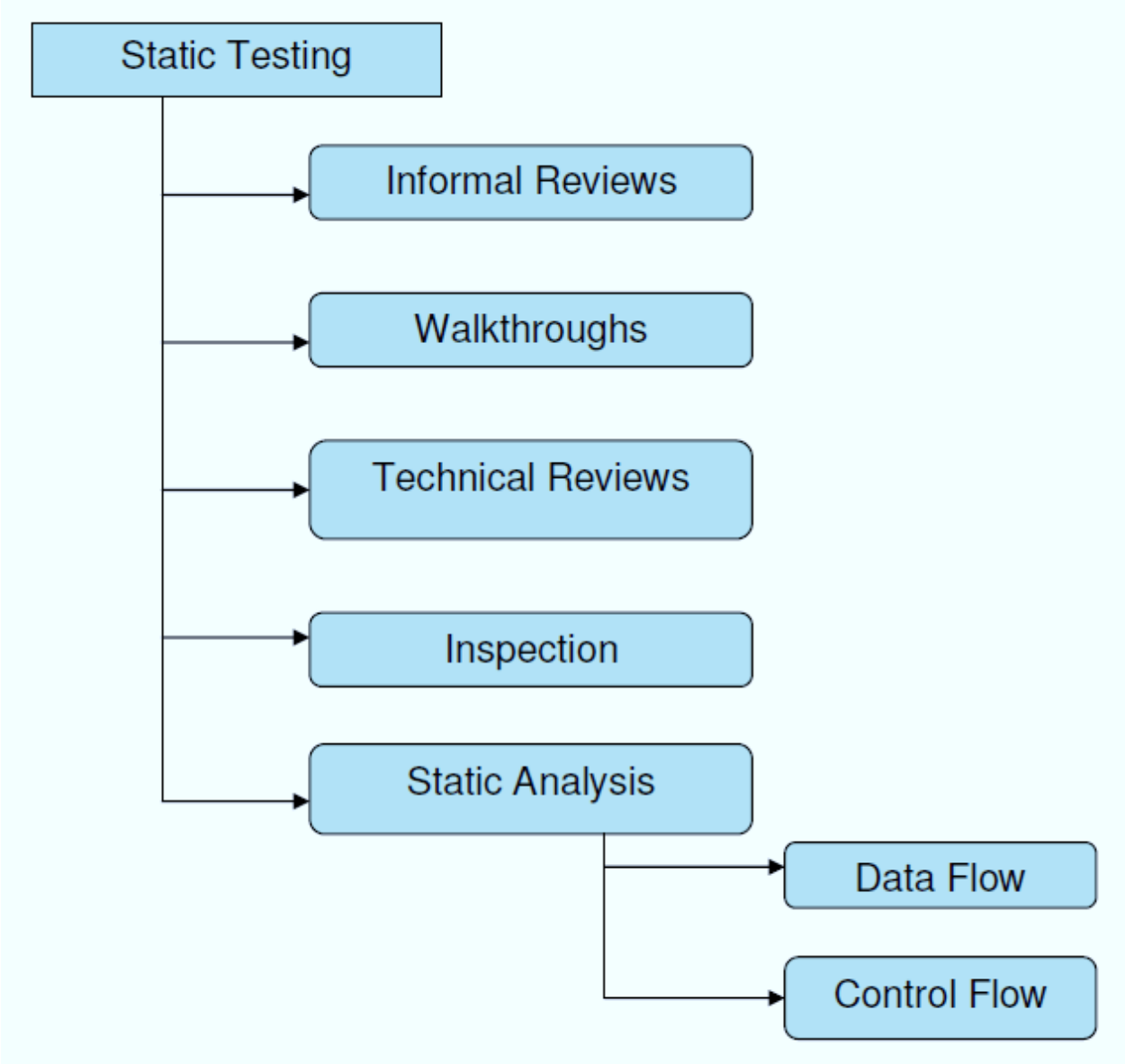
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

