# CHAPTER 1 INTRODUCTION

## 1.1 Abstract

The OMR sheet is read using the typical OMR machine. Optical mark recognition is a technique for interpreting information that people mark on surveys, examinations, and other paper documents (also known as optical mark reading or OMR).OMR is used to interpret shaded regions on surveys and multiple-choice exam papers. To address this issue, we designed and built a model that allows an instructor or anyone to scan an OMR sheet and obtain information regarding marks and data. We concentrated on shaded regions and used OPEN CV to check real shaded markings to limit the danger of manipulation. In this description, we develop a scanning model that is capable of identifying and delivering information as well as creating proper marks and grades for instructors or other individuals that utilize this model. We also assess the model's performance by investigating the OMR sheet identification procedure.

## 1.2 What is OMR

The acronym “OMR” stands for Optical Mark Recognition[1]. This popular and highly accurate recognition technology is used for collecting data from “fill-in-the-bubble” types of questions on student tests, surveys, ballots, assessments, evaluations, and many other types of forms. Optical Mark Recognition (OMR) is a technique for detecting specific "marks" on an image and utilizing those markings as a reference point to extract other regions of interest (ROI) on the page. Optical Mark Recognition enables the respondent to select an answer to a question by filling in a “bubble” or “mark” associated with an answer choice.

The OMR technology has evolved dramatically in recent years. OMR technology is used everywhere, including in schools, universities, and classrooms. Exams currently employ an OMR answer sheet checking system because it makes exam administration simple, powerful, and inexpensive. The answer sheet layout is created utilizing sheet design based on our specifications. The completed document is scanned by the scanner. A created software checks an answer sheet and displays the findings.

OMR is a technique that detects the presence or absence of a mark but not its form. OMR software comprehends scanner output and translates it to ASCII output. A scanner scans forms that include filled little circles known as bubbles. OMR can detect the presence of written markings or filled bubbles by detecting the darkness on the page.

The OMR response sheet is first scanned, and the image of the answer sheet is then entered into the software system. Image processing assists us in discovering solutions to all of our queries by locating the region of interest. The recommended system is designed to be very simple to use;

The university atmosphere is the ideal application location for OMR systems. Aside from that, the government and the medical industry are important application areas for OMR systems.

### Uses of OMR

- Surveys
- Insurance and Banking applications
- Examinations
- Election
- Evaluation & Feedback form

## 1.3 What is OPTICAL MARK RECOGNITION (OMR) MCQ Automated Grading

Optical mark recognition (OMR) MCQ Automated Grading[2] is a Python & Open CV-based AI model that can operate on any Python compiler. It accepts an image of an answered omr sheet sheet as input, processes the sheet considered the dark shaded area as answer compare the list of shaded area(filled bubbles) with the referenced input list of answers and give the outputs as well as grades the omr sheet. We can load the omr sheet image from dataset or captured from the webcam this project model will process the input image and also the that input image.

Some people do not fill in the dark colours, or they fill in two hues in one column this type of answered omr sheet will not scan properly by using manual omr machine and it will be a lengthy process but we have advantage in this model to scan the image again and again until we get the correct output because it comsumes the less time as compare to OMR scanning machine. The model is easy to use as well as handy. it does not require any particular training to operate, and it will be highly cost-effective in future.

## 1.4 Objective of the study

- Identify the conventional methods and issues to acquire data about any omr sheet.
- Checking correct dark shades of OMR sheet[3] using open cv.
- Test performance analysis of all dark shades in every single column factor for the feasibility of the model.
- Design a fast and quick model for teachers or any person.

## 1.5 Advantages of OMR MCQ Automated Grading Model

- OMR MCQ automated gradding model[3] is very easy to use and handy as compared OMR scanning machine
- We must insert our omr sheet into the OMR machine, which then scans it. It also requires software to link the omr machine to the system in order to provide the output (scanned image of the OMR sheet and grades) of the OMR sheets. In this model we only need to load image from dataset and we can also use captured image from webcam rest of the process will done by the model like loading image,edge detection, contours etc
- OMR scanning models, on the other hand, scan the image and grade omr sheet as output.
- An OMR MCQ automated gradding model is cheaper than omr machine.
- In some cases, the omr machine may skip bubbles in the OMR sheet, but in the omr mcq automated gradded model, we need a data set or a live image of the OMR sheet, and it will convert it to grey and then contour[11] the filled bubbles as there are present correct answer bubbles, and it compares contour bubbles with the pre-setup correct answers and gives the grades because there are fewer chances of mistakes in the omr mcq automated gradding model.

# CHAPTER 2: TECHNOLOGIES USED

## 2.1 Python

Python[4] is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

## 2.2 Characteristics of Python

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.

## 2.3 Applications of Python

- **Easy-to-learn** − Python has few keywords, a simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

## 2.4 Artificial intelligence (AI)

Artificial intelligence (AI)[5] is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by animals including humans. AI research has been defined as the field of study of intelligent agents, which refers to any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

The term "artificial intelligence" had previously been used to describe machines that mimic and display "human" cognitive skills that are associated with the human mind, such as "learning" and "problem-solving". This definition has since been rejected by major AI

researchers who now describe AI in terms of rationality and acting rationally, which does not limit how intelligence can be articulated.

AI applications include advanced web search engines (e.g., Google), recommendation systems (used by YouTube, Amazon and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Tesla), automated decision-making and competing at the highest level in strategic game systems (such as chess and Go).

Artificial intelligence (AI) is the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans because they require human intelligence and discernment. Although there are no AIs that can perform the wide variety of tasks an ordinary human can do, some AIs can match humans in specific tasks.

### 2.4.1 Types of Artificial Intelligence:

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.

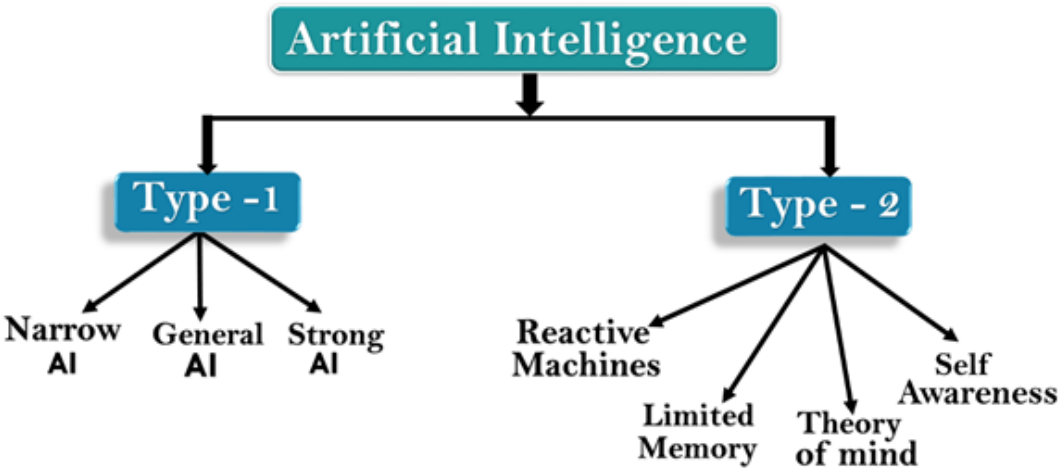



Fig 1.1 Types Of AI(Source Code-Javatpoint)

## AI type-1: Based on Capabilities

### 1. Weak AI or Narrow AI:

- Narrow AI is a type of AI which is able to perform a dedicated task with intelligence.The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
- Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
- Apple Siriis a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
- IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
- Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

### 2. General AI:

- General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

### 3. Super AI:

- Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
- Some key characteristics of strong AI include capability include the ability to think, to reason,solve the puzzle, make judgments, plan, learn, and communicate by its own.

- Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.

### Artificial Intelligence type-2: Based on functionality

#### 1. Reactive Machines

- Purely reactive machines are the most basic types of Artificial Intelligence.
- Such AI systems do not store memories or past experiences for future actions.
- These machines only focus on current scenarios and react on it as per possible best action.
- IBM's Deep Blue system is an example of reactive machines.

## 2.4.2 Application of AI

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

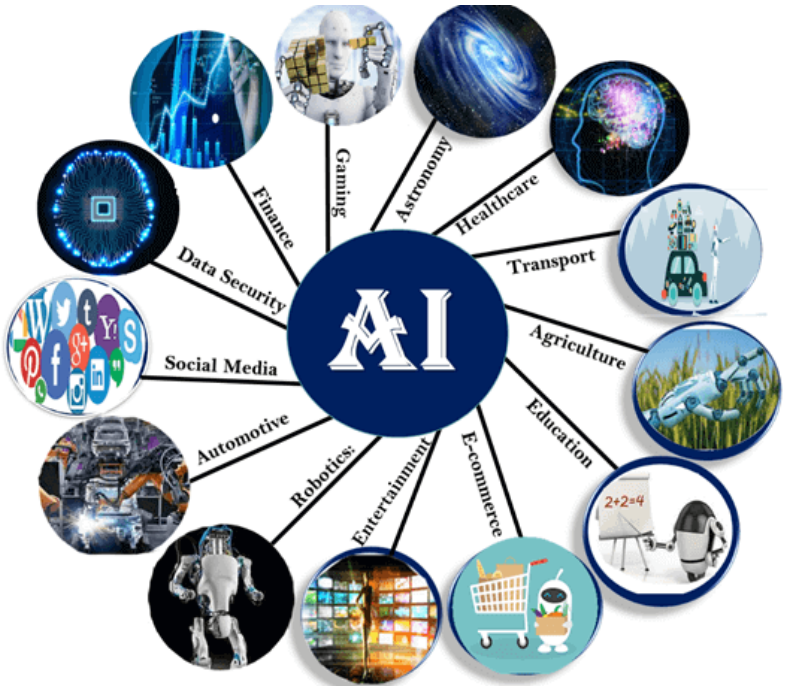



Fig 1.2 Application of AI(Source Code-Javatpoint)

## 1. AI in Astronomy

- Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

## 2. AI in Healthcare

- In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
- Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

## 3. AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

## 4. AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

## 5. AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

## 6. AI in Social Media

- Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

## 7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

## 8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

## 9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

## 10. AI in Entertainment

- We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

## 11. AI in Agriculture

- Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

## 12. AI in E-commerce

- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

### 13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

## 2.5 NumPY

NumPy[6] is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms basic linear algebra, basic statistical operations, random simulation and much more. The array object is fundamental to the NumPy library. This wraps n-dimensional arrays of homogenous data types, with many operations performed in compiled code for speed. There are some significant differences between NumPy arrays and ordinary Python sequences.

- Unlike Python lists, NumPy arrays have a fixed size when they are created (which can grow dynamically). When the size of an array is changed, a new array is created and the old one is deleted.
- A NumPy array's elements must all be of the same data type and consequently have the same memory size. The exception is that arrays of (Python, including NumPy) objects can be used, allowing for arrays with varying element sizes.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
- A rising number of scientific and mathematical Python-based programs use NumPy arrays; while they often accept Python-sequence input, they convert it to NumPy arrays before processing it, and they frequently output NumPy arrays. In other words, knowing how to utilize Python's built-in sequence types is insufficient for efficiently using most (or even most) of today's scientific/mathematical Python-based software; understanding how to use NumPy arrays is also required.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with

the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element:

```
c = []

for i in range(len(a)):

    c.append(a[i]*b[i])
```

This produces the correct

answer, but if a and b each contain millions of numbers, we will pay the price for the inefficiencies of looping in Python. We could accomplish the same task much more quickly in C by writing (for clarity we neglect variable declarations and initializations, memory allocation, etc.)

```
for (i = 0; i < rows; i++): {

   c[i] = a[i]*b[i];

}
```

This saves all the overhead involved in interpreting the Python code and manipulating Python objects, but at the expense of the benefits gained from coding in Python. Furthermore, the coding work required increases with the dimensionality of our data. In the case of a 2-D array, for example, the C code (abridged as before) expands to

```
for (i = 0; i < rows; i++): {

 for (j = 0; j < columns; j++): {

 c[i][j] = a[i][j]*b[i][j];

 }

}
```

NumPy gives us the best of both worlds: element-by-element operations are the “default mode” when an ndarray is involved, but the element-by-element operation is speedily executed by pre-compiled C code. In NumPy

```
c = a * b
```

does what the earlier examples do, at near-C speeds, but with the code simplicity, we expect from something based on Python. Indeed, the NumPy idiom is even simpler! This last example illustrates two of NumPy's features which are the basis of much of its power: vectorization and broadcasting.

### 2.5.1 Why is NumPy Fast?

Vectorization refers to the absence of explicit looping, indexing, and so on in the code; these things, of course, occur "behind the scenes" in optimized, pre-compiled C code. Among the numerous benefits of vectorized code are:

- vectorized code is more concise and easier to read
- fewer lines of code generally mean fewer bugs
- the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)
- vectorization results in more "Pythonic" code. Without vectorization, our code would be littered with inefficient and difficult reads for loops.

Broadcasting is the phrase used to describe the implicit element-by-element behaviour of operations; in general, all operations in NumPy, including arithmetic, logical, bit-wise, functional, and so on, operate in this implicit element-by-element way, i.e., broadcast. Furthermore, in the above example, a and b might be multidimensional arrays of the same form, a scalar and an array, or even two arrays of different shapes, as long as the smaller array is "expandable" to the shape of the bigger in such a way that the final broadcast is clear. See basics. broadcasting for further information on the "rules" of broadcasting.

### 2.5.2 Who Else Uses NumPy?

NumPy completely supports an object-oriented approach, beginning with ndarray once more. For example, an array is a class with several methods and properties. Many of its methods are replicated by functions in the NumPy namespace's outermost namespace, allowing the programmer to code in the paradigm of their choice. Because of this versatility, the NumPy array dialect and NumPy array class have become the de-facto language of multi-dimensional data transfer in Python.

## 2.6 Open CV-

OpenCV (Open Source Computer Vision Library) [7] is a programming function library geared mostly at real-time computer vision. It was created by Intel and was subsequently sponsored by Willow Garage and Itseez (which was later acquired by Intel). The library is cross-platform and available for free under the Apache 2 License. Since 2011, OpenCV has supported GPU acceleration for real-time tasks.

OpenCV is built in C++ and has a C++ interface as its primary interface, but it also has a less thorough but still significant older C interface. All new advances and algorithms are visible in the C++ interface. Python, Java, and MATLAB/OCTAVE bindings are available. The online documentation contains the API for these interfaces. Wrappers in a variety of programming languages have been created to enable greater use. JavaScript bindings for a subset of OpenCV functions were provided as OpenCV.js in version 3.4, to be utilized on online platforms.

If the library discovers Intel's Integrated Performance Primitives on the system, it will accelerate itself using these proprietary optimized procedures.Since September 2010, a CUDA-based GPU interface has been in development.

Since October 2012, an OpenCL-based GPU interface has been under development; documentation for version 2.4.13.3 is available at docs.opencv.org.

Windows, Linux, macOS, FreeBSD, NetBSD, and OpenBSD are the desktop operating systems that support OpenCV. OpenCV is compatible with the following mobile operating systems: Android, iOS, Maemo, and BlackBerry 10. The user can obtain official releases from SourceForge or the most recent source code from GitHub. CMake is used by OpenCV.

### 2.6.1 Open CV Usages-

**Image Filtering:** It is a technique for modifying or enhancing an image. Image filtering is of two types. The one is linear image filtering, in which, the value of an output pixel is a linear combination of the values of the pixels of the input pixel's neighborhood. The second one is the non-linear image filtering, in which, the value of output is not a linear function of its input.

**Image Transformation:** Image transformation generates "new" image from two or more sources which highlight particular features or properties of interest, better than the

original input images. Basic image transformations apply simple arithmetic operations to the image data. Image subtraction is often used to identify changes that have occurred between images collected on different dates. Main image transformation methods are

**Hough Transform:** used to find lines in an image

**Radon Transform:** used to reconstruct images from fan-beam and parallel-beam projection data

**Discrete Cosine Transform:** used in image and video compression

**Discrete Fourier Transform:** used in filtering and frequency analysis

**Wavelet Transform:** used to perform discrete wavelet analysis, denoise, and fuse images

**Object Tracking:** Object tracking is the process of locating a object (or multiple objects) over a sequence of images. It is one of the most important components in a wide range of applications in computer vision, such as surveillance, human computer interaction, and medical imaging.

**Feature Detection:** A feature is defined as an "interesting" part of an image and features are used as a starting point for many computer vision algorithms. Since features are used as the starting point and main primitives for subsequent algorithms, the overall algorithm will often only be as good as its feature detector. Feature detection is a process of finding specific features of a visual stimulus, such as lines, edges or angle. It will be helpful for making local decisions about the local information contents (image structure) in the image.

The modules of openCV for image processing applications are given below

**a.CORE** module contains basic data structures and basic functions used by other modules.

**b.IMGPROC** module contains image processing related functions such as linear, non-linear image filtering and geometrical image transformations etc.

