VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI - 590 018, KARNATAKA.

MINI PROJECT REPORT
On

"OMR MCQ AUTOMATED GRADING"

Submitted in the partial fulfillment of requirements

for

COMPUTER GRAPHICS AND IMAGE PROCESSING LABORATORY

(21CSL66)

Submitted by

ANUSHREE BASAVARAJ KUPPAST **4BD21CS020**

Project Guide:

Dr. Naveen H M M. Tech., Ph.D
Associate Professor
Department of CS&E
B.I.E.T., Davanagere

Project Co-Guide:

Prof. Mohamed Muthahir R M. Tech
Assistant Professor
Department of CS&E
B.I.E.T., Davanagere

Department of Computer Science and Engineering.
Bapuji Institute of Engineering & Technology
Davangere- 577004

2024

Bapuji Institute of Engineering and Technology

Davangere -577004

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **ANUSHREE BASAVARAJ KUPPAST** bearing **USN 4BD21CS020** of **Computer Science and Engineering** department have satisfactorily submitted the Mini project report entitled **"OMR MCQ AUTOMATED GRADING"**. The report of the project has been approved as itsatisfies the academic requirements in respect of project work prescribed for the academic year 2024.

Project Guide

Dr. Naveen H M M.Tech., Ph.D
Associate Professor
Department of CS&E
B.I.E.T., Davangere

Project Co-Guide:

Prof. Mohamed Muthahir M. Tech.
Assistant Professor
Department of CS&E
B.I.E.T., Davangere

Head of Department

Dr. Nirmala C.R M. Tech., Ph. D
Dept. Head & Placement Dean
Department of CS&E
B.I.E.T., Davangere

Date:

Place: Davangere

Signature of Examiners:

(1) _____

(2) _____

ACKNOWLEDGEMENT

Salutations to our beloved and highly esteemed institute, **"BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY"** for having well qualified staff and labs furnished with necessary equipments.

We express our sincere thanks to our guide **Dr. Naveen H M** and co-guide**Prof. Mohamed Muthahir** for giving us constant encouragement, support and valuable guidance throughout the course of the project without whose stable guidance this project would not have been achieved.

We express whole hearted gratitude to **Dr. Nirmala C R** who is our respectable HOD of Computer Science & Engineering Department. We wish to acknowledge her help who made our task easy by providing with his valuable help and encouragement.

We also express our whole hearted gratitude to our principal, **Dr. Aravind H B,** for his moralsupport and encouragement.

We would like to extend my gratitude to all staff of **Department of Computer Science and Engineering** for the help and support rendered to me. I have benefited a lot from the feedback, suggestions given by them.

We would like to extend my gratitude to all my family members and friends especially for their advice and moral support.

ANUSHREE BASAVARAJ KUPPAST(4BD21CS020)

KOMAL PRABHUDEV JADIMATH(4BD21CS062)

CONTENTS

ABSTRACT

The OMR (Optical Mark Recognition) Sheet Scanner Computer Graphics Project aims to develop an advanced system for accurately scanning and processing OMR sheets used in examinations and surveys. The project focuses on creating a software application that leverages computer graphics and image processing techniques to detect and interpret marks on OMR sheets. Key components include capturing high-resolution images of the sheets, applying preprocessing steps to enhance image quality, and utilizing algorithms to identify and analyze marked areas. The system will be designed to handle various types of OMR sheets and provide accurate, real-time results. A user-friendly interface will enable easy scanning and result interpretation. This project has significant applications in educational institutions, market research, and data collection processes .

CHAPTER 1: INTRODUCTION

1.1 Introduction to Python

Python Definition

Python is an interpreted, high-level, general-purpose (designed to be used for writing software in the widest variety of application domains) programming language.

1.1.1 History

Python is a high-level programming language that was conceived in the late 1980s and first released in 1991 by Guido van Rossum. Here's a brief history of how Python came to be and evolved over the years:

Origin: Python's development began in December 1989 when Guido van Rossum, then working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands, started work on a successor to the ABC programming language.