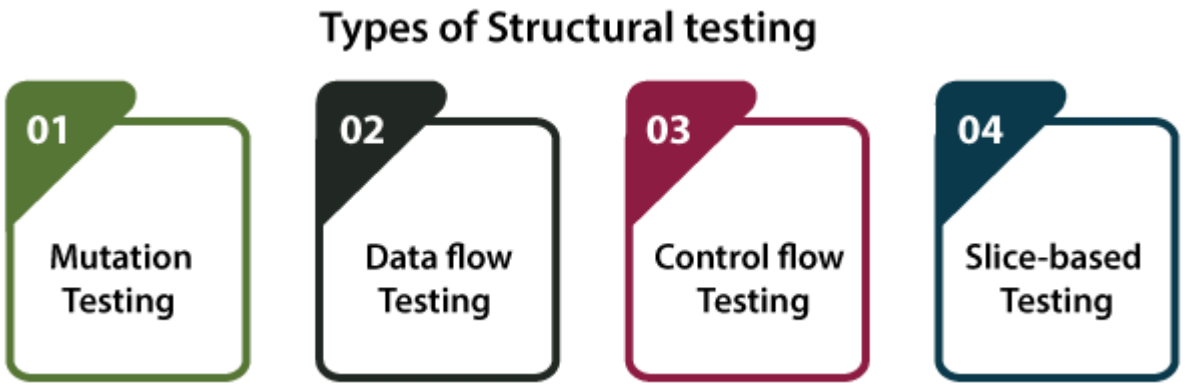
The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.




**Software Testing Strategies**

## Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).




**Behavioral Testing:**

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

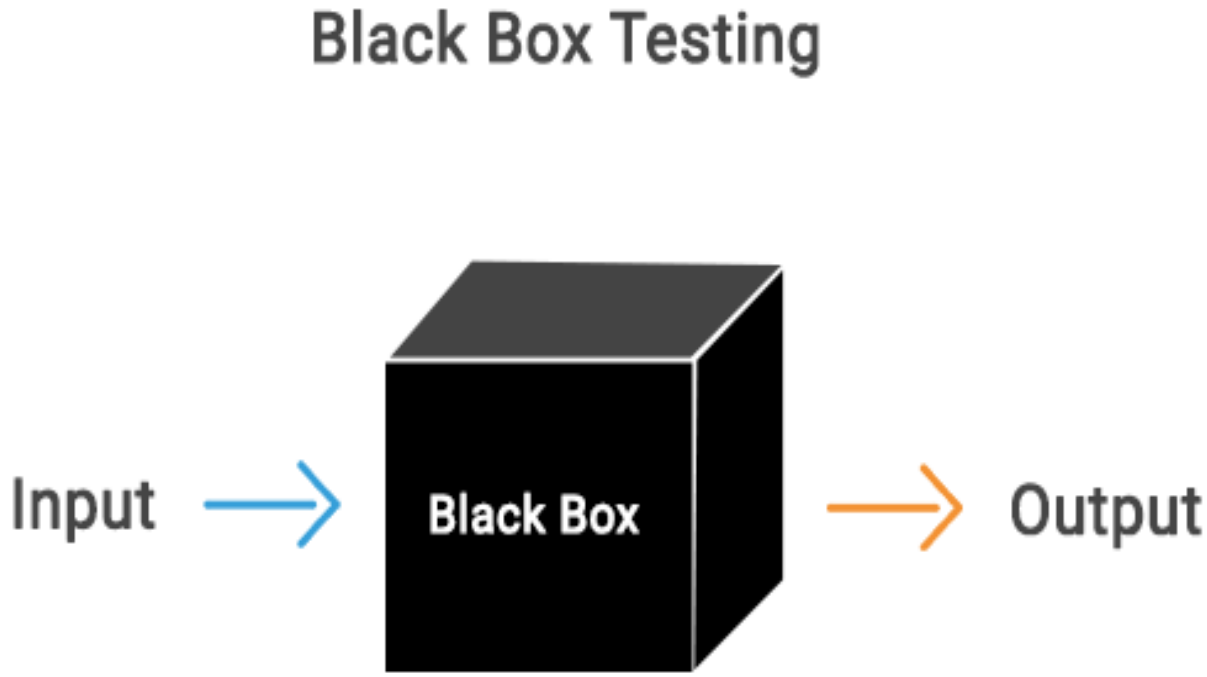


**Fig. 10 Black Box Testing**

**TEST CASES:**

| S.NO | INPUT | If available | If not available |
|---|---|---|---|
| 1 | Generate & Load BMI & Face Detection Models | we will load facial detection CV2 library and BMI detection CNN model | There is no process |
| 2 | Upload image | we will upload image to application | There is no process |
| 3 | Run Face & BMI Detection Algorithm | Will extract face from given input image and then facial features will be analyze to predict BMI | There is no process |