**c.VIDEO** module contains motion estimation and object tracking algorithms.

**d.ML** module contains machine-learning interfaces.

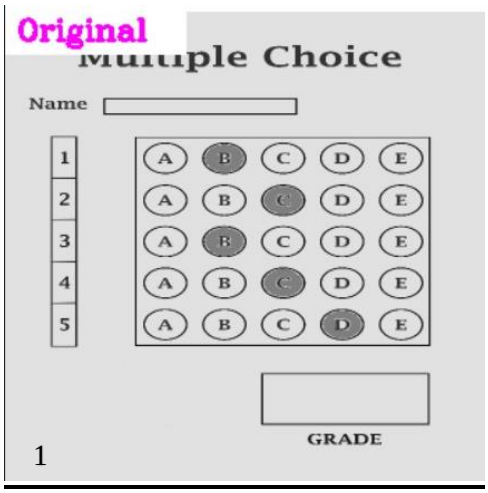
**e.HighGUI** module contains the basic I/O interfaces and multi-platform windowing capabilities

**Video Processing using Android Phone**

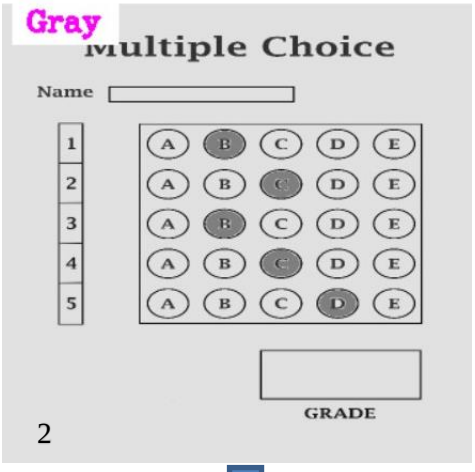
As mobile devices such as smart phones, iPads and tablet pcs are equipped with cameras, the demand of the image processing applications increased. These applications need to be faster and consumes lower power because the mobile device is only powered by a battery. The easy way to increase the performance of the device is to replace the older hardware with newer hardware. The hardware technology depends on the semiconductor technology. If a semiconductor chip contains more number of transistors, the density of transistors is increased. As the density is increased, the leakage of current is also increased. Hence we have to choose an efficient programming language to write an image processing application for the mobile devices.

In these days, android became the famous operating system for the mobile devices. Android developers introduce new application to satisfy the needs of the Smartphone users. Libraries such as OpenGL (Open Graphics Library) and OpenCV (Open Computer Vision) are used for the development of the application . Application that uses camera usually involves an image processing method such as Gaussian, Median, Mean Laplacian, Sobel filter and others. In 2010, a new module is introduced in openCV to deal with GPU acceleration. OpenCV implements a container for images called cv::Mat that exposes access to image raw data. In the GPU module the container cv::gpu::GpuMat stores the image data in the GPU memory

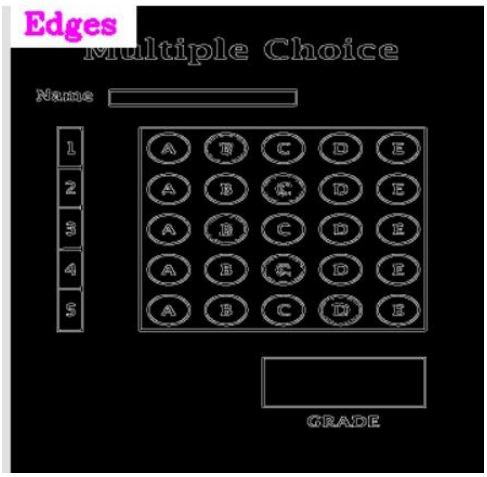
# CHAPTER 3 METHODOLOGY



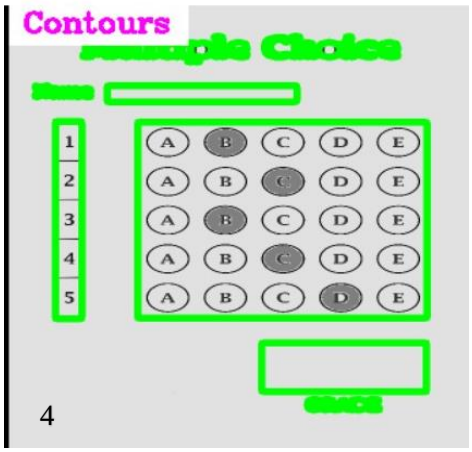
1



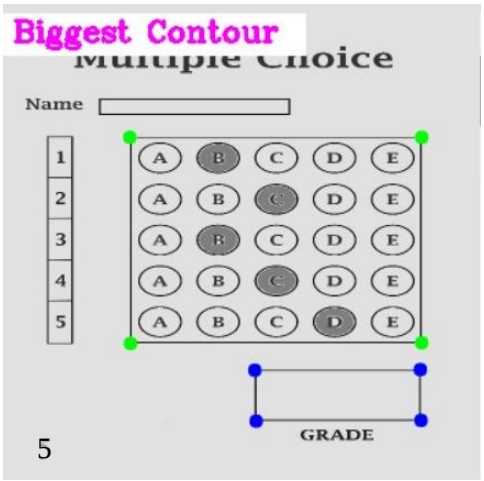
2



3



4



5

6

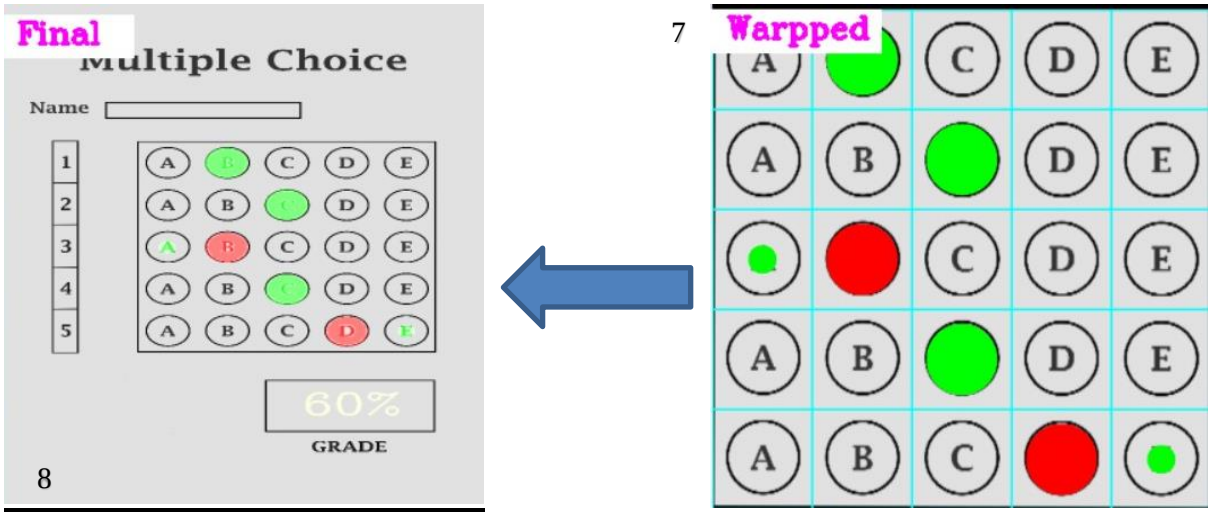



**Fig 2 Model Running steps**

**Fig 1: Original Image→** It is the original MCQ sheet to scan.

**Fig 2: Gray image→** Converted Original image to Gray

**Fig 3: Edges→** Finding edges

**Fig 4: Contour→** converted Images to green.

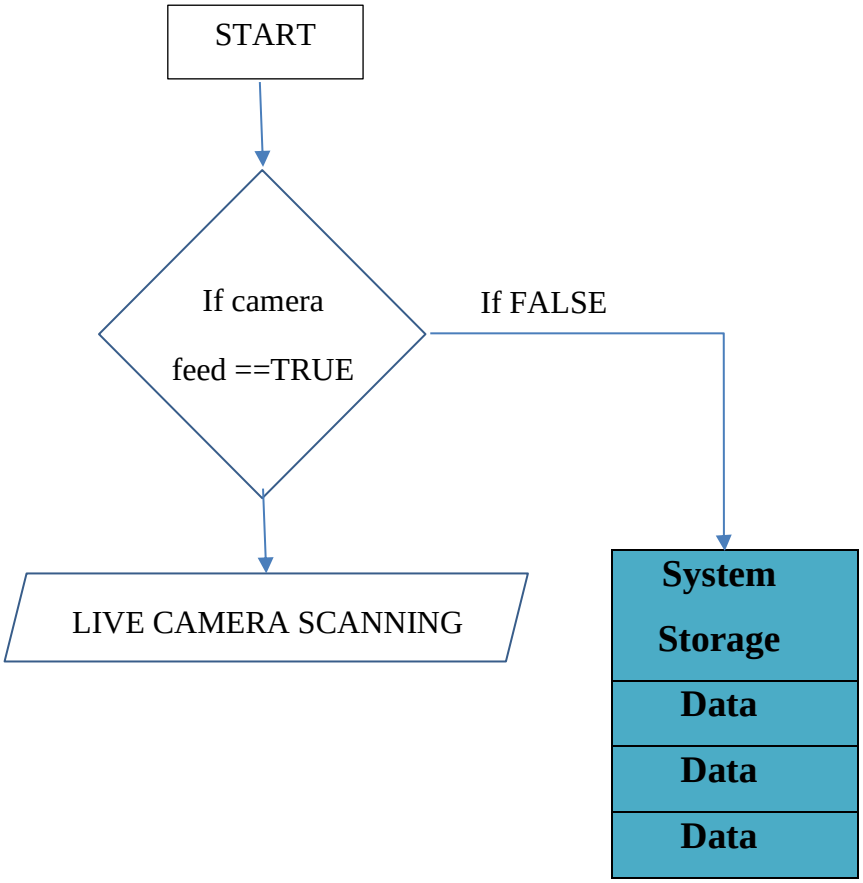
**Fig 5: Biggest contour→** Edges Pointed to green

**Fig 6: Threshold→** will Darken the answer in the image

**Fig 7: Wrapped→**differentiate between right & wrong answer

**Fig 8: Final→**Finalize the Result by Right & Wrong answers.

## Model Flowchart-




**Model Flowchart**

# CHAPTER 4 PROJECT OVERVIEW

In Optical mark recognition (OMR) MCQ Automated Grading we used OpenCV and python and Python library like Numpy for numeric calculation that can operate on any Python compiler. It accepts an image of an answered omr sheet and captured image from the webcam as input and after processing and applying the modes like grey scale, contour, edge dectoretc. The OMR MCQ automated gradding model will considered the dark shaded area as answer compare the list of shaded area(filled bubbles) with the referenced input list of answers and give the outputs as well as grades the omr sheet. We can load the omr sheet image from dataset or captured from the webcam this project model will process the input image and also the that input image.

Some people do not fill in the dark colours, or they fill in two hues in one column this type of answered omr sheet will not scan properly by using manual omr machine and it will be a lengthy process but we have advantage in this model to scan the image again and again until we get the correct output because it comsumes the less time as compare to OMR scanning machine. The model is easy to use as well as handy. it does not require any particular training to operate, and it will be highly cost-effective in future.

## 4.1 Algorithim and Implementation:

This stage will discuss detailed working of the OMR MCQ Automated Gradding Model and will cover basic institution how the model is working:

• Step 1: Here it first capture the veideo/image from the webcam and or load image from the dataset from the given path.

• Step 2: Then Model will convert the captured or loaded image in grey color by using grey scale.

• Step 3: Model will find the edges from the grey scaled image using canny.

• Step 4: Model will apply the contours over the grey scaled image.

• Step 5: Model will find the biggest rectangles present in the image and their corner points

• Step 6: Then it will use wrap IPU which is the wrap perspective on the image from this we get the required portion of the image

• Step 7: Later model will apply some threshold on the image for the better visiualization of the dark shaded bubbles.

• Step 8:  Model will find the filled bubbles and compare the list of  input with the reference list which we defined in the model. And generate the output.

• Step 9: lastly, we need to enter the “s” key on the keyboard to save the output.

# CHAPTER 5 FEATURES

In this work, the system of OMR has been developed out using Python software language with OpenCV. Python is a programming language written by a Dutch programmer named Guido Van Rossum. Python; is an ndependent platform, interpretable, interactive object-oriented high-level programming language . In this language; There is no need for a compiler as in C, C ++, Java. OpenCV (Open Source Computer Vision) is an open source library introduced by intel for image and video analysis. On the image obtained with a scanner, the following operations were performed in order:

## 5.1 Loading the data:

Loading data means fetchin reqired data from particular location and from input devices.

In this project model will load tha data that is an image which is from the given path of dataset and capture from webcam

## 5.2 Data Preprocessing

Data preprocessing is the process of preparing raw data for use by a Arttificial Intelligence model. It is the first and most important stage in developing any Artificial Intelligence model.

When developing a project of artificial intelligence by using Opencv, Numpy Libraries of it, we do not always have access to clean and prepared data. And, before doing any action with data, it must be cleaned and formatted. So we use the data preprocessing job for our Model.

## Why do we need Data Preprocessing?

Real-world data typically contains noise, and missing values, and may be in an unsuitable format that cannot be utilized directly for artificial intelligence models. Data preprocessing is a necessary job for cleaning the data and preparing it for a artificial intelligence model, which improves the accuracy and efficiency of the model.

- Resize: Resize the input image of the omr sheet define the height and with as per the requirement.

## 5.3 Data Processing:

In the data processing the data is transform into the useable information. In we enhance the colr quality of the pixels of the image and apply som required operation on the data.

Grey Scaling: In this image will convert into the grey in color by applying grey scaling the pixels value are appointed as 0, that is, black; the others as 255, that is, white.

**5.4 Edge Detection:**After applying Gaussian blur to blur the image The technique[8] is used in order to define the borders of the objects on the image named as Canny Edge Detector and used to find the corners, this method was developed by J. Canny in 1986. Canny Edge Detector is one of the most commonly used image processing devices that detects the edges very precisely. It is considered to be as the standard edge detecting method in industry. Canny accepted the edge detecting problem as a problem of signal processing optimization, thus developed it in order to optimize an objective function.

**1.Contours:** Contour is used to contoured the image the highlight the boreder of image what the imager is contain.

**2.Corner Points:**After applying the contour we find the corner point of the required region og the image. The rectangles.

**3.Threshold:** threshold[9] is used to darken the filled region the omr.

**4.Warp Perspective:** we used the warp perspective for the better insights that we can get from the given information of the image.

**5.Answer Shottiong:** to sort thr answer we model compare the input list of the referenced answer with the anwers which we get from the precessed image.

**6.Gradding:** after answer shortiog we apply some which are mentioned in utlis file to get the final output that is gradded answer omr sheet.

Save Image by entering thr ‘s’ key on keyboard the will saved.

# CHAPTER 6 -PROJECT MODULES

## 6.1 OMR_Main.py

OMR_Main.py is main python file which contains all the important codes of the model like code for webcam maodule,image laoading, Preprocessing, Contours, Warp perspective etc. it is this first python file for the OMR MCQ Automated Gradding Model

### 6.1.1 Webcam Module/Answer Module:

**Description:**

The webcam module/ Answer module will enable the webcam to detect the image and capture the video of the image or load the image from the dataset. If it needs to resize the image, we can resize the image which capture by the webcam in this module. In this module we also set the reference answers of the OMR Sheet which we will further use when we input a new Omr sheet from dataset or by using the webcam and also set the number of questions and the choices.

**Code for Webcam Module/Answer Module:**

```
################################################################################
webCamFeed = False
pathImage = "14.jpg"
cap = cv2.VideoCapture(0)
cap.set(10,160)
heightImg = 700
widthImg  = 700
questions=5
choices=5
ans= [1,2,0,2,4]
################################################################################
```

### 6.1.2 Preprossing Module:

**Description:**

In Preprocessing module, it will load the image from the dataset or use the captured image from the webcam. Image will resize in this module using the OpenCV library's resize function. We will also create a blank image for testing and debugging if it will be required. Then by using grey scale convert the image into grey. After getting the grey image we will apply the gaussian blur to blur the image. On the blur image we apply canny for the edge detection to extract the

useful information from the image like the edges of the rectangle present in the image, the corner points and the rectangles present int the image.

### Code for Preprocessing Module:

```
count=0

while True:

    if webCamFeed:success, img = cap.read()
    else:img = cv2.imread(pathImage)
    img = cv2.resize(img, (widthImg, heightImg)) # RESIZE IMAGE
    imgFinal = img.copy()
    imgBlank = np.zeros((heightImg,widthImg, 3), np.uint8) # CREATE A BLANK IMAGE FOR TESTING DEBUGGING IF REQUIRED
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # CONVERT IMAGE TO GRAY SCALE
    imgBlur = cv2.GaussianBlur(imgGray, (5, 5), 1) # ADD GAUSSIAN BLUR
    imgCanny = cv2.Canny(imgBlur,10,70) # APPLY CANNY
```

### 6.1.3 Contour Module:

### Description:

In the Contour Module model will apply Contour over the pre-processed image which load from dataset or capture from webcam, like first it will copy the images for the display purpose then find all the Contour, Draw the all-detected contours and in the last we need to apply filter for the rectangle contours then we will get the corner points for the biggest rectangle and the second biggest rectangle.

### Code for the Contour Module:

```
try:
    ## FIND ALL COUNTOURS
    imgContours = img.copy() # COPY IMAGE FOR DISPLAY PURPOSES
    imgBigContour = img.copy() # COPY IMAGE FOR DISPLAY PURPOSES
    contours, hierarchy = cv2.findContours(imgCanny, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE) # FIND ALL CONTOURS
    cv2.drawContours(imgContours, contours, -1, (0, 255, 0), 10) # DRAW ALL DETECTED CONTOURS
    rectCon = utlis.rectContour(contours) # FILTER FOR RECTANGLE CONTOURS
    biggestPoints= utlis.getCornerPoints(rectCon[0]) # GET CORNER POINTS OF THE BIGGEST RECTANGLE
    gradePoints = utlis.getCornerPoints(rectCon[1]) # GET CORNER POINTS OF THE SECOND BIGGEST RECTANGLE
```

### 6.1.4 Warp Perspective:

### Description:

In Warp Perspective Module, we will use the Word IPU for the wrap perspective. The warp perspective will we apply on contoured images that we will get after the contour module. First, we need to apply the wrap perspective on the contoured biggest rectangle the apply wrap perspective on the contoured second biggest rectangle present in the image.