First Release: The first public release, Python 0.9.0, came in February 1991. It included features like exception handling, functions, and the core data types of list, dict, str, and more.

Python 2 Era:

Python 2.x: Following the initial release, Python 2.0 was released in 2000, introducing features like garbage collection and support for Unicode. The Python 2 series continued to evolve with incremental updates, with Python 2.7 being the final release of the 2.x branch, supported until January 1, 2020.

Python 3 Transition:

Python 3.x: Python 3.0, released in 2008, marked a significant overhaul aimed at addressing fundamental design flaws and inconsistencies in the language. It introduced backward in compatible changes to improve consistency, simplicity, and future maintainability.

Adoption Challenges: The transition from Python 2 to Python 3 faced challenges due to compatibility issues, which led to a period where both versions coexisted in production environments.

Recent Developments:

Python 3.x Series: Since Python 3.0, the language has continued to evolve with regular updates, improving performance, adding new features, and refining existing ones. Notable releases include Python 3.5 (2015), Python 3.6 (2016), Python 3.7 (2018), Python 3.8 (2019), Python 3.9 (2020), and Python 3.10 (2021).

Community and Ecosystem Growth: Python's popularity has grown significantly over the years, supported by a vibrant community contributing libraries, frameworks, and tools across various domains such as web development, data science, machine learning, and more.

1.1.2 Features of python

1. **Easy to Learn and Use:**

Python's syntax is designed to be intuitive and readable, making it easy for beginners to grasp. It emphasizes code readability and allows developers to express concepts in fewer lines of code compared to other languages.

2. **Expressive Language**:

Python enables developers to write clear and logical code for both small and large-scale projects.It supports multiple programming paradigms including procedural, object-oriented, and functional programming.

3. **Interpreted and Interactive:**

Python is an interpreted language, meaning code is executed line by line, which allows for rapid prototyping and debugging. It also supports an interactive mode where you can test code snippets and get immediate feedback.

4. Large Standard Library:

Python comes with a broad range of modules and libraries, known as the Python Standard Library,which provides ready-to-use tools for tasks such as file I/O, string manipulation, operating systeminterfaces, and more. This reduces the need for external libraries for many common tasks.

5. Third-Party Libraries and Frameworks:

In addition to the standard library, Python has a rich ecosystem of third-party libraries and frameworks that extend its functionality. Examples include Django and Flask for web development, NumPy and Pandas for data analysis, TensorFlow and PyTorch for machine learning,and many more**.**

6. Open Source and Community-driven:

Python is developed under an open-source license, fostering a supportive and active community. This community contributes to the language's growth by creating libraries, frameworks, and tools,and providing support through forums, conferences, and online resources**.**

7. Portability:

Python runs on various platforms including Windows, macOS, Linux, and Unix. This portability makes it suitable for developing applications that can run on different systems without modification.

8. Integration Capabilities:

Python can integrate with other languages such as C, C++, and Java via modules and libraries, allowing developers to leverage existing code and tools.

9. Scalability:

While Python is often used for small to medium-sized projects, it is also capable of scaling up to handle complex applications. This scalability is supported by its robust standard library, thirdparty frameworks, and tools for profiling and optimizing code.

10. Versatility:

Python is used in various domains including web development, scientific computing, data analysis, artificial intelligence, machine learning, automation, scripting, and more. Its versatility and ease of use make it a preferred choice for diverse applications.

1.2 Introduction to OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library for computer vision and image processing. Originally developed by Intel, it is now maintained by the OpenCV community. It provides a comprehensive set of tools and functions for handling and manipulating images and performing various computer vision tasks.