**Table 3.6.1 Test Cases**

# CHAPTER 4

# SYSTEM REQUIREMENTS

# CHAPTER 4

# SYSTEM REQUIREMENTS

## 4.1HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to operating systems running on architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random-access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1)Operating System: Windows Only**
**2)Processor: i5 and above**
**3)Ram: 4gb and above**
**4)Hard Disk: 50 GB**

## 4.2 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

- Operating System: Windows 11
- Visual Studio Community Version 1.77.3
- Nodejs (12.3.1)
- Python IDLE (Python 3.7)

## 4.3 PYTHON LANGUAGE:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

**Features in Python:**

There are many features in Python, some of which are discussed below as follows:

**1. Free and Open Source**

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

**2. Easy to code**

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

**3. Easy to Read**

As you will see, learning Python is quite simple. As was already established, Python's syntax is straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

**4. Object-Oriented Language**

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

**5. GUI Programming Support**

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

**6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

**7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

**8. Easy to Debug**

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

**9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

**10. Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

**11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **bytecode**.

**12. Large Standard Library**

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

**13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

**14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

**15. Allocating Memory Dynamically**

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

## 4.4 LIBRARIES OR PACKAGES: -

**Tensorflow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object

- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to integrate with a wide variety of databases seamlessly and speedily.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Imagine pointing your camera to some object and the camera tells you the name of that object, yes, *Google Lens* in Android smart phones is doing the same thing using Image Processing. This gives computer a *vision* to detect and recognize the things and take actions accordingly. Image processing has lot of applications like Face detection & recognition, thumb impression, augmented reality, OCR, Barcode scan and many more. There are lot of softwares available for image processing, among them **MATLAB** is the most suitable to start with.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

### Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

# CHAPTER 5

# SOURCE CODE AND IMPLEMENTATION

# CHAPTER 5

# SOURCE CODE AND IMPLEMENTATION

## 5.1 SOURCE CODE

```
import cv2
import numpy as np
tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
from train_generator import train_generator, plot_imgs_from_generator
from mae_callback import MAECallback
import config
batches_per_epoch=train_generator.n //train_generator.batch_size
def train_top_layer(model):
    print 'Training top layer...'
    for l in model.layers[:-1]:
        l.trainable = False
    model.compile(
        loss='mean_squared_error',
        optimizer='adam'
    )
    mae_callback = MAECallback()
    early_stopping_callback = EarlyStopping(
        monitor='val_mae',
        mode='min',
        verbose=1,
        patience=1)
    model_checkpoint_callback = ModelCheckpoint(
        'saved_models/top_layer_trained_weights.{epoch:02d}-{val_mae:.2f}.h5',
        monitor='val_mae',
        mode='min',
        verbose=1,
        save_best_only=True
    )
    tensorboard_callback = TensorBoard(
        log_dir=config.TOP_LAYER_LOG_DIR,
        batch_size=train_generator.batch_size
    )
    model.fit_generator(
        generator=train_generator,
        steps_per_epoch=batches_per_epoch,
        epochs=20,
        callbacks=[
            mae_callback,
            early_stopping_callback,
            model_checkpoint_callback,
```

```
            tensorboard_callback
        ]
    )
def train_all_layers(model):
    print 'Training all layers...'
    for l in model.layers:
        l.trainable = True
    mae_callback = MAECallback()
    early_stopping_callback = EarlyStopping(
        monitor='val_mae',
        mode='min',
        verbose=1,
        patience=10)
    model_checkpoint_callback = ModelCheckpoint(
        'saved_models/all_layers_trained_weights.{epoch:02d}-{val_mae:.2f}.h5',
        monitor='val_mae',
        mode='min',
        verbose=1,
        save_best_only=True)
    tensorboard_callback = TensorBoard(
        log_dir=config.ALL_LAYERS_LOG_DIR,
        batch_size=train_generator.batch_size
    )
    model.compile(
        loss='mean_squared_error',
        optimizer='adam'
    )
    model.fit_generator(
        generator=train_generator,
        steps_per_epoch=batches_per_epoch,
        epochs=100,
        callbacks=[
            mae_callback,
            early_stopping_callback,
            model_checkpoint_callback,
            tensorboard_callback
        ]
    )
def test_model(model):
    with open(config.CROPPED_IMGS_INFO_FILE, 'r') as f:
        test_images_info = f.read().splitlines()[-config.VALIDATION_SIZE:]
    test_X = []
    test_y = []
    for info in test_images_info:
        bmi = float(info.split(',')[1])
        test_y.append(bmi)
```

```
        file_name = info.split(',')[4]
        file_path = '%s/%s' % (config.CROPPED_IMGS_DIR, file_name)
        img = cv2.imread(file_path)
        img = cv2.resize(img, (config.RESNET50_DEFAULT_IMG_WIDTH,
    config.RESNET50_DEFAULT_IMG_WIDTH))
        test_X.append(img)
    test_X = np.array(test_X)
    test_y = np.array(test_y)
    mae = get_mae(test_y, model.predict(test_X))
    print '\nMAE:', mae
```