Second step will be the reorder for the warpping, then draw the biggest contour and prepare the points for the warp. We will get the transformation matrix and apply the warp perspective as the last step of the wrapping perspective. We will do the same warp perspective process and follow the previous steps again for the second biggest rectangle

### Code for Wrap Perspective:

```
if biggestPoints.size != 0 and gradePoints.size != 0:

    # BIGGEST RECTANGLE WARPING
    biggestPoints=utlis.reorder(biggestPoints) # REORDER FOR WARPING
    cv2.drawContours(imgBigContour, biggestPoints, -1, (0, 255, 0), 20) # DRAW THE BIGGEST CONTOUR
    pts1 = np.float32(biggestPoints) # PREPARE POINTS FOR WARP
    pts2 = np.float32([[0, 0],[widthImg, 0], [0, heightImg],[widthImg, heightImg]]) # PREPARE POINTS FOR WARP
    matrix = cv2.getPerspectiveTransform(pts1, pts2) # GET TRANSFORMATION MATRIX
    imgWarpColored = cv2.warpPerspective(img, matrix, (widthImg, heightImg)) # APPLY WARP PERSPECTIVE

    # SECOND BIGGEST RECTANGLE WARPING
    cv2.drawContours(imgBigContour, gradePoints, -1, (255, 0, 0), 20) # DRAW THE BIGGEST CONTOUR
    gradePoints = utlis.reorder(gradePoints) # REORDER FOR WARPING
    ptsG1 = np.float32(gradePoints)  # PREPARE POINTS FOR WARP
    ptsG2 = np.float32([[0, 0], [325, 0], [0, 150], [325, 150]])  # PREPARE POINTS FOR WARP
    matrixG = cv2.getPerspectiveTransform(ptsG1, ptsG2)# GET TRANSFORMATION MATRIX
    imgGradeDisplay = cv2.warpPerspective(img, matrixG, (325, 150)) # APPLY WARP PERSPECTIVE
```

### 6.1.5 Threshold Module:

### Description:

In Threshold Module, we will apply some threshold on over the image which we get after warp perspective module we get the only rectangle area after the wrapping process, we do the further process on this image we need to convert into grey in color before applying the threshold on over it to the image into grey we used the greyscale then we applied the threshold on the image. we apply threshold to detect bubbles that marked as answers.

## Code for the Threshold Module:

```
# APPLY THRESHOLD
imgWarpGray = cv2.cvtColor(imgWarpColored,cv2.COLOR_BGR2GRAY) # CONVERT TO GRAYSCALE
imgThresh = cv2.threshold(imgWarpGray, 170, 255,cv2.THRESH_BINARY_INV )[1] # APPLY THRESHOLD AND INVERSE
```

### 6.1.6 Find User Answer Module:

### Description:

In Find User Answer Module, when threshold process is done, we split the boxes (the rectangle which contain the marked answers of the omr sheet). Then we stored the non-zero values of the box because the non-zero values are the value of non-zero pixels which is the area of marked answer from this our model will able to detect area of the answer sheet is marked or not this will also give the matrix value of the non-zero pixels.

to find answer that highest value will be considered as the answer. we get the highest non-zero value in the rows of the matrix that highest values are our marked answer of the user and that highest value will be considered as the answer.

And lastly, we put them in the list for the convenient output the list will be a array that contain only non-zero highest value (marked answer).

## Code for the Find User Answerer Module

```
boxes = utlis.splitBoxes(imgThresh) # GET INDIVIDUAL BOXES
cv2.imshow("Split Test ", boxes[3])
countR=0
countC=0
myPixelVal = np.zeros((questions,choices)) # TO STORE THE NON ZERO VALUES OF EACH BOX
for image in boxes:
    #cv2.imshow(str(countR)+str(countC),image)
    totalPixels = cv2.countNonZero(image)
    myPixelVal[countR][countC]= totalPixels
    countC += 1
    if (countC==choices):countC=0;countR +=1

# FIND THE USER ANSWERS AND PUT THEM IN A LIST
myIndex=[]
for x in range (0,questions):
    arr = myPixelVal[x]
    myIndexVal = np.where(arr == np.amax(arr))
    myIndex.append(myIndexVal[0][0])
#print("USER ANSWERS",myIndex)
```

### 6.1.7 Compare the Answers Module:

### Description:

In the Compare Answer module, we will compare the list of marked that we got after the previous find the user answer module. Now we will compare the list of marked answer with Correct answer list that we define in the webcam module then after the comparison of both the list and print the score in the terminal.

### Code for the Compare answer Module:

```
# COMPARE THE VALUES TO FIND THE CORRECT ANSWERS
grading=[]
for x in range(0,questions):
    if ans[x] == myIndex[x]:
        grading.append(1)
    else:grading.append(0)
#print("GRADING",grading)
score = (sum(grading)/questions)*100 # FINAL GRADE
print("SCORE",score)
```

### 6.1.8 Display Answer Module:

### Description:

In display answer module, we display the answers of the omr sheet image. To display the answer fist, we will draw detected answers and high light them in neon-green color if they are correct and in red color if detected answers are wrong and draw the grid and also add a new blank image with the warp image size in this, we also get highlighted answers then we applied matrix transformation and inverse image wrap to show the highlighted colored answer on the final image.

**Code for the Display Answer Module:**

```
        # DISPLAYING ANSWERS
        utlis.showAnswers(imgWarpColored,myIndex,grading,ans) # DRAW DETECTED ANSWERS
        utlis.drawGrid(imgWarpColored) # DRAW GRID
        imgRawDrawings = np.zeros_like(imgWarpColored) # NEW BLANK IMAGE WITH WARP IMAGE SIZE
        utlis.showAnswers(imgRawDrawings, myIndex, grading, ans) # DRAW ON NEW IMAGE
        invMatrix = cv2.getPerspectiveTransform(pts2, pts1) # INVERSE TRANSFORMATION MATRIX
        imgInvWarp = cv2.warpPerspective(imgRawDrawings, invMatrix, (widthImg, heightImg)) # INV IMAGE WARP
```

### 6.1.9 Grade Module:

### Description:

In the Display Grade Module, first we will use a blank image and other window output add the grade on this blank image and after applying the inverse transformation matrix this new image which contain the grade will merge in the main output image where the grade rectangle is present.

### Code for the Display Grade Module:

```
        # DISPLAY GRADE
        imgRawGrade = np.zeros_like(imgGradeDisplay,np.uint8) # NEW BLANK IMAGE WITH GRADE AREA SIZE
        cv2.putText(imgRawGrade,str(int(score))+"%",(70,100)
                    ,cv2.FONT_HERSHEY_COMPLEX,3,(0,255,255),3) # ADD THE GRADE TO NEW IMAGE
        invMatrixG = cv2.getPerspectiveTransform(ptsG2, ptsG1) # INVERSE TRANSFORMATION MATRIX
        imgInvGradeDisplay = cv2.warpPerspective(imgRawGrade, invMatrixG, (widthImg, heightImg)) # INV IMAGE WARP

        # SHOW ANSWERS AND GRADE ON FINAL IMAGE
        imgFinal = cv2.addWeighted(imgFinal, 1, imgInvWarp, 1,0)
        imgFinal = cv2.addWeighted(imgFinal, 1, imgInvGradeDisplay, 1,0)

        # IMAGE ARRAY FOR DISPLAY
        imageArray = ([img,imgGray,imgCanny,imgContours],
                      [imgBigContour,imgThresh,imgWarpColored,imgFinal])
        cv2.imshow("Final Result", imgFinal)
except:
    imageArray = ([img,imgGray,imgCanny,imgContours],
                  [imgBlank, imgBlank, imgBlank, imgBlank])
```

### 6.1.10 Labels For Display Window Module:

### Description:

In this Module, we added the labels for the Display windows which contains different processed image like original image, grey image, contoured imaged etc in single display.

First, we add the original label in original image then add other labels as the label are required for the image like "Original", "Gray", "Edges", "Contours", “Biggest Contour", "Threshold", "Warpped", "Final"

### Code for the Labels for the Display Window Module:

```
# LABELS FOR DISPLAY
lables = [["Original","Gray","Edges","Contours"],
          ["Biggest Contour","Threshold","Warpped","Final"]]

stackedImage = utlis.stackImages(imageArray,0.5,lables)
cv2.imshow("Result",stackedImage)
```

### 6.1.11 Save Image Module:

### Description:

In save Image Module, when we enter “s” on the key board the output image with grading will save.

### Code for the Save Image Module:

```
# SAVE IMAGE WHEN 's' key is pressed
if cv2.waitKey(1) & 0xFF == ord('s'):
    cv2.imwrite("Scanned/myImage"+str(count)+".jpg",imgFinal)
    cv2.rectangle(stackedImage, ((int(stackedImage.shape[1] / 2) - 230), int(stackedImage.shape[0] / 2) + 50),
                  (1100, 350), (0, 255, 0), cv2.FILLED)
    cv2.putText(stackedImage, "Scan Saved", (int(stackedImage.shape[1] / 2) - 200, int(stackedImage.shape[0] / 2)),
                cv2.FONT_HERSHEY_DUPLEX, 3, (0, 0, 255), 5, cv2.LINE_AA)
    cv2.imshow('Result', stackedImage)
    cv2.waitKey(300)
    count += 1
```

## 6.2 Utlis.py File

Utlis.py is the second python file which useto code for the utilities of the OMR MCR Automated Gradding Model that contains all the important function and Numeric calculation codes of the model like code for slike stack all output windows together module,image contour module, corner point module Warp perspective module split important window module etc. it is this second python file for the OMR MCQ Automated Gradding Model

### 6.2.1 Stack All Images in One Window Module:

**Description:**

Stack All Images in One Window Module, this module will unable the model to stack all the processing images in the single window by which we will able to see the different stages of output in the single window and also observe the changes occurred in previous and current image of the processed output. This will be the main output window for our model.

**Codes for Stack All Images in One Window Module:**

```
## TO STACK ALL THE IMAGES IN ONE WINDOW
def stackImages(imgArray,scale,lables=[]):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None,
scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]=
cv2.cvtColor( imgArray[x][y], cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
            hor_con[x] = np.concatenate(imgArray[x])
        ver = np.vstack(hor)
        ver_con = np.concatenate(hor)
    else:
        for x in range(0, rows):
            imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale,
scale)
            if len(imgArray[x].shape) == 2: imgArray[x] =
cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
        hor= np.hstack(imgArray)
        hor_con= np.concatenate(imgArray)
        ver = hor
    if len(lables) != 0:
        eachImgWidth= int(ver.shape[1] / cols)
        eachImgHeight = int(ver.shape[0] / rows)
        #print(eachImgHeight)
        for d in range(0, rows):
            for c in range (0,cols):
```

```
cv2.rectangle(ver,(c*eachImgWidth,eachImgHeight*d),(c*eachImgWidth+len(labl
es[d][c])*13+27,30+eachImgHeight*d),(255,255,255),cv2.FILLED)

cv2.putText(ver,lables[d][c],(eachImgWidth*c+10,eachImgHeight*d+20),cv2.FON
T_HERSHEY_COMPLEX,0.7,(255,0,255),2)
    return ver
```

### 6.2.2 Contour Module:

**Description:**

Contour Module, this is module contains the functions in utlis.py file to apply contour on the image and enable the model to apply Contour over the pre-processed image which load from dataset or capture from webcam, find all the Contour, Draw the all-detected contours and in the last we need to apply filter for the rectangle contours.

**Code for the Contour Module:**

```
def rectContour(contours):

    rectCon = []
    max_area = 0
    for i in contours:
        area = cv2.contourArea(i)
        if area > 50:
            peri = cv2.arcLength(i, True)
            approx = cv2.approxPolyDP(i, 0.02 * peri, True)
            if len(approx) == 4:
                rectCon.append(i)
    rectCon = sorted(rectCon, key=cv2.contourArea,reverse=True)
    #print(len(rectCon))
    return rectCon
```

### 6.2.3 Reorder Contour Module:

**Description:**

Reorder contour module, this module is used to reorder the contour points of the image which load from dataset and captured from the webcam.

**Code for reorder contour module:**

```
def reorder(myPoints):

    myPoints = myPoints.reshape((4, 2)) # REMOVE EXTRA BRACKET
    print(myPoints)
    myPointsNew = np.zeros((4, 1, 2), np.int32) # NEW MATRIX WITH ARRANGED
POINTS
    add = myPoints.sum(1)
    print(add)
    print(np.argmax(add))
    myPointsNew[0] = myPoints[np.argmin(add)]  #[0,0]
    myPointsNew[3] =myPoints[np.argmax(add)]   #[w,h]
```

```
    diff = np.diff(myPoints, axis=1)
    myPointsNew[1] =myPoints[np.argmin(diff)]  #[w,0]
    myPointsNew[2] = myPoints[np.argmax(diff)] #[h,0]

    return myPointsNew
```

### 6.2.4 Corner Points Module:

**Description:**

Corner Point Module, this module is used to define the corner point of two rectangle the biggest rectangle which contain the answers of the omr sheet and the second biggest rectangle which contain the grade.

**Code for the Corner point module**

```
def getCornerPoints(cont):
    peri = cv2.arcLength(cont, True) # LENGTH OF CONTOUR
    approx = cv2.approxPolyDP(cont, 0.02 * peri, True) # APPROXIMATE THE
POLY TO GET CORNER POINTS
    return approx
```

### 6.2.5 Split Boxes Module:

**Description:**

Split boxes Module, this module is used to split the windows we used this in our model to split the answer image box and original image answer sheet images box from the stack window where each step image boxes stacked together.

**Code for the split Boxes Module:**

```
def splitBoxes(img):
    rows = np.vsplit(img,5)
    boxes=[]
    for r in rows:
        cols= np.hsplit(r,5)
        for box in cols:
            boxes.append(box)
    return boxes
def drawGrid(img,questions=5,choices=5):
    secW = int(img.shape[1]/questions)
    secH = int(img.shape[0]/choices)
    for i in range (0,9):
        pt1 = (0,secH*i)
        pt2 = (img.shape[1],secH*i)
        pt3 = (secW * i, 0)
        pt4 = (secW*i,img.shape[0])
        cv2.line(img, pt1, pt2, (255, 255, 0),2)
        cv2.line(img, pt3, pt4, (255, 255, 0),2)

    return img
```

### 6.2.6 Display Answer Module:

**Description:**

In display answer module, we display the answers of the omr sheet image. To display the answer fist, we will draw detected answers and draw the grid and also add a new blank image with the warp image size in this, we also get highlighted answers then we applied matrix transformation and inverse image wrap to show the highlighted colored answer on the final image.

After done the previous step in omr main.py file we will define the further function in utlis.py file to give color to the detected answer in highlighted color in neon-green color if they are correct and in red color if detected answers are wrong.

**Code for the Display Answer Module:**

```
def showAnswers(img,myIndex,grading,ans,questions=5,choices=5):
     secW = int(img.shape[1]/questions)
     secH = int(img.shape[0]/choices)

     for x in range(0,questions):
         myAns= myIndex[x]
         cX = (myAns * secW) + secW // 2
         cY = (x * secH) + secH // 2
         if grading[x]==1:
            myColor = (0,255,0)

#cv2.rectangle(img,(myAns*secW,x*secH),((myAns*secW)+secW,(x*secH)+secH),my
Color,cv2.FILLED)
            cv2.circle(img,(cX,cY),50,myColor,cv2.FILLED)
         else:
            myColor = (0,0,255)
            #cv2.rectangle(img, (myAns * secW, x * secH), ((myAns * secW) +
secW, (x * secH) + secH), myColor, cv2.FILLED)
            cv2.circle(img, (cX, cY), 50, myColor, cv2.FILLED)
```

### 6.2.7 Correct Answer Module:

**Description:**

Correct answer module, in this module it considered the green colour filled circle as correct answer and give automated grade on the omr sheet image with help function which define in utlis.py file.