1.2.1 Features of OpenCV:

1. **Cross-platform:** OpenCV is compatible with various operating systems including Windows, Linux, macOS, Android, and iOS. This makes it highly versatile for developing applications across different platforms.
2. **Extensive Functionality:** OpenCV offers a vast array of functions and algorithms for performing tasks such as image and video processing, feature detection, object recognition, machine learning, camera calibration, and more.
3. **Support for Multiple Languages:** While OpenCV is primarily implemented in C++, it also provides bindings for Python, Java, and MATLAB, making it accessible to developers using different programming languages.
4. **Efficiency:** OpenCV is optimized for real-time applications and is known for its high computational efficiency. It leverages hardware acceleration (like SIMD instructions) and supports multi-threading to maximize performance.
5. **Community and Development**: OpenCV is maintained by a large community of developers and researchers worldwide. Contributions from the community ensure continuous improvement, bug fixes, and the addition of new features.
6. **Modules and Libraries:** OpenCV is modularized into different libraries and modules, each focusing on specific tasks or functionalities. Some key modules include.

- **Core:** Basic data structures, matrix operations, and fundamental functions.
- **Imgproc:** Image processing operations such as filtering, edge detection, and morphological operations.
- **Video:** Video analysis, optical flow, background subtraction.
- **Features2D:** Feature detection (e.g., keypoints) and description algorithms (e.g., SIFT, SURF, ORB).
- **Objdetect:** Object detection using Haar cascades, HOG (Histogram of Oriented Gradients), etc.
- **Machine Learning:** Algorithms for machine learning tasks like clustering, classification, and regression.
- **Calib3d:** Camera calibration and 3D reconstruction.

1.2.2 Applications of OpenCV:

- **Computer Vision:** OpenCV is widely used for tasks such as face detection and recognition, object tracking, gesture recognition, motion analysis, and augmented reality.
- **Image Processing:** It provides numerous functions for basic to advanced image processing tasks like image filtering, transformation, enhancement, and segmentation.
- **Machine Learning:** OpenCV integrates with machine learning libraries like TensorFlow and PyTorch for tasks such as image classification, object detection, and image generation.
- **Robotics:** OpenCV plays a crucial role in robotics applications, including autonomous navigation, object manipulation, and environment mapping.
- **Medical Imaging:** It is utilized for tasks such as medical image analysis, pathology detection, and diagnostic assistance.
- **Industrial Applications:** OpenCV is applied in industrial automation for quality control, defect detection, object sorting, and process monitoring.

1.2.3 OpenCV Packages

OpenCV (Open Source Computer Vision Library) provides a wide range of modules and packages that cater to different aspects of computer vision, image processing, and machine learning tasks. These packages encapsulate various functionalities and algorithms, making OpenCV a versatile tool for developing vision-based applications. Here are some key packages/modules provided by OpenCV:

1. **Core (cv2 module):**
   - **cv2.imread(), cv2.imwrite():** Functions for reading and writing images. ○ **cv2.VideoCapture():** Interface for capturing video from cameras or files.
   - **cv2.VideoCapture() and cv2.VideoWriter():** Classes for handling video streams and writing videos.
2. **Image Processing**

   Imgproc (cv2 module):
   - **cv2.cvtColor():** Convert between color spaces (e.g., BGR to grayscale).
   - **cv2.GaussianBlur(), cv2.medianBlur():** Apply Gaussian or median blurring to images.
   - **cv2.Canny():** Perform Canny edge detection. ○ **cv2.threshold():** Apply thresholding to an image. ○ **cv2.resize(), cv2.rotate(), cv2.flip():** Resize, rotate, and flip images.
   - **cv2.warpAffine(), cv2.warpPerspective():** Perform affine and perspective transformations.
   - **cv2.matchTemplate():** Template matching for finding instances of a template image within a larger image.
3. **Feature Detection and Description**

   Features2D (cv2 module):
   - **cv2.SIFT_create(), cv2.SURF_create(), cv2.ORB_create():** Create instances of feature detection algorithms like SIFT, SURF, and ORB.
   - **cv2.drawKeypoints(), cv2.drawMatches():** Draw key points or matches between two images.
4. **Python Bindings and Utilities**

   Python (cv2 module):
   - **cv2.waitKey(), cv2.destroyAllWindows():** Manage user interface events like key presses and window destruction.
   - **cv2.imwrite(), cv2.imshow():** Display and save images.