# CHAPTER 6

# EXPERIMENTAL RESULTS

# CHAPTER 6

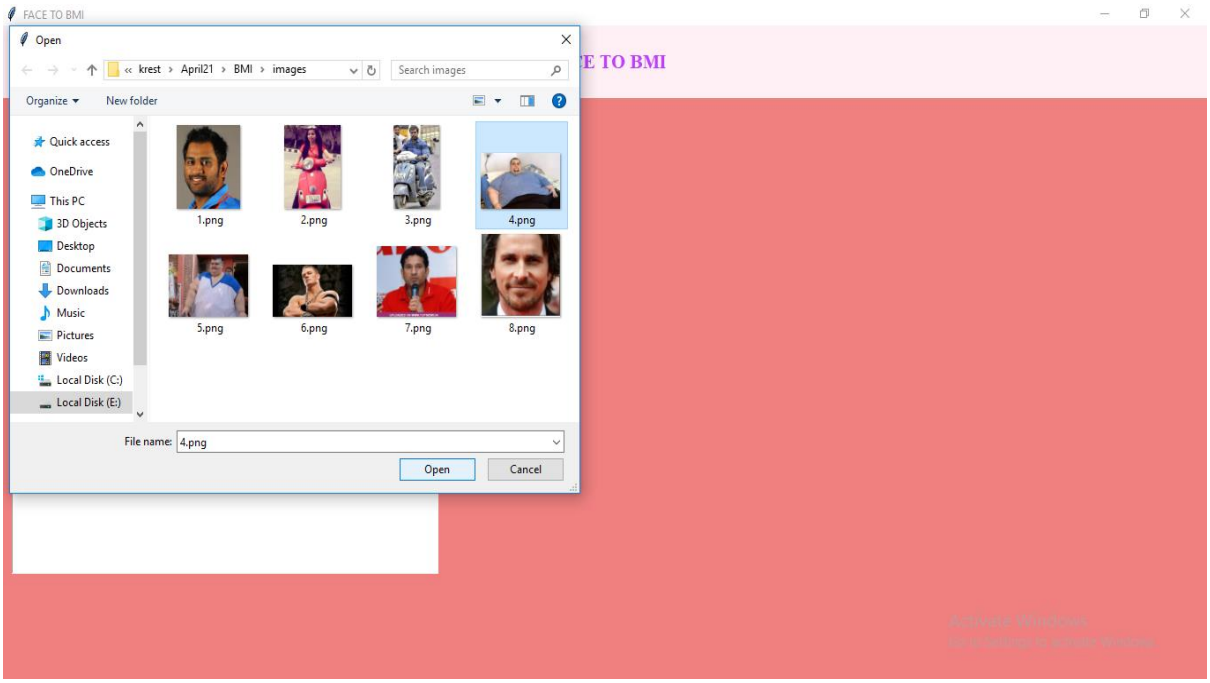
# EXPERIMENTAL RESULTS

## 6.1 RESULTS AND DISCUSSIONS

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

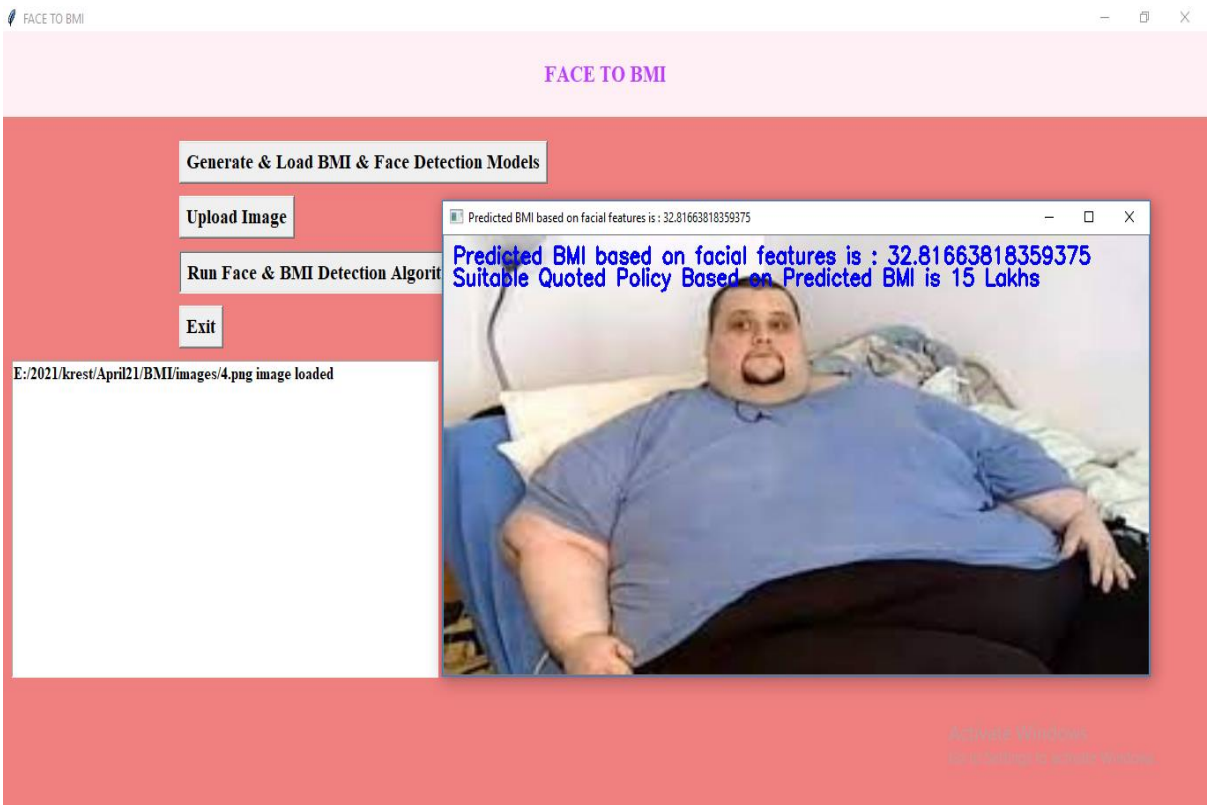

In above screen click on 'Generate & Load BMI & Face Detection Models' button to load face detection library and CNN model to detect BMI.


In above screen both libraries are loaded and now click on 'Upload Image' button to upload image.