**Code for the correct answer module:**

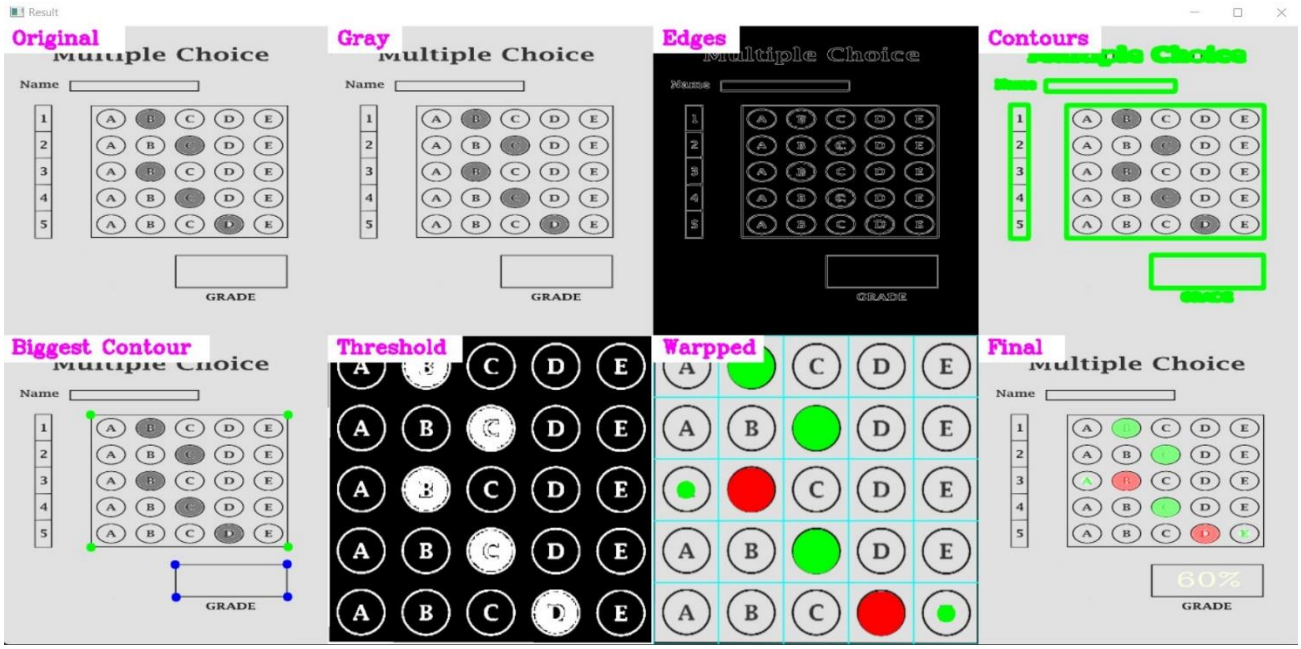
```
# CORRECT ANSWER
myColor = (0, 255, 0)
correctAns = ans[x]
```

```
cv2.circle(img,((correctAns * secW)+secW//2, (x * secH)+secH//2),
20,myColor,cv2.FILLED)
```

# CHAPTER 7 TESTING

**Part 1-**Testing via In system stored images.

## TESTING 1-



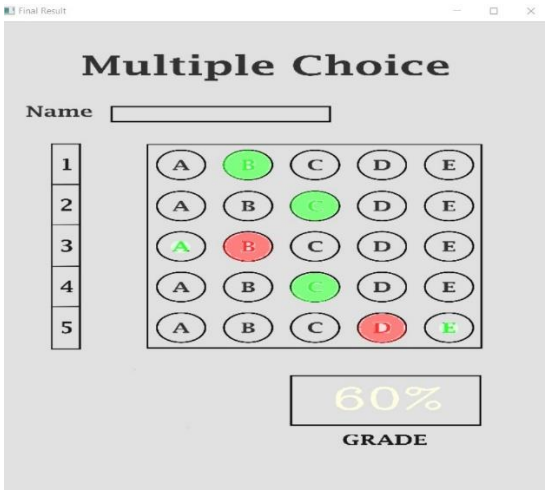
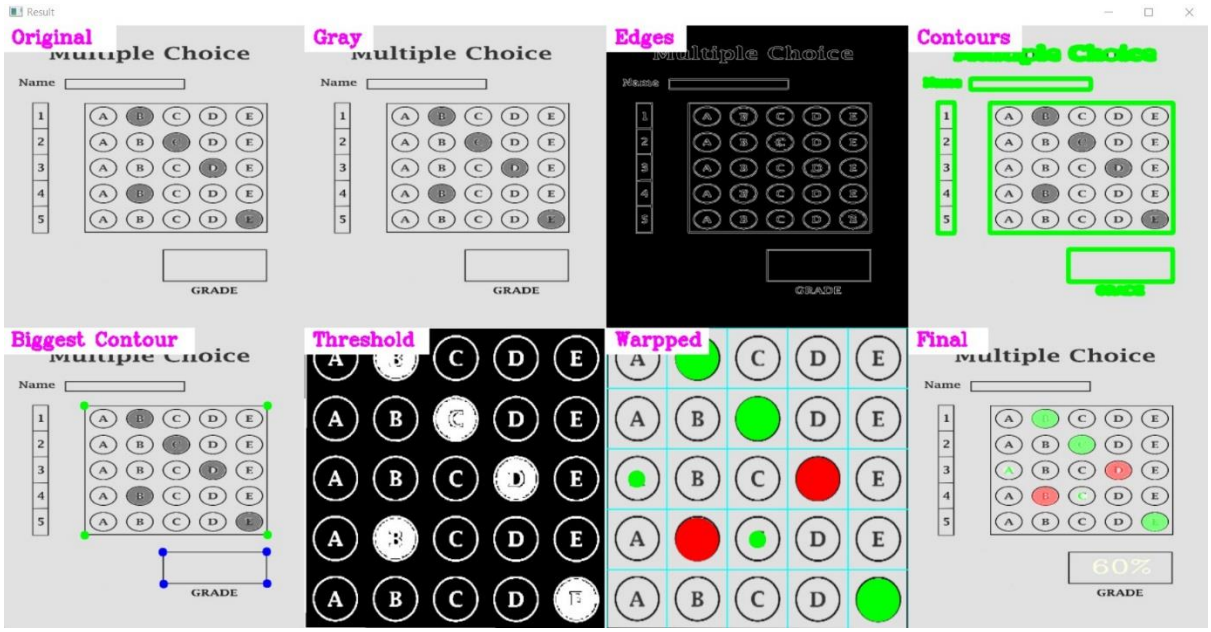




**Fig:** Multiple Choice ( It Scans the Right & Wrong answers and finalizes the result.)

## TESTING 2-



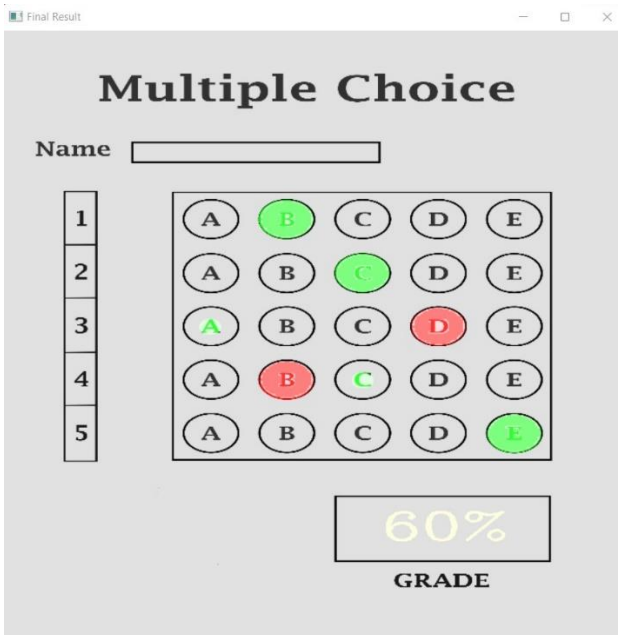
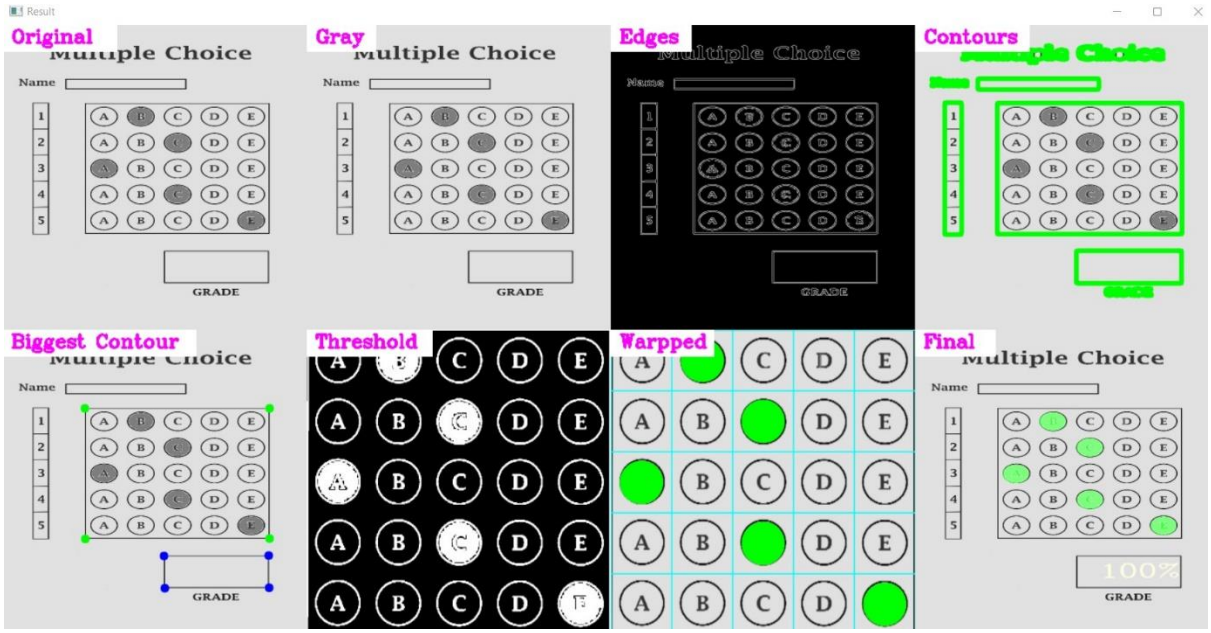




**Fig:** Shows 60% Result

## TESTING 3-



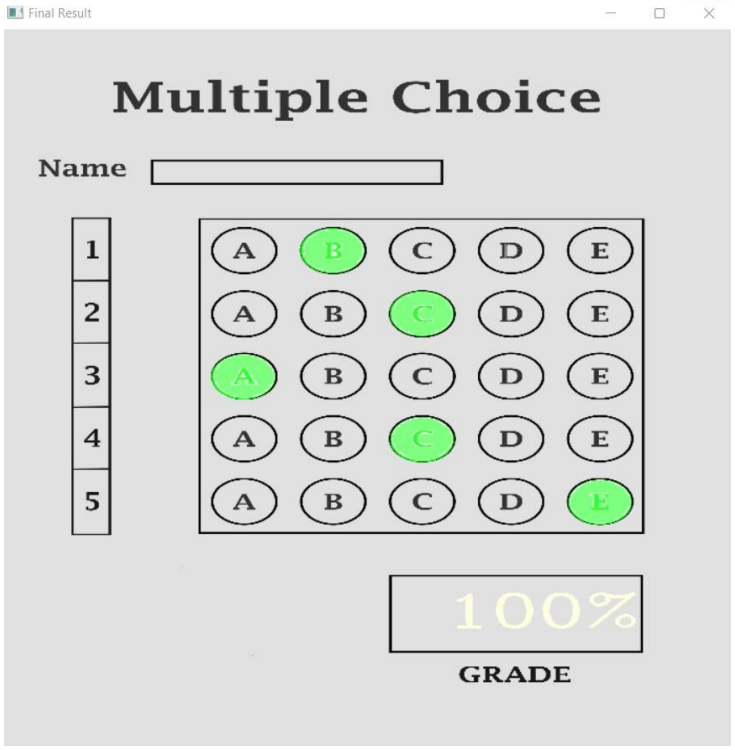
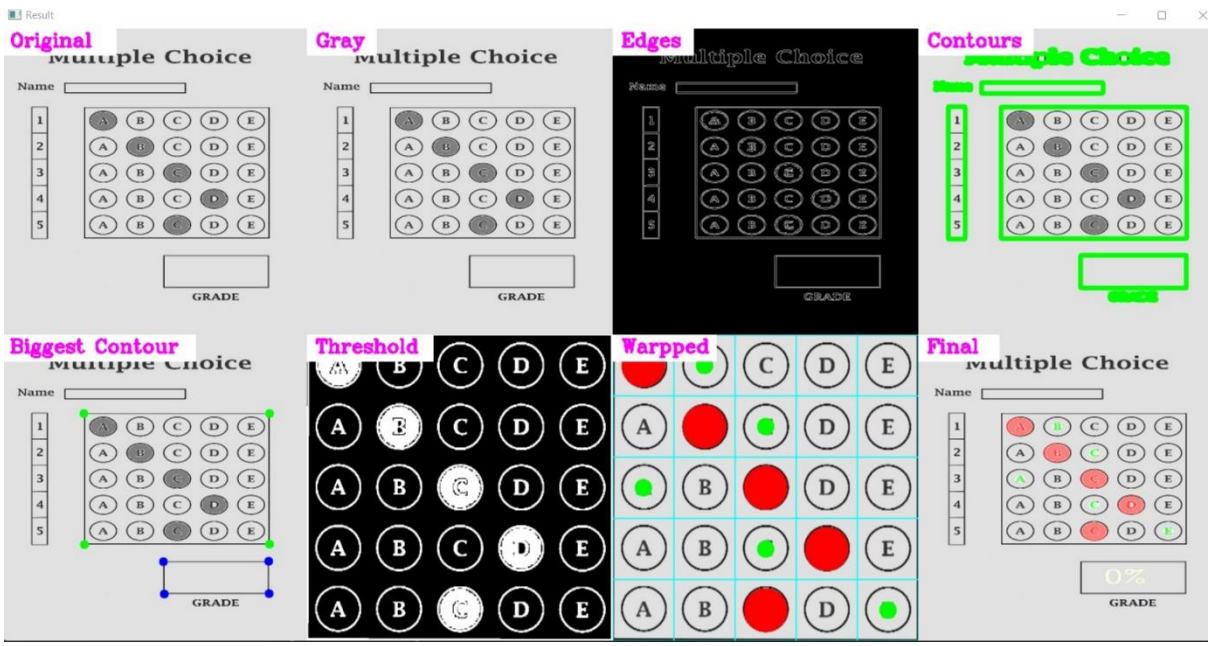




**Fig:** Shows 100% Result

## TESTING 4-



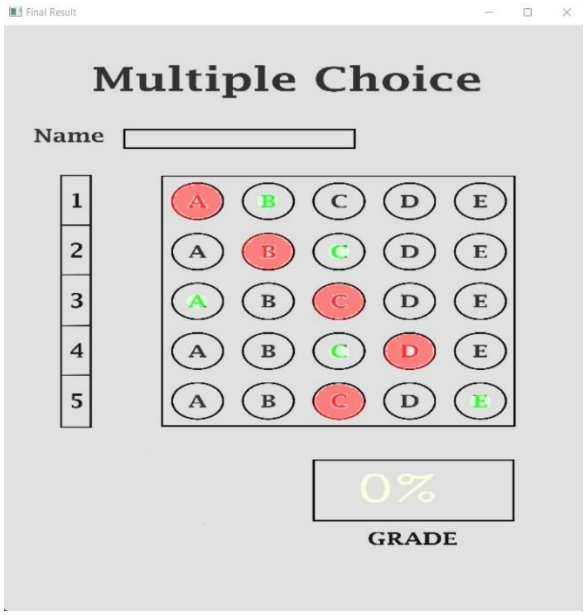
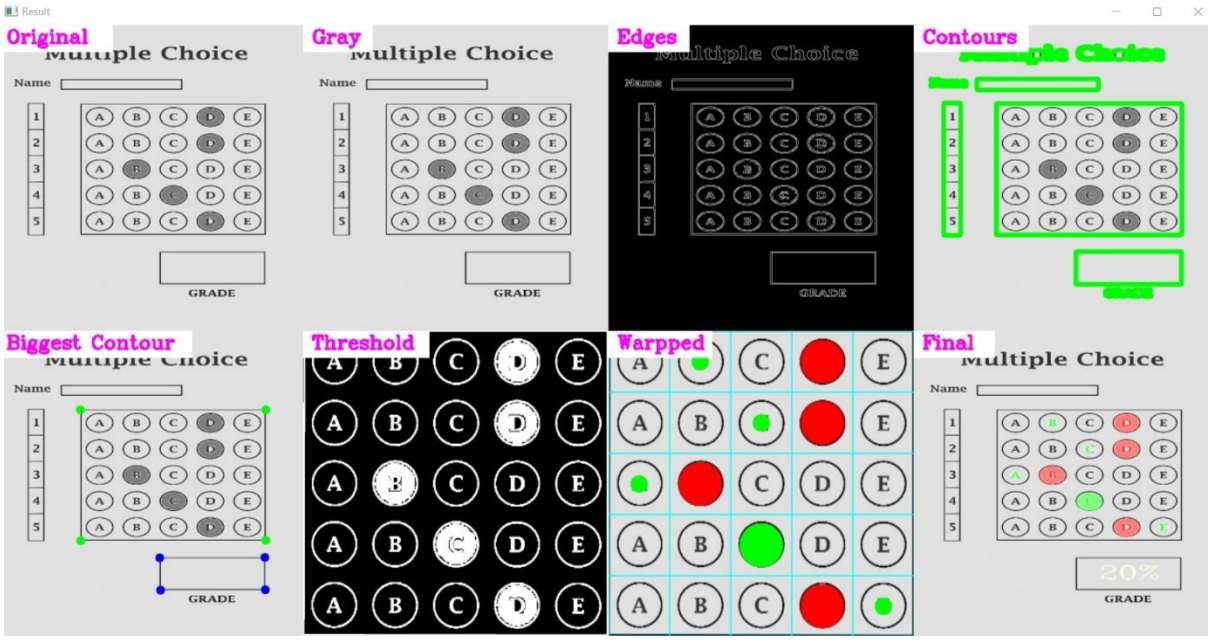