CHAPTER 2 :SYSTEM REQUIREMENTS

2.1 Software Requirements

1. Operating System : Microsoft Windows 11
2. Compiler used: PyCharm
3. Language used: Python

2.2 Hardware Requirements

1. Main Processor: intel core i5
2. Processor Speed: 3 GHz or more
3. RAM Size: 4GB

CHAPTER 3 : SYSTEM DESIGN

3.1. Methodology

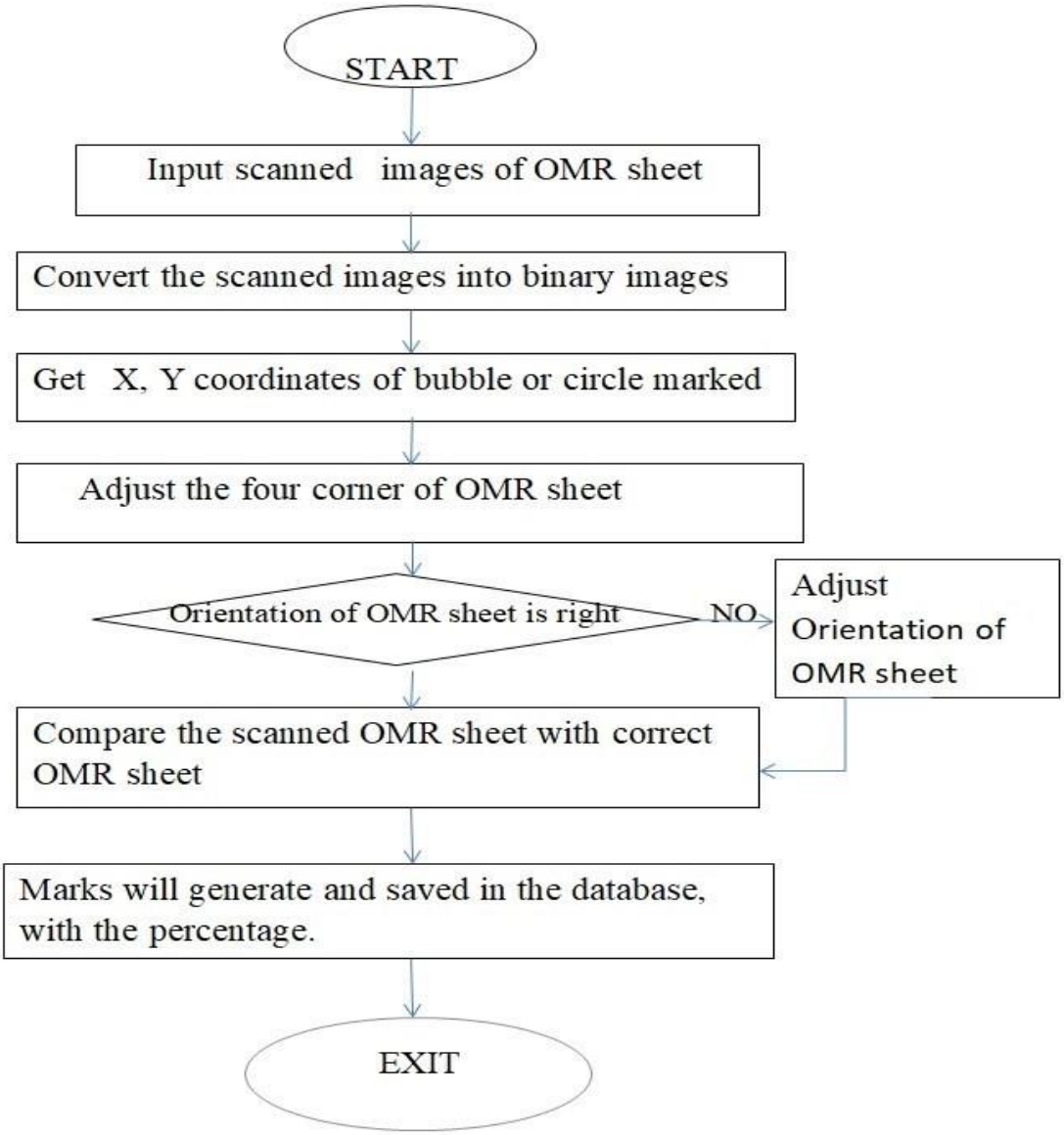

Fig 3: Methodology diagram for OMR MCQ Automated Grading

The methodology for this project involves several key steps:

1. **Image Acquisition**:
    - **Capture**: Obtain images of the OMR sheets using a scanner or high-resolution camera.
    - **Preprocessing**: Ensure images are clear, aligned, and have consistent lighting.
2. **Image Preprocessing**:
    - **Grayscale Conversion**: Convert the image to grayscale to simplify processing.
    - **Noise Reduction**: Apply filters (e.g., Gaussian blur) to reduce noise.
    - **Binarization**: Convert the grayscale image to a binary image using thresholding techniques (e.g., Otsu's method).
    - **Skew Correction**: Detect and correct any skew in the image to ensure alignment.
3. **Region of Interest (ROI) Detection**:
    - **Sheet Layout Identification**: Identify and extract the relevant areas of the sheet where answers are marked.
    - **Mark Detection Zones**: Define and extract zones where marks are expected (e.g., multiple-choice options).
4. **Mark Detection**:
    - **Segmentation**: Divide the image into smaller regions corresponding to individual answer choices.
    - **Mark Identification**: Use image processing techniques to detect marks (e.g., blob detection, contour analysis).
5. **Data Extraction**:
    - **Feature Extraction**: Extract features from detected marks (e.g., position, intensity).
    - **Mark Interpretation**: Translate detected marks into readable data (e.g., converting positions into answers).
6. **Post-Processing**:
    - **Error Correction**: Apply algorithms to correct common errors or inconsistencies (e.g., multiple marks in a single choice).
    - **Validation**: Validate the extracted data to ensure accuracy (e.g., cross-check with predefined answer keys).

7. **Output Generation**:
    - **Data Formatting**: Format the extracted data for output (e.g., generate a report or save in a database).
    - **User Interface**: Optionally, create a user interface for reviewing and managing results.
8. **Testing and Evaluation**:
    - **Accuracy Assessment**: Evaluate the accuracy of the mark detection and data extraction processes.
    - **Performance Metrics**: Measure performance using metrics like precision, recall, and F1 score.
9. **Documentation and Reporting**:
    - **Document Process**: Create detailed documentation of the methodology, challenges, and solutions.
    - **Final Report**: Prepare a final report summarizing the project outcomes and findings.

CHAPTER 4 : IMPLEMENTATION

4.1 Installation Steps

To install and run the project:

1. **Prerequisites:**
   - Python environment with necessary libraries: opencv-python, numpy, utlis
   - Ensure web browser module is available (standard in Python).

2. **Steps:**
   - Image Acquisition: Capture high-quality images of the answer sheets using a camera or scanner.
   - Preprocessing: Gray scale Conversion: Convert the image to gray scale to simplify processing.
   - Gaussian Blur: Apply a Gaussian blur to reduce noise.
   - o Thresholding :Apply a binary threshold to create a black and white image, highlighting the edges and contours.
   - Contour Detection: Detect contours to identify the regions of interest (e.g., answer circles).
   - Feature Detection: Use methods like the Hough Circle Transform to detect circles (bubbles).
   - Answer Identification: Determine the filled circles by checking the intensity or the area within the detected circles.
   - Answer Comparison: Compare detected answers with the answer key. Color-code the results (e.g., red for incorrect, green for correct).
   - Result Display: Display or save the final annotated image showing the identified correct and incorrect answers.
   - Segmentation and Region of Interest (ROI):Segment specific areas for further analysis or visual representation

3. **Execution:**
    - Run the Python script. It will display the processed images of the scanned OMR sheet with the final percentage of the student.
    - Press 'q' to exit the program.