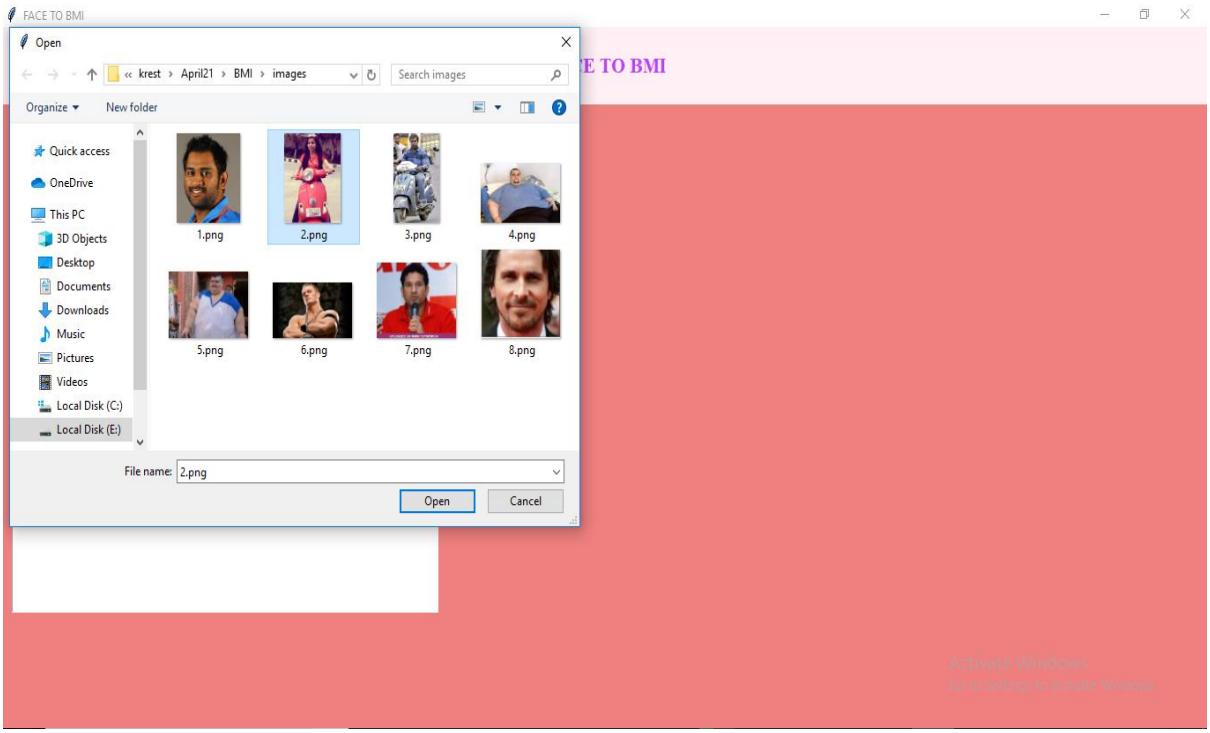


In above screen selecting and uploading ‘4.png’ file and then click on ‘Open’ button to load image and will get below screen.