**Fig:** Shows 0(Fail) Result

## TESTING 5-



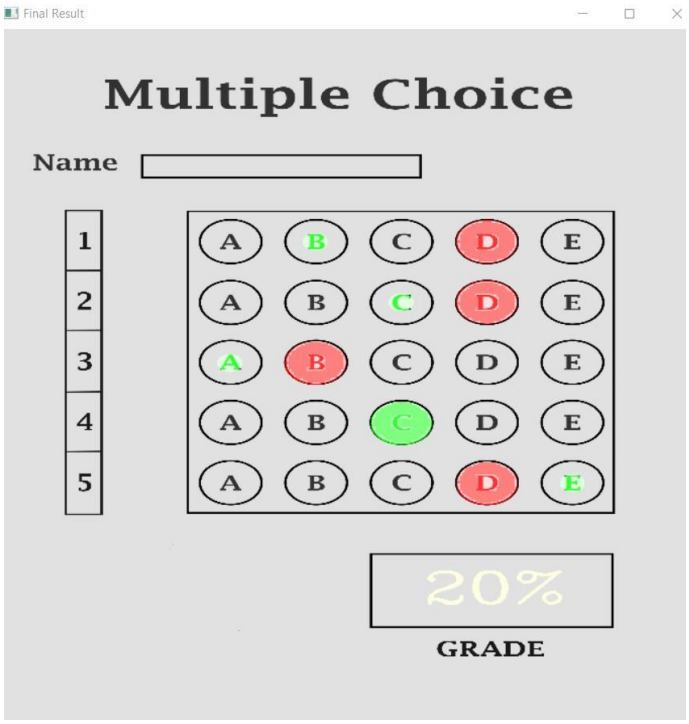
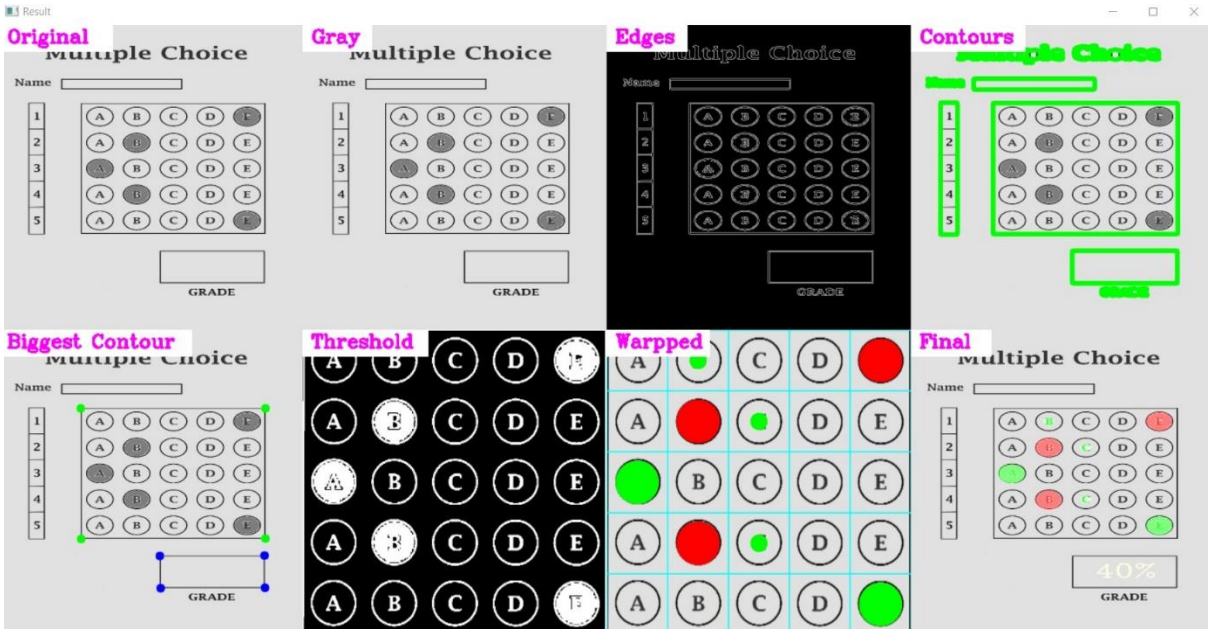




**Fig:** Shows 20% Result

## TESTING 6-



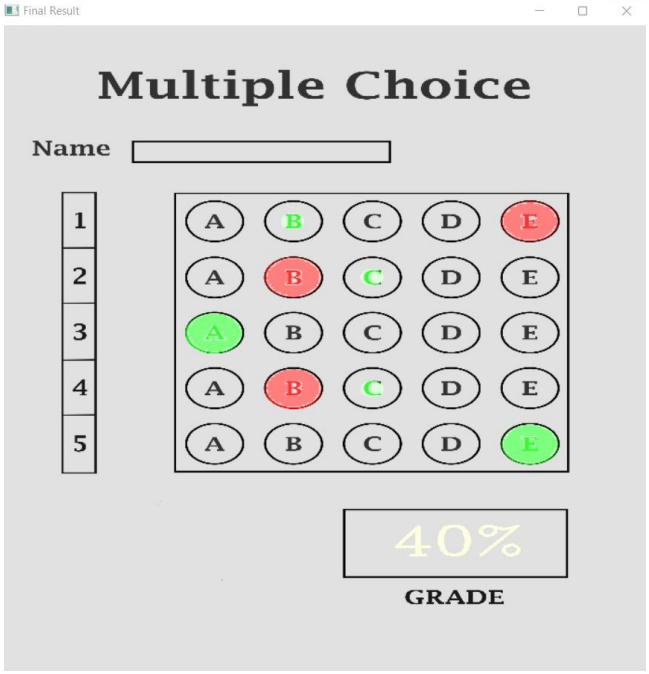
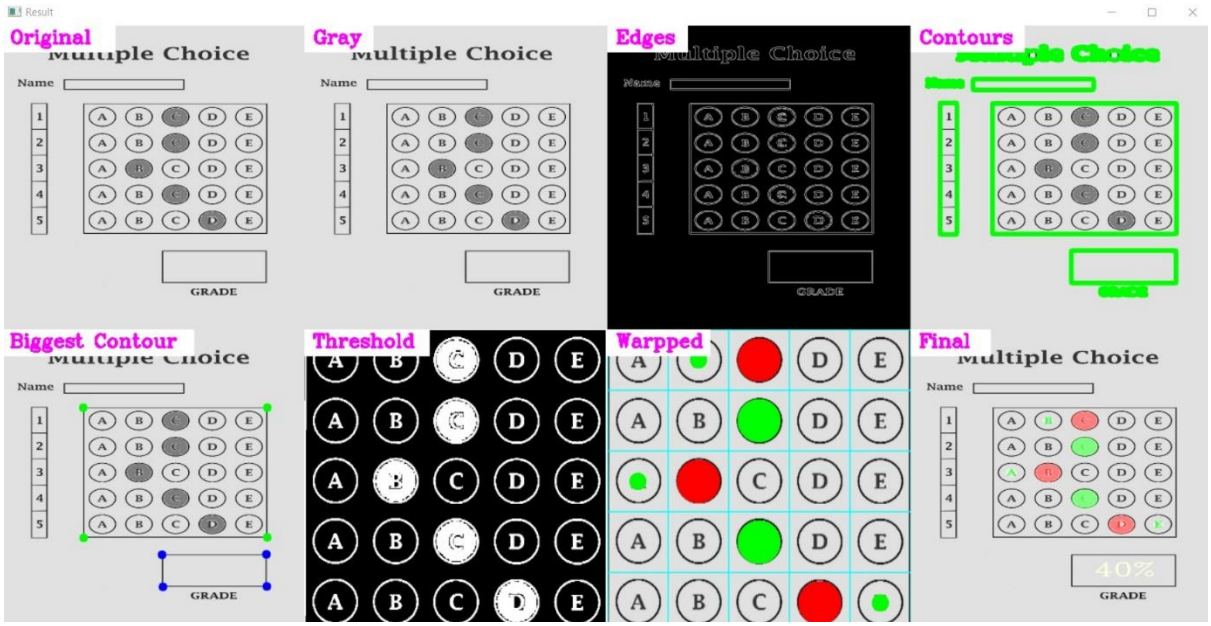




**Fig:** Shows 40% Result

## TESTING 7-



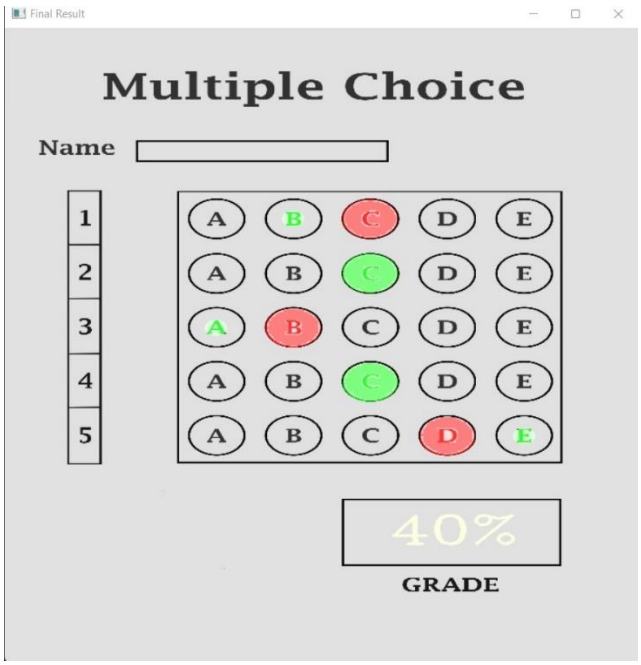
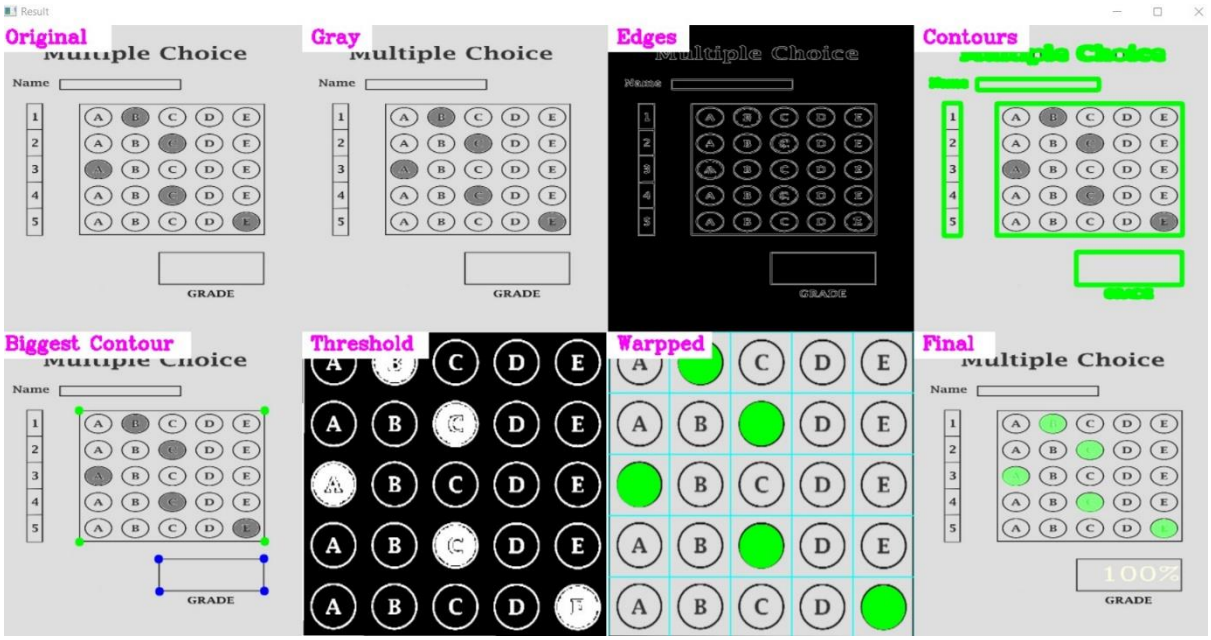




**Fig:** Shows 40% Result

## TESTING 8-



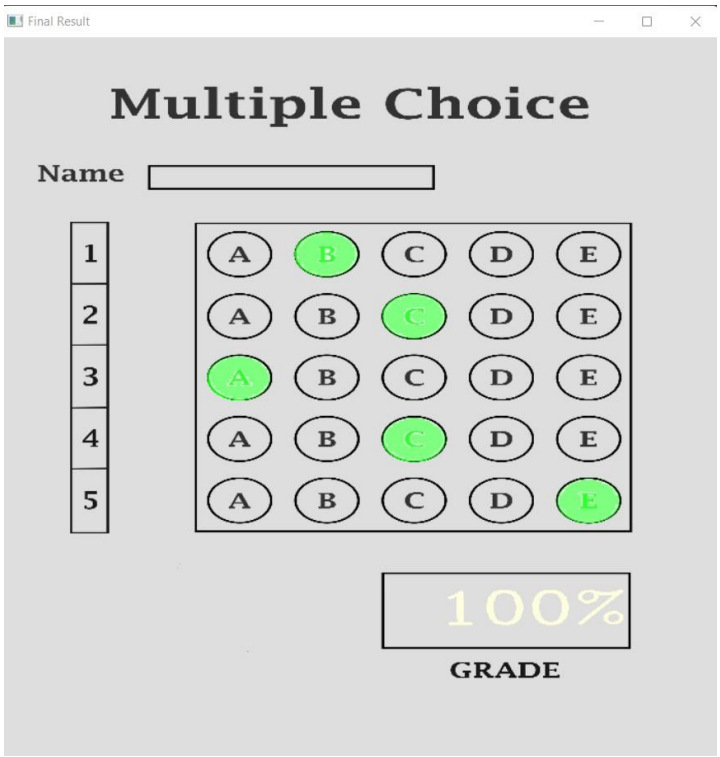
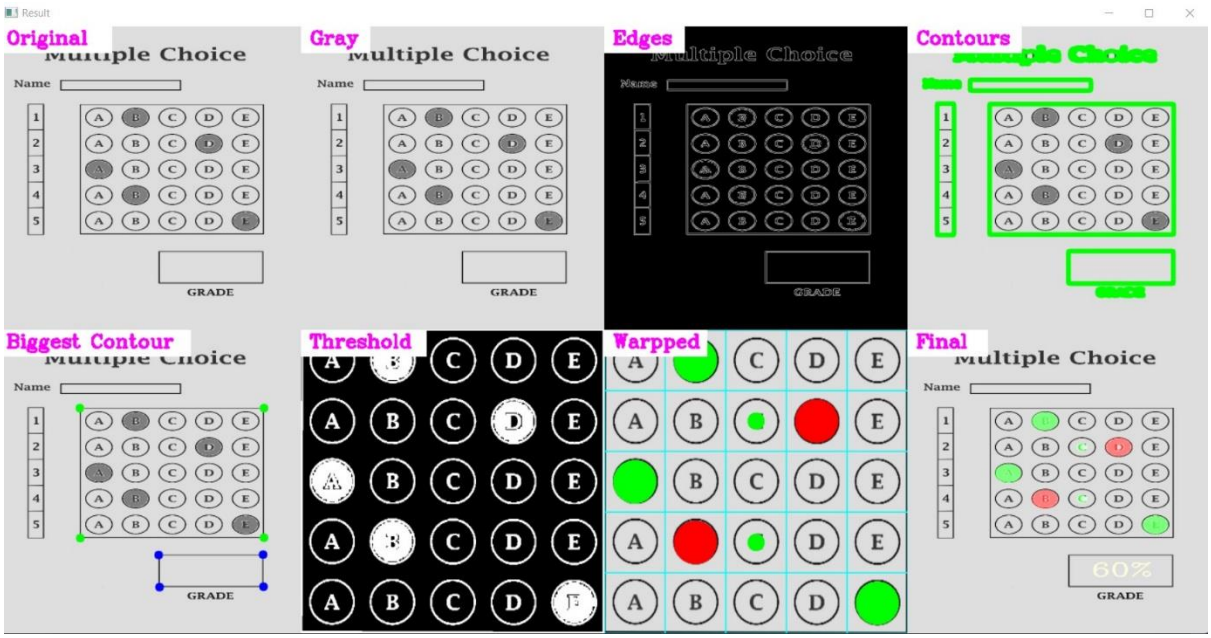