4.2 User Interface

The user interface (UI) of OMR MCQ automated grading project consists of:

1. OMR Processing:

- **Image Acquisition:** Capture high-quality images of the answer sheets using a camera or scanner.
- **Preprocessing:** Enhance the quality of the scanned images (e.g., noise reduction, thresholding).
- **Detection:** Identify and extract marks on the OMR sheet. This might involve using image processing techniques or machine learning models.

2. Grading Logic:

- **Template Matching:** Match detected marks to predefined answer templates.
- **Answer Comparison:** Compare the detected answers with the correct answers and calculate scores.

3. User Interface Design:

- **Upload Interface:** Allow users to upload OMR sheets or images for grading.
- **Feedback Display:** Show grading results, including scores, correct/incorrect answers, and potentially a breakdown of each question.

4. Integration and Tools:

- **Backend Processing**: Use libraries like OpenCV for image processing and algorithms for mark detection**.**

5. Additional Features:

- **Error Handling:** Handle issues like misreads or ambiguous marks.
- **User Management:** If applicable, include features for managing different users or classes.

4.3 Module Implementation

Set Up Your Environment :

- Install necessary libraries.
- pip install opencv-python numpy

Prepare the Data:

- Collect a set of scanned OMR sheets for training and testing.
- Define the format of the OMR sheets (number of questions, choices, layout, etc.).

Preprocessing the Images:

- Convert images to grayscale.
- Apply thresholding to convert the image to a binary image.
- Use contour detection to find the bubbles.

Detecting and Grading the Bubbles:

- Identify the regions where the answers are marked.
- Compare the marked answers with the correct answers to compute the score.