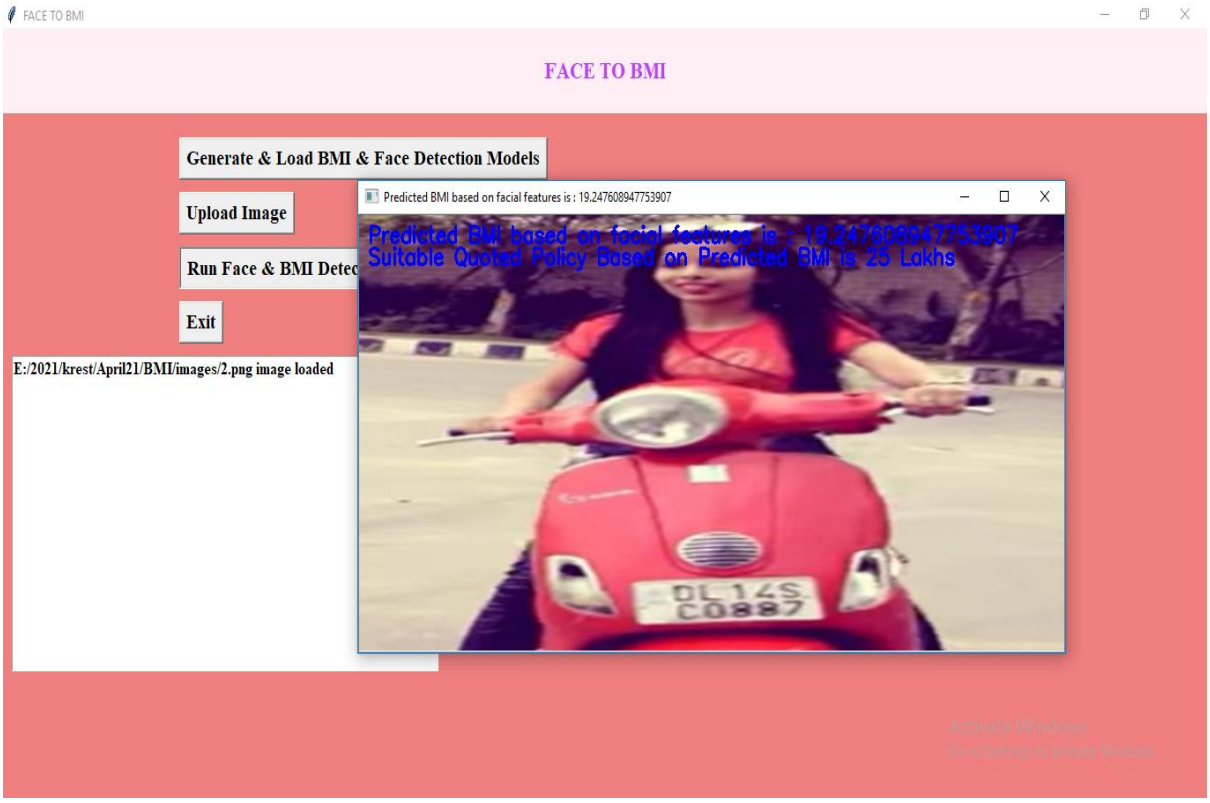

In above screen image is loaded and now click on ‘Run face & BMI Detection Algorithm’ button to get below result