**Fig:** Shows 100% Result

## TESTING 9-



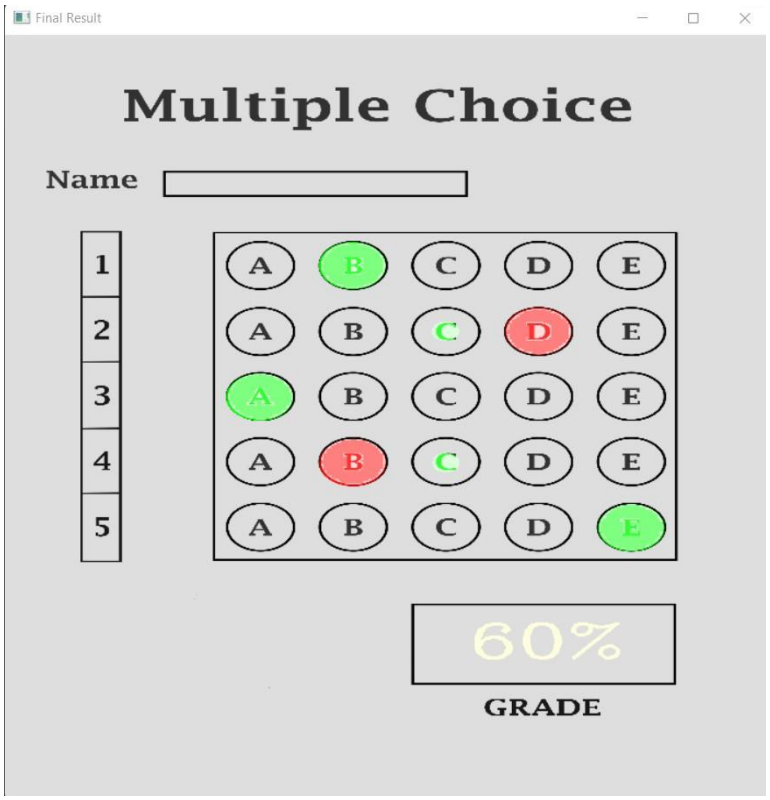
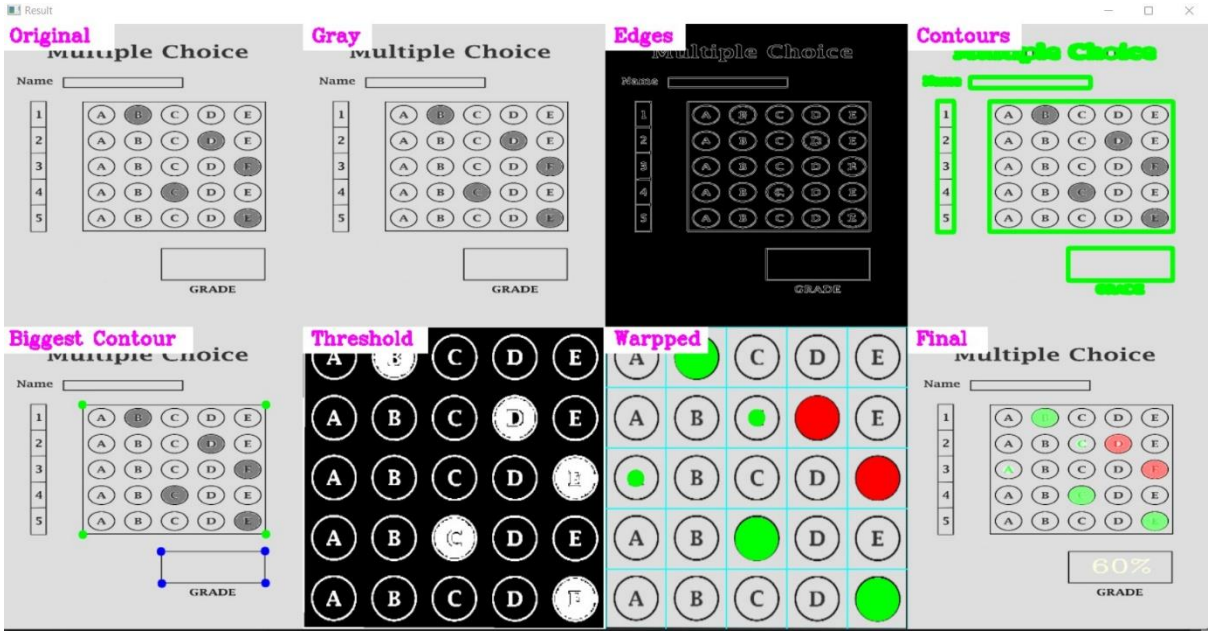




**Fig:** Shows 60% Result

## TESTING 10-



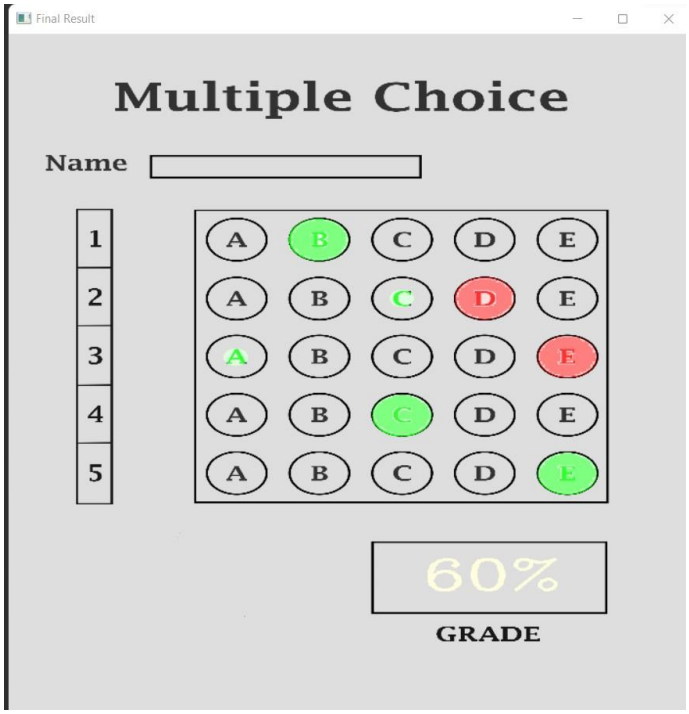
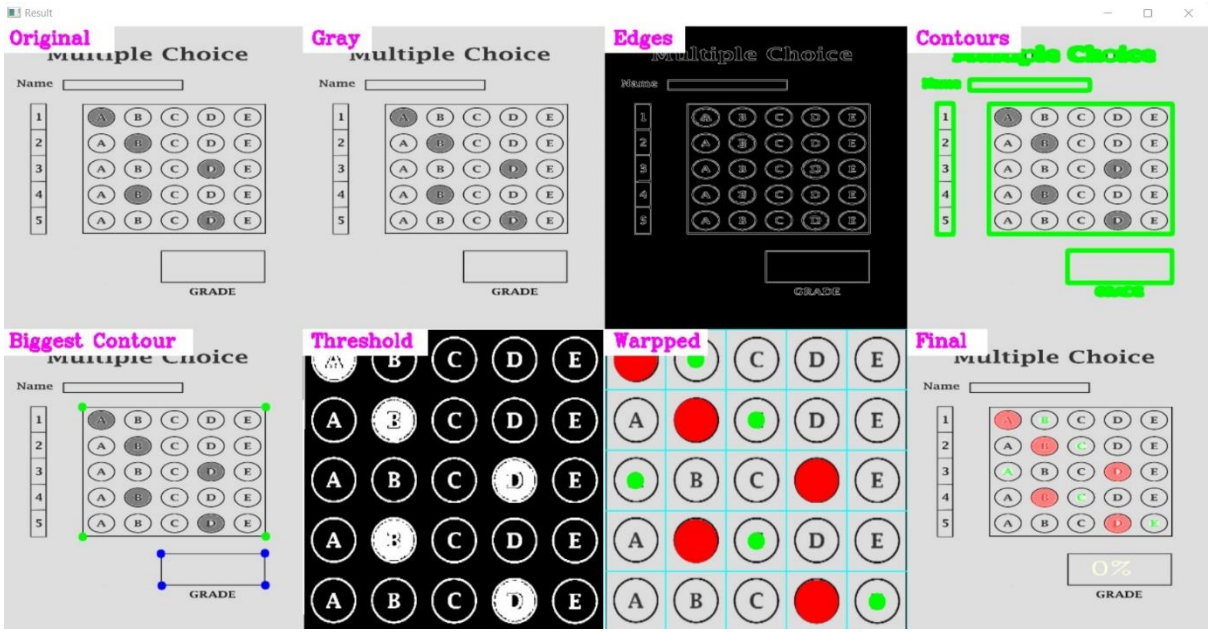




**Fig:** Shows 60% Result

## TESTING 11-



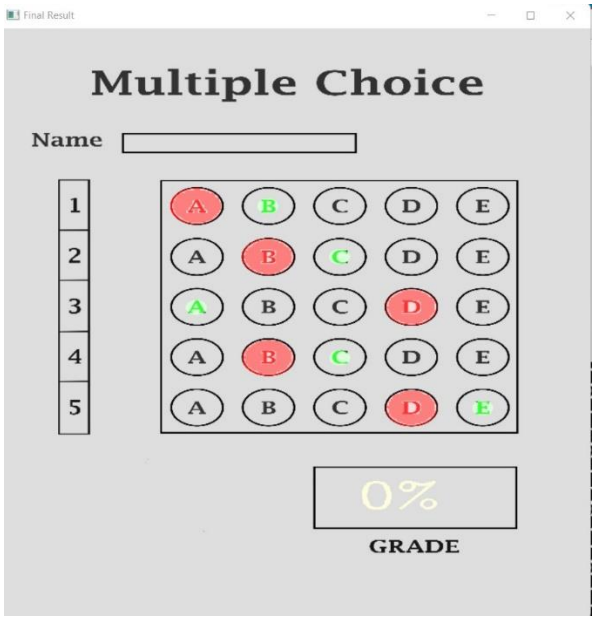
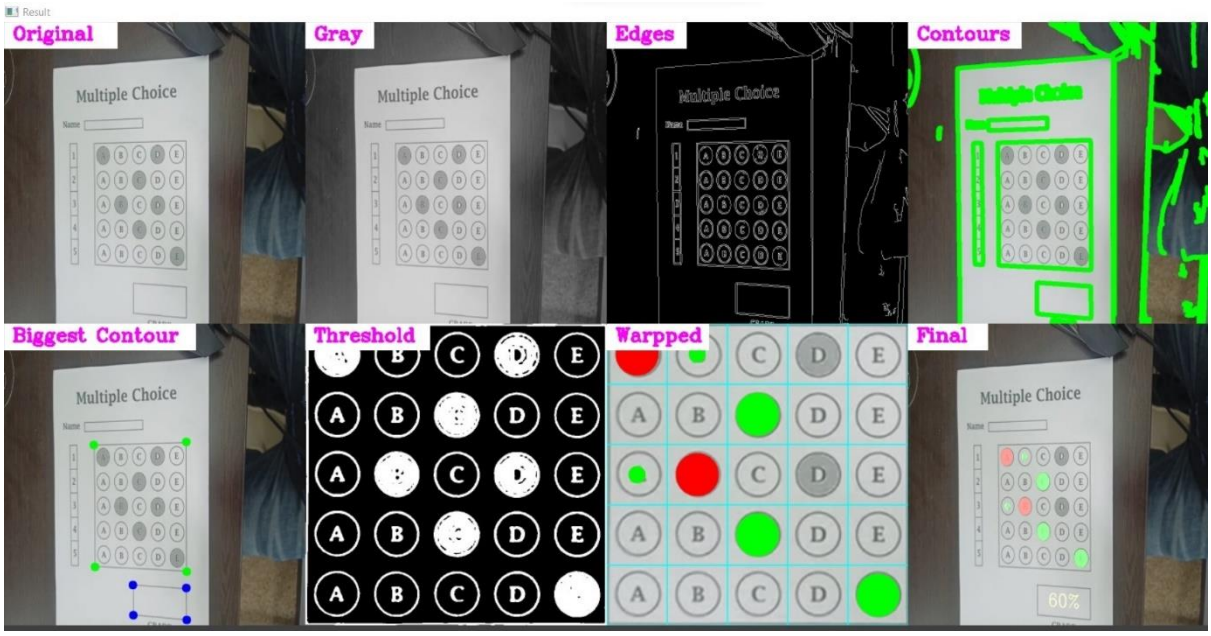




**Fig:** Shows 0% Result

**Part2-**Testing via webcam

## TESTING 1-



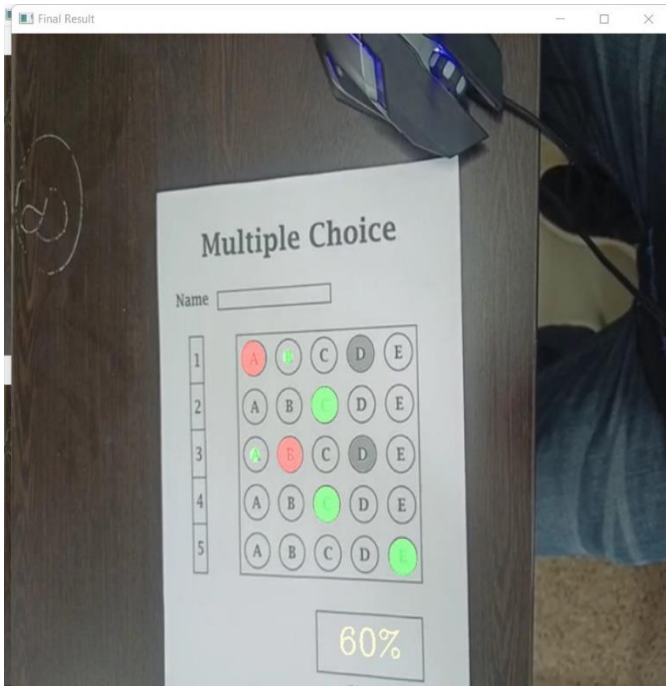
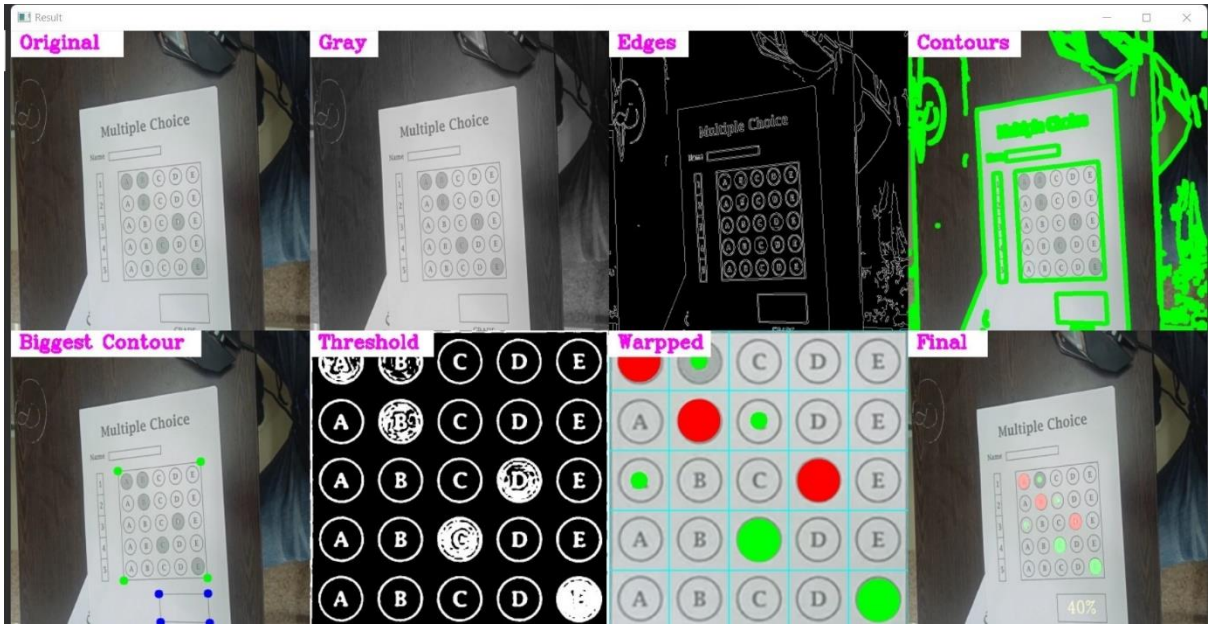




**Fig:** Shows 60% Result

## TESTING 2-



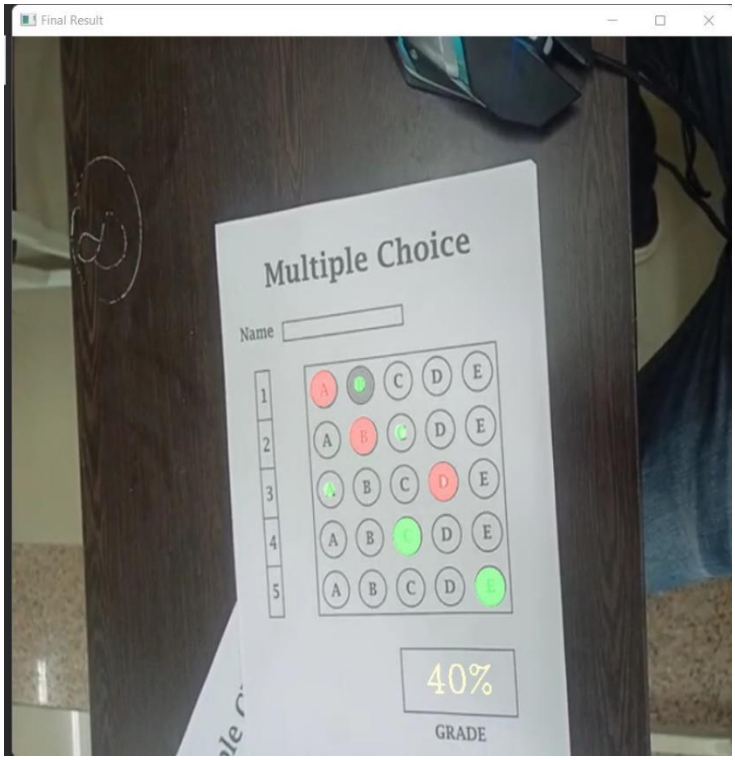
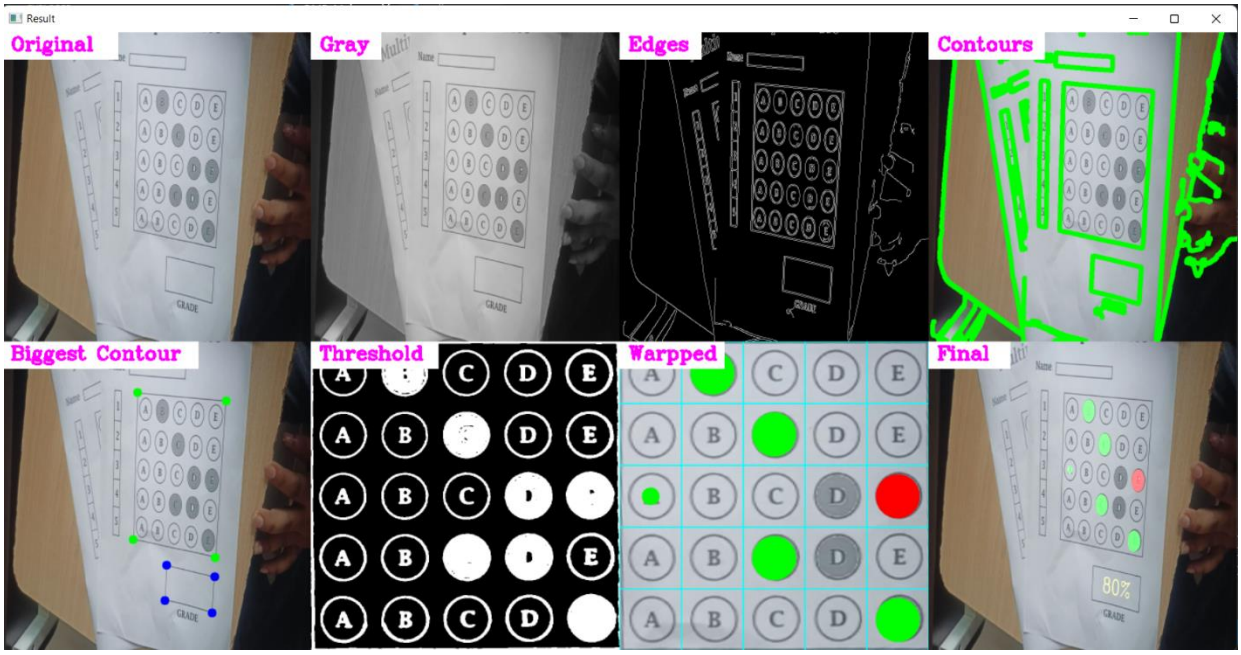