CHAPTER 5 : RESULTS

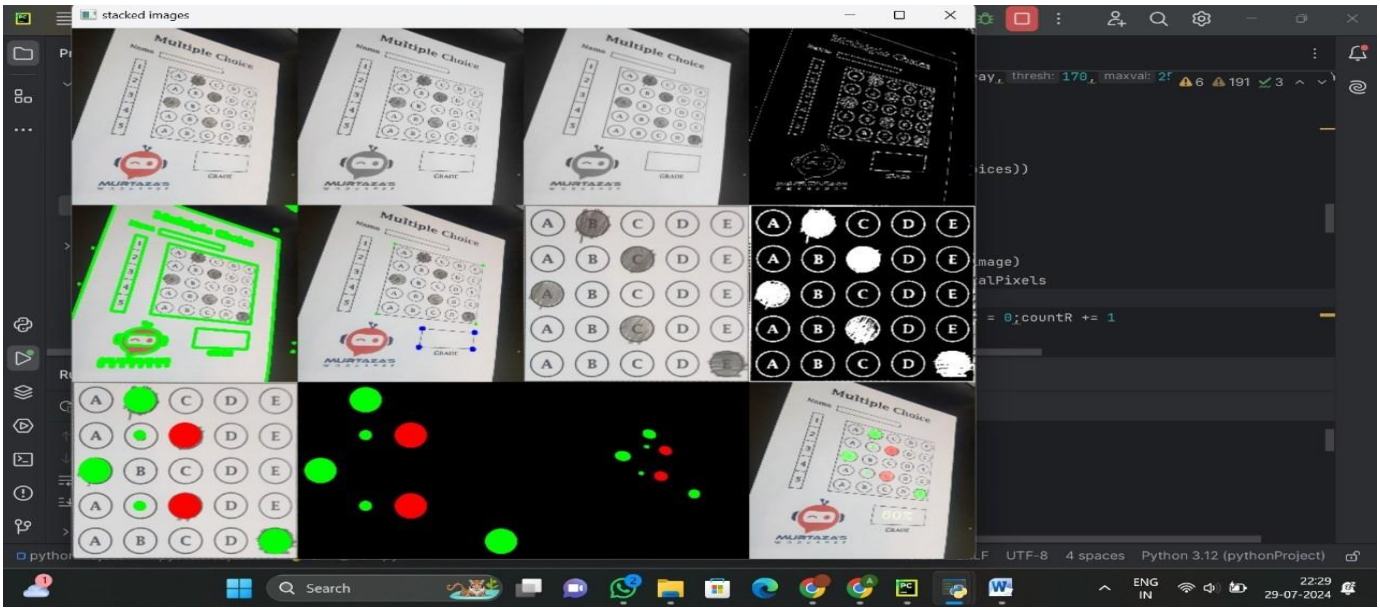

Fig 1 : Image Processing of the OMR sheet

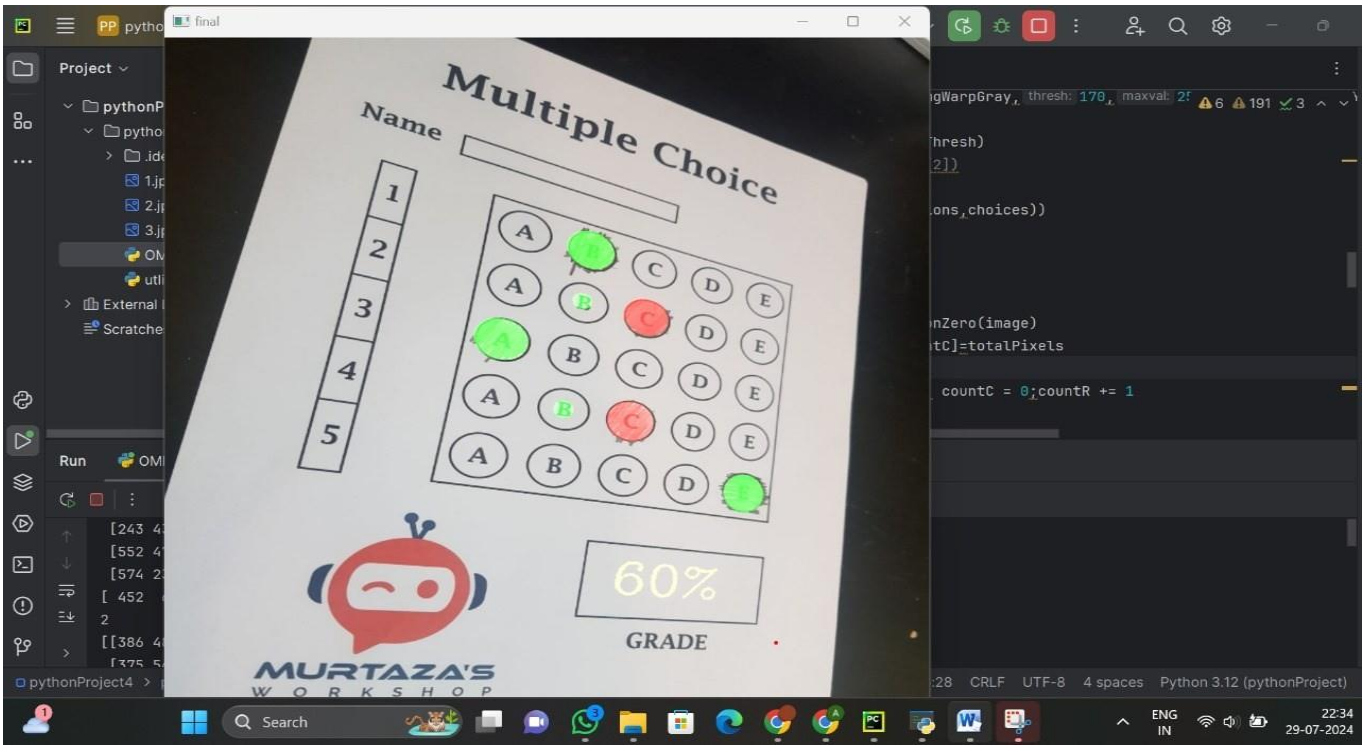

Fig 2: Final Result with Grading

REFERENCES

1. "Programming Computer Vision with Python" by Jan Erik Solem: This book covers practical computer vision techniques using Python and OpenCV, including image processing and machine learning applications.

2. "OpenCV 4 for Secret Agents" by Joseph Howse: Focuses on practical applications of OpenCV for real-world projects, including image and video processing tasks.