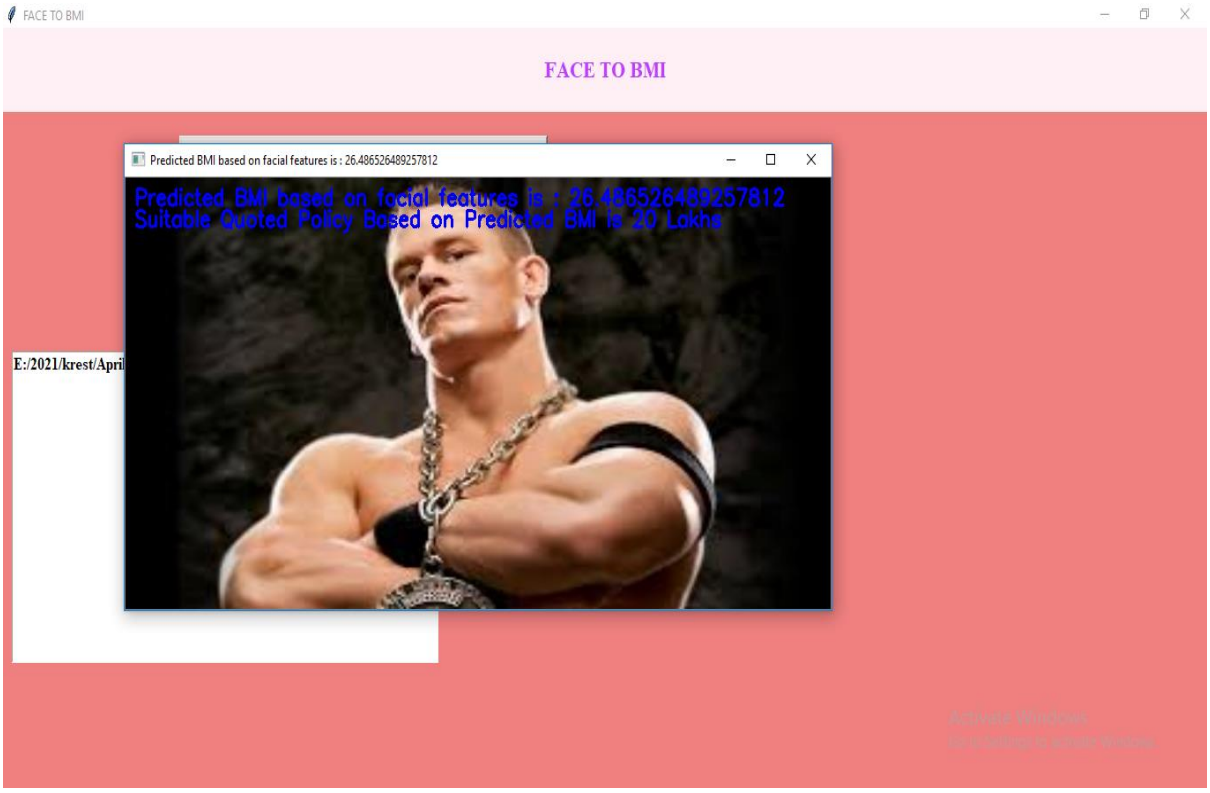

In above screen for given image detected BMI is 32.81 and suggested insurance policy is for 15 lakhs and now try other image