**Fig:** Shows 40% Result

## TESTING 3-



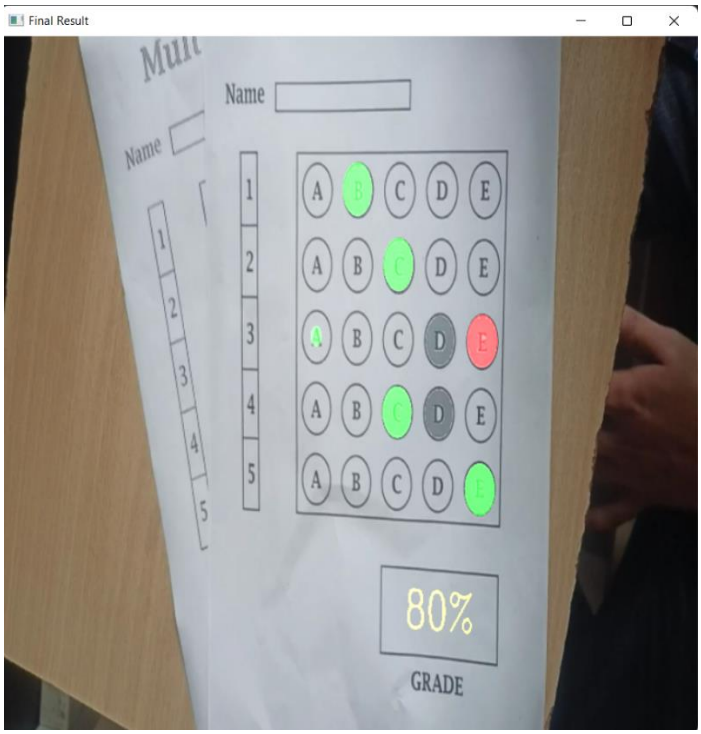
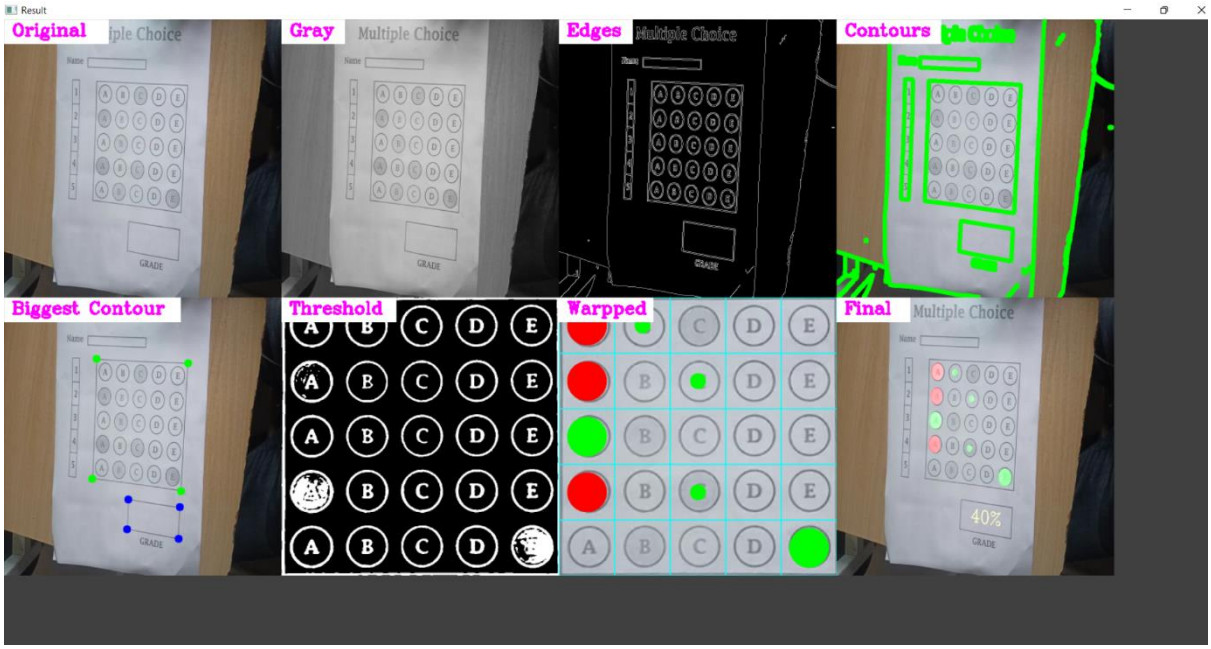




**Fig:** Shows 80% Result

## TESTING 4-



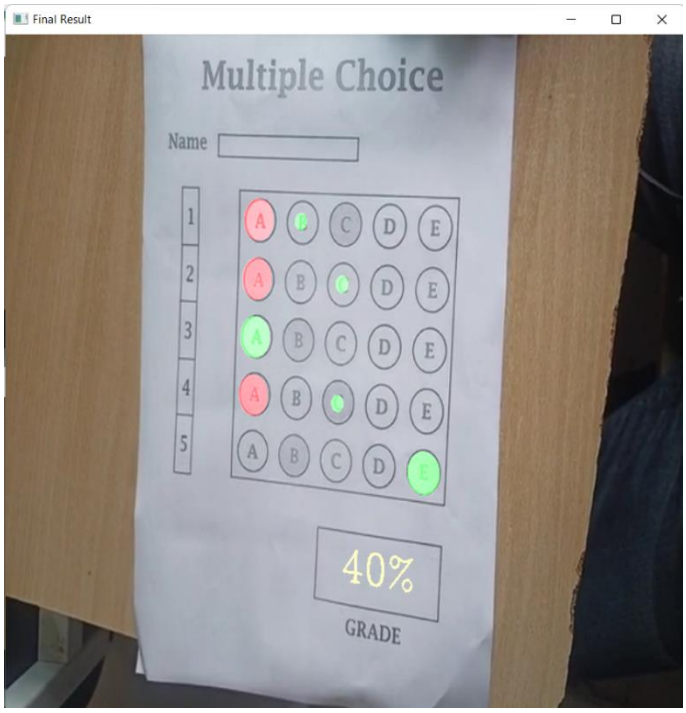
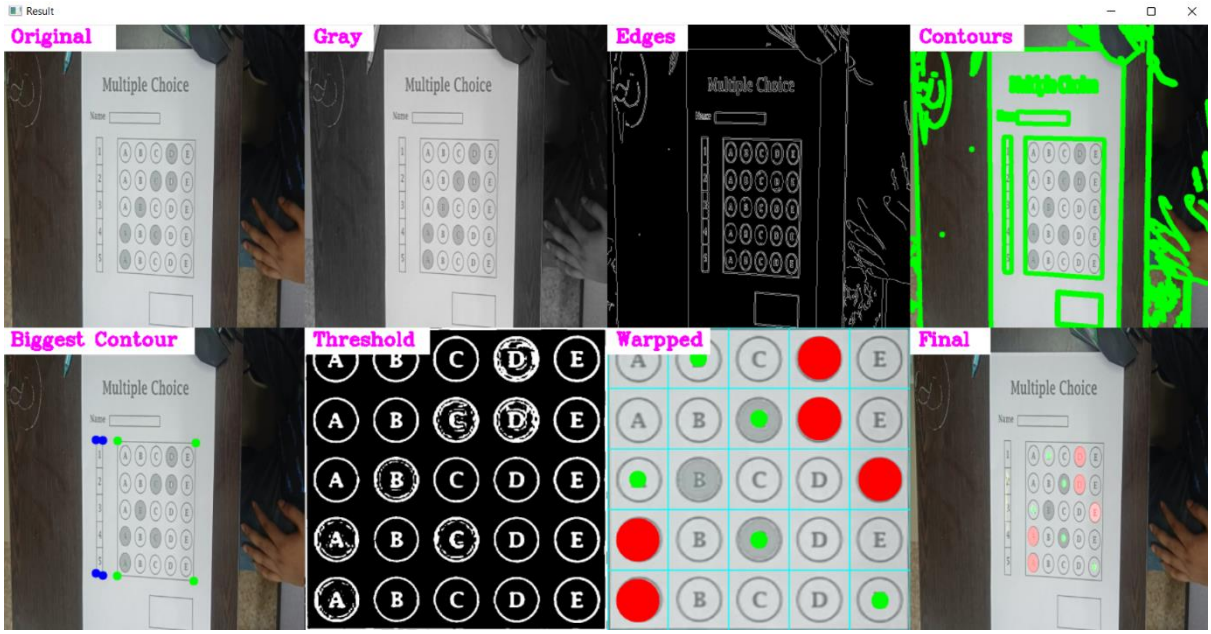




**Fig:** Shows 40% Result

## TESTING 5-



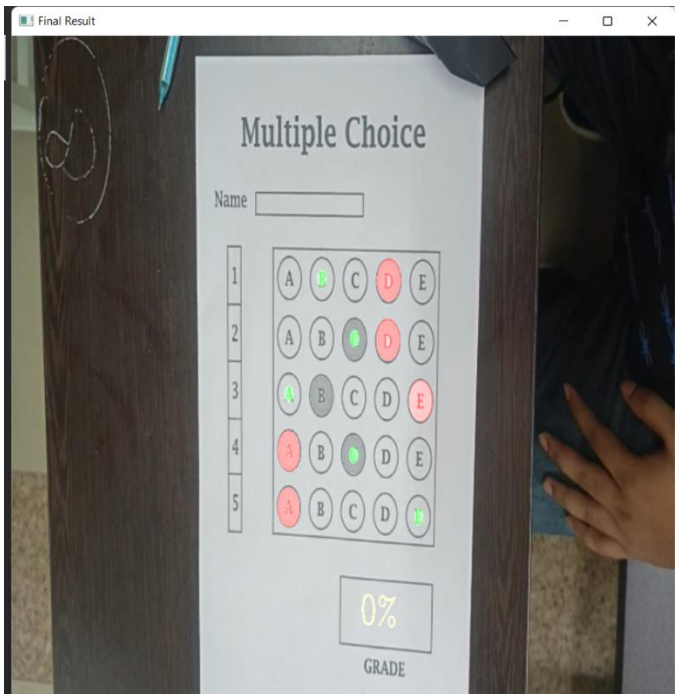
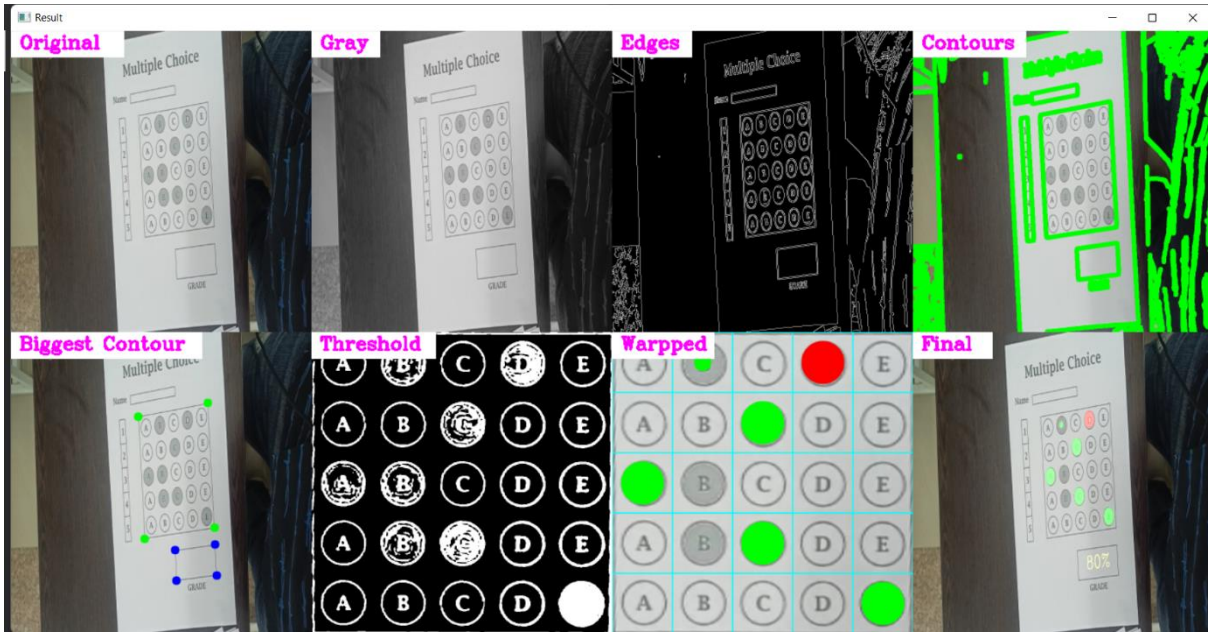




**Fig:** Shows 0% Result

## TESTING 6-



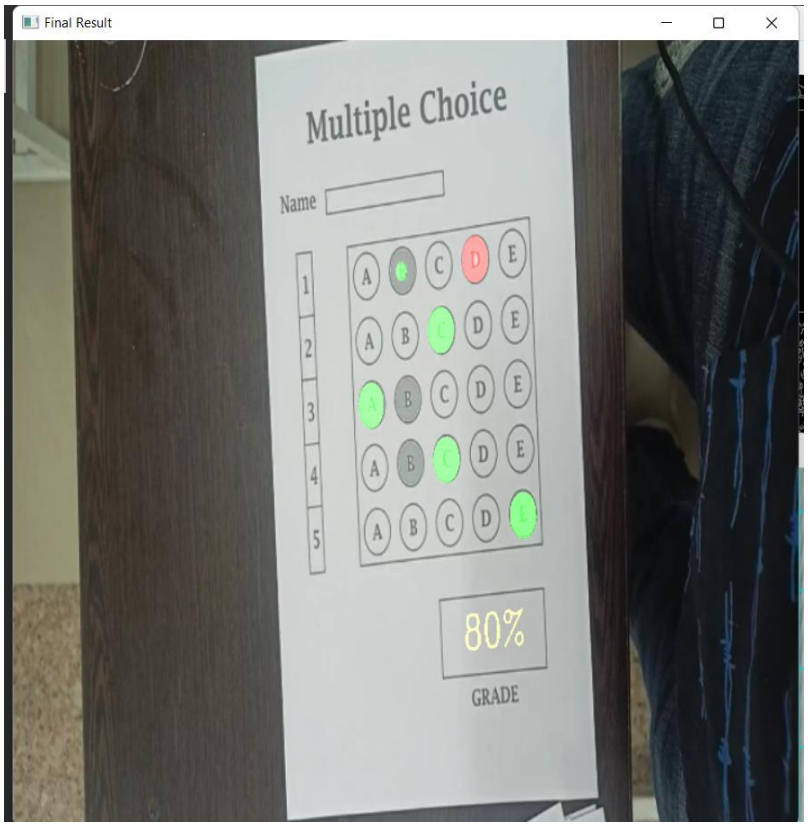
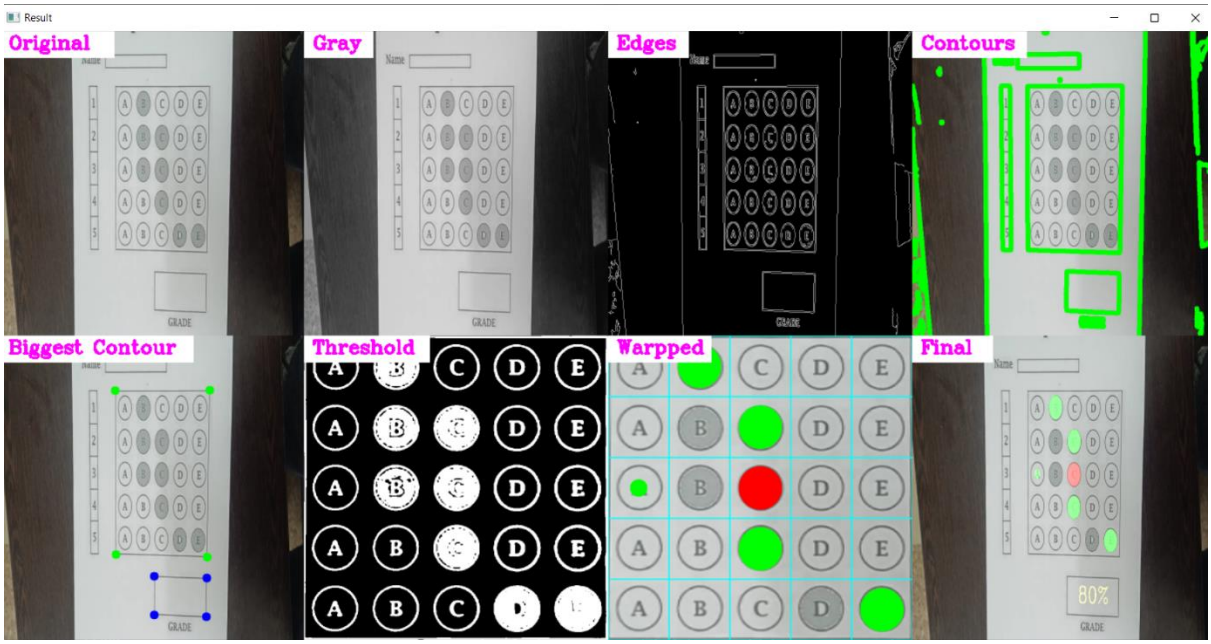