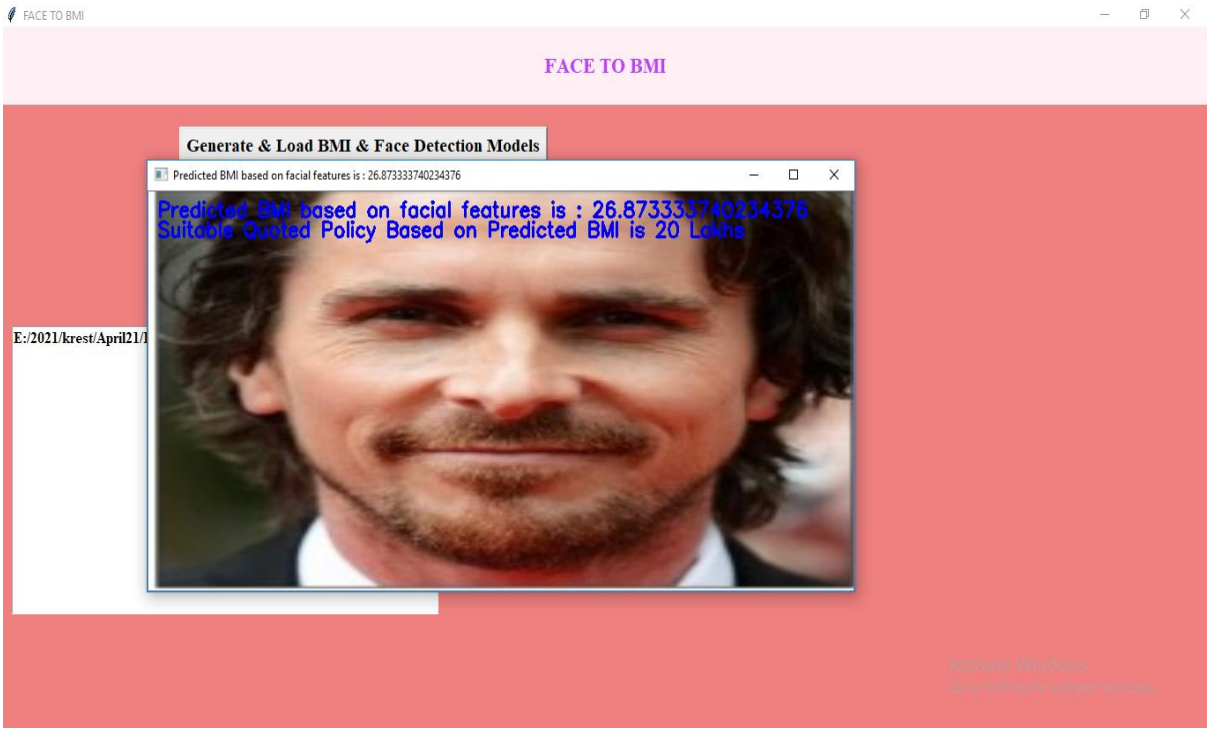
In above screen selecting and uploading 2.png file and then click on ‘Open’ button and then click on ‘Run Face & BMI Detection Algorithm’ button to get below screen




In above screen for uploaded image predicted BMI from face is 19.24 and suggested policy amount is 25 lakhs and similarly you can upload other images and test




In above screen for uploaded image predicted BMI from face is 26.48 and suggested policy amount is 20 lakhs and similarly you can upload other images and test




In above screen for uploaded image predicted BMI from face is 26.87 and suggested policy amount is 20lakhs and similarly you can upload other images and test

# CHAPTER 7

# CONCLUSIONS AND FUTURE SCOPE

# CHAPTER 7

# CONCLUSIONS AND FUTURE SCOPE

## 7.1 Conclusion

We observed that people with more BMI have a higher risk of developing health issues. We found that there exists a strong association between BMI and the face of a human. So, we proposed an approach to predict BMI from facial images using deep learning. We preprocessed the facial data by aligning the faces to the center using the BMI detection algorithm.

In future work, we will apply our method to social media profile pictures to model population-level obesity rates. Preliminary results show that both regional and demo graphic differences in BMI are reflected in large amounts of Instagram profile pictures.

# BIBLIOGRAPHY

1. [Girshick and others 2014] Girshick, R., et al. 2014. Richfeature hierarchies for accurate object detection and seman-tic segmentation. In CVPR, 580–587.
2. [Guntuku and others 2015] Guntuku, S. C., et al. 2015. Doothers perceive you as you want them to?: Modeling person-ality based on selfies. In ASM, 21–26.
3. [Henderson and others 2016] Henderson, A. J., et al. 2016.Perception of health from facial cues. Philosophical Trans-actions of the Royal Society of London B: Biological Sci-ences 371(1693).
4. [Krizhevsky and others 2012] Krizhevsky, A., et al. 2012.Imagenet classification with deep convolutional neural net-works. In NIPS, 1097–1105.
5. [Little and Perrett 2007] Little, A. C., and Perrett, D. I. 2007.Using composite images to assess accuracy in personality at-tribution to faces. British Journal of Psychology 98(1):111–126.
6. [Liu and others 2016] Liu, L., et al. 2016. Analyzing person-ality through social media profile picture choice. In ICWSM,211–220.