**Fig:** Shows 80% Result

## TESTING 7-



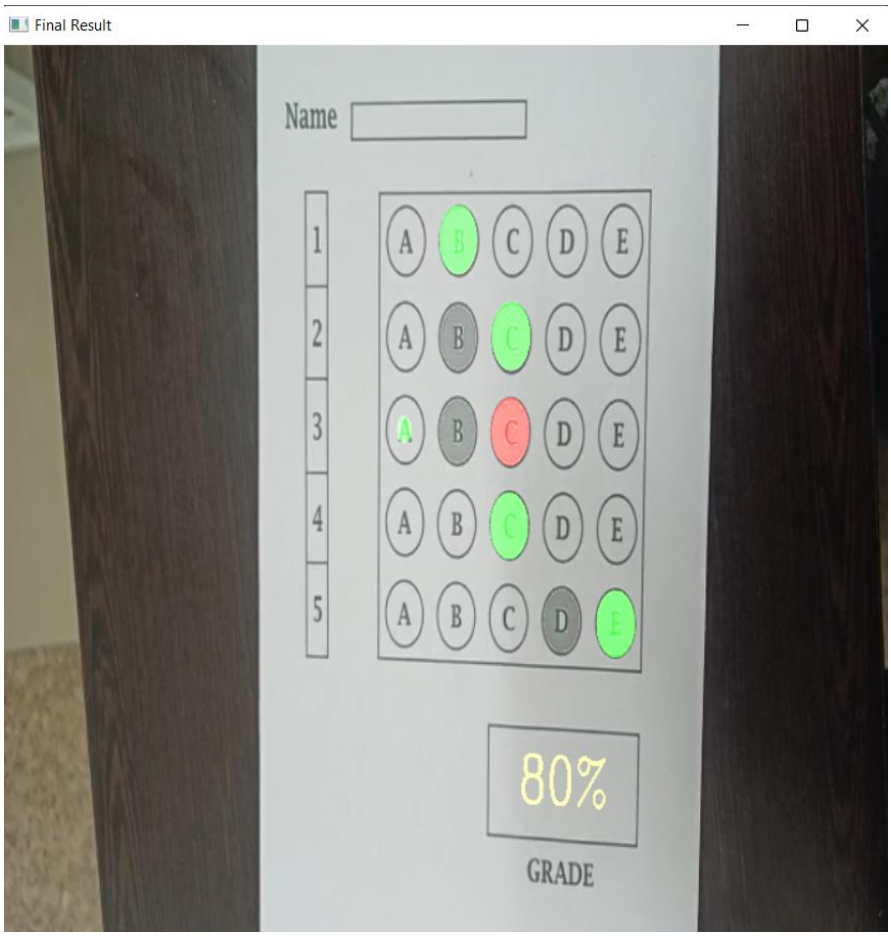
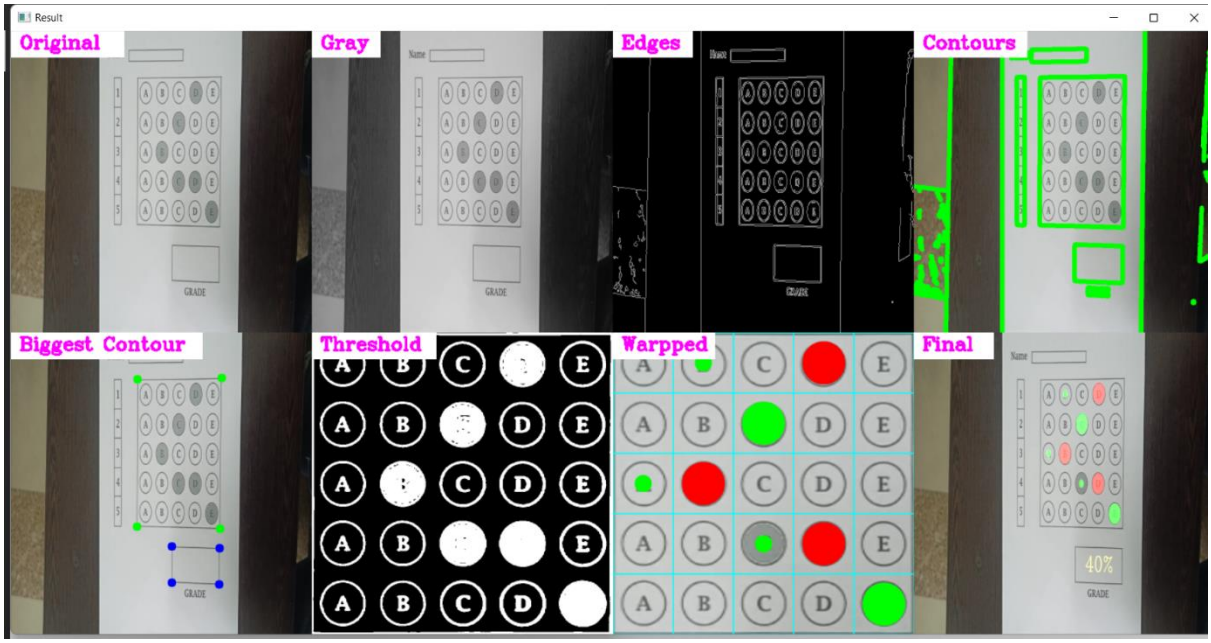




**Fig:** Shows 80% Result

## TESTING 8-



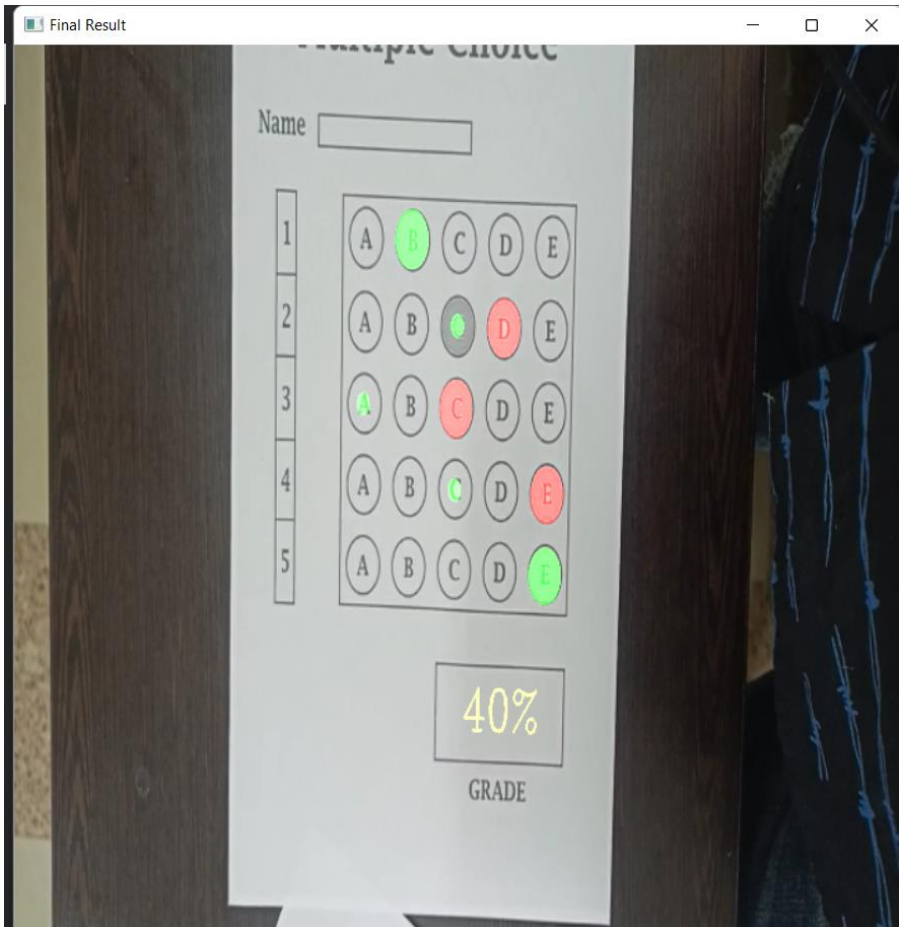
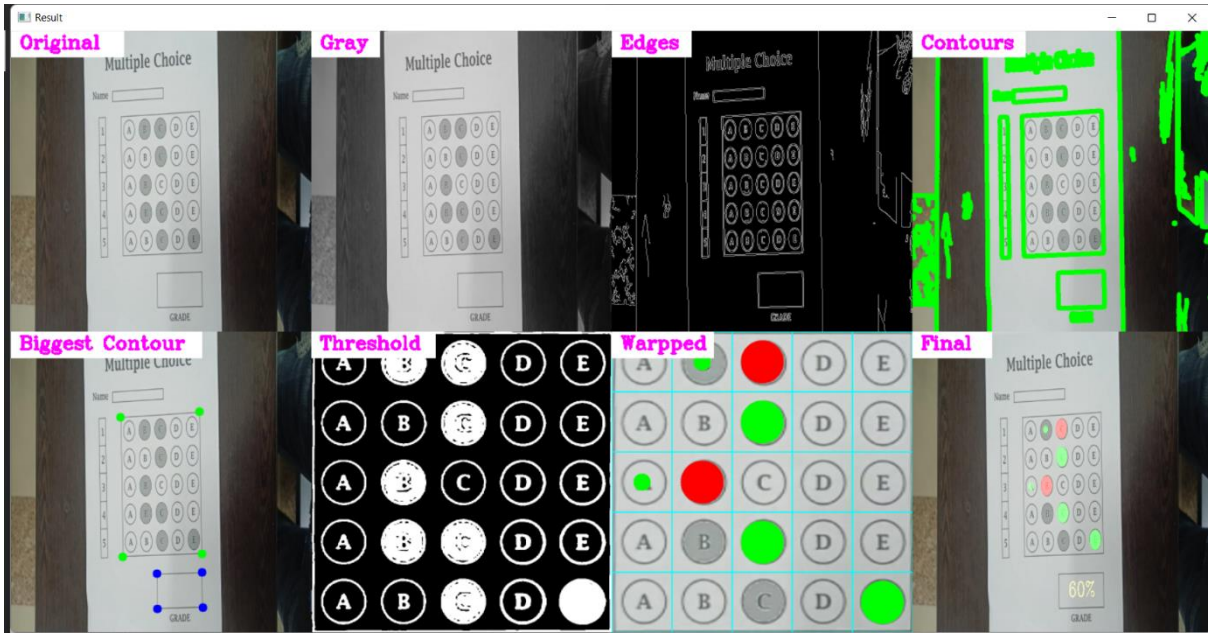




**Fig:** Shows 40% Result

## TESTING 9-



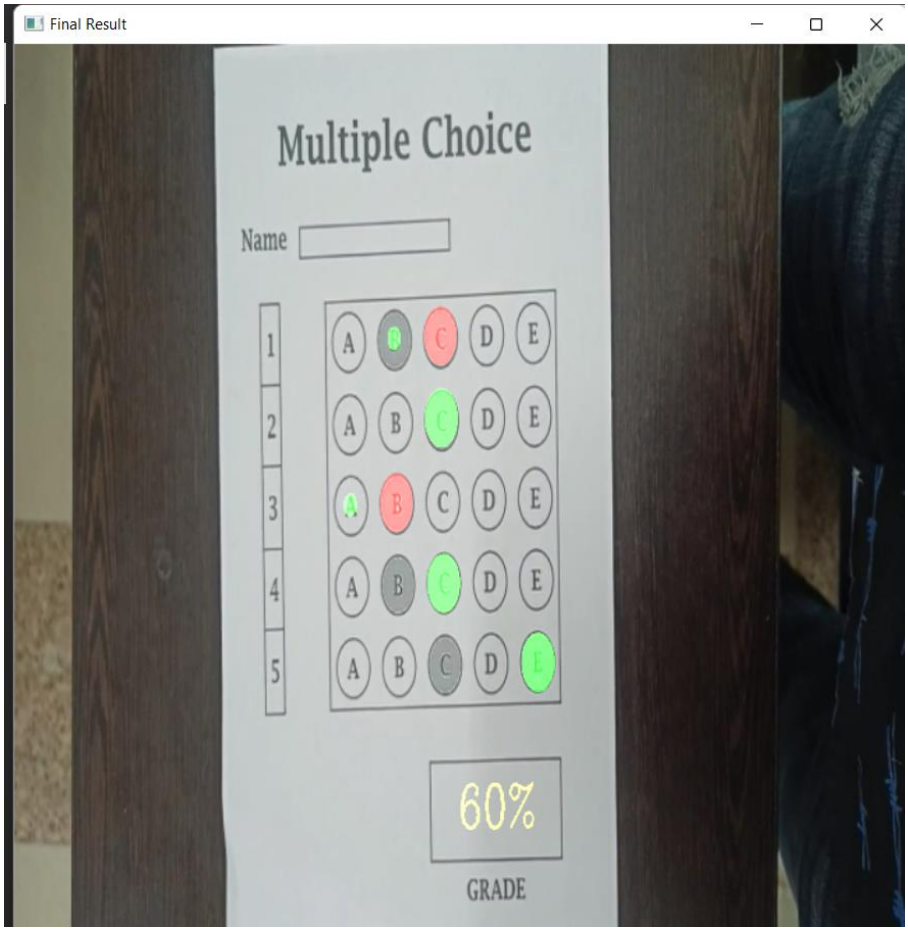




**Fig:** Shows 60% Result

# CHAPTER 8 PROJECT REQUIREMENTS

## SOFTWARE REQUIREMENTS-

1.Anaconda

2.Python 3.8

3.OPEN CV

4.Windows 10 or higher versions

## HARDWARE REQUIREMENTS-

1.Intel i5 or higher versions

2.4GB Ram

3.Webcam

4.25GB storage HDD or SSD

# CHAPTER 9 CONCLUSION & FUTURE SCOPE

## CONCLUSION-

This work presents a system for Optical Mark Recognition developed for multiple choice tests with the programming language Python. OMR scanners were traditionally hardware-focused, but we created one using an underlying model that simply required a collection of photos or live image from webcam. Scanned photos that have been rotated by some angle are also accepted, therefore the need for hardware for perfect alignment is not necessary here. A user interface is also supplied so that persons with less expertise may utilize it as well. The OMR sheet will be in front of the user, and if the picture is scanned incorrectly, it can be skipped and scanned again. Sometimes applicants may not fill out their information correctly or neglect to indicate certain important elements, which are then appropriately recorded in the database. As it is easy to use, the software can be easily used by teachers or school managers as well. Hence, fast and effective model like this will not only enable the personnel to save time but also it will be very cheap. Furthermore, students will be able to learn the results earlier.

## FUTURE SCOPE-

Currently we give the input image from the from webcam or from dataset model processed the input and apply algorithim on the image and we get the gradded omr sheet as output.

In the future, we may give user authentication to the system so that we have data on who has scanned the photographs and can reach out to the concerned person if something has transpired illegally. This allows us to restrict system access to users who have been educated in this program.

## REFERENCES:


1. Martinez, M. E., Ferris, J. J., Kraft, W., & Manning, W. H. (1992). Automated scoring of paper-and-pencil figural responses. *Journal of Educational Technology Systems*, *20*(4), 251-260.
2. Wagstaff, B., Lu, C., & Chen, X. A. (2019, March). Automatic exam grading by a mobile camera: Snap a picture to grade your tests. In Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion (pp. 3-4).
3. Jingyi, T., Hooi, Y. K., & Bin, O. K. (2021, July). Image Processing for Enhanced OMR Answer Matching Precision. In *2021 International Conference on Computer & Information Sciences (ICCOINS)* (pp. 322-327). IEEE.
4. Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial* (Vol. 620). Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.
5. Winston, P. H. (1984). *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc..
6. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362.
7. Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".
8. Rong, W., Li, Z., Zhang, W., & Sun, L. (2014, August). An improved CANNY edge detection algorithm. In *2014 IEEE international conference on mechatronics and automation* (pp. 577-582). IEEE.
9. Weszka, J. S., & Rosenfeld, A. (1978). Threshold evaluation techniques. *IEEE Transactions on systems, man, and cybernetics*, *8*(8), 622-629.
10. Fisteus, J. A., Pardo, A., & García, N. F. (2013). Grading multiple choice exams with low-cost and portable computer-vision techniques. *Journal of Science Education and Technology*, *22*(4), 560-571.
11. Androulidakis, I. I., & Androulidakis, N. I. (2005, February). On a versatile and costless OMR system. In Proceedings of the 4th WSEAS International Conference on Signal Processing, Robotics and Automation (pp. 1-6).