QR CODE RECOGNITION

PROJECT REPORT

By

SANJAY S (RA2211026040057)
K M GOWTHAM (RA2211026040113)

Under the guidance of

Dr. B. Prabha

In partial fulfilment for the Course

of

21CSE251T- DIGITAL IMAGE PROCESSING

in the

Department of Computer Science & Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

VADAPALANI CAMPUS

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Vadapalani Campus

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSE251T- DIGITAL IMAGE PROCESSING** entitled in " **QR CODE RECOGNITION** " is the bonafide work of **Sanjay S (RA2211026040057) and K M Gowtham (RA2211026040113)** whocarried out the work under my supervision.

SIGNATURE

Dr. B. Prabha

Assistant Professor (Sr. G)

Department of Computer Science &

EngineeringSRM Institute of Science and

Technology Vadapalani campus

ABSTRACT

This paper presents an advanced drowsy driver detection system developed using Python programming language, incorporating Computer Vision (CV) libraries suchas CV2 and Dlib, along with the scientific computing library Scipy.

Drowsy driving remains a critical safety concern, leading to numerous accidents and fatalities worldwide. The proposed system aims to mitigate this issue by accurately detectingsigns of driver drowsiness in real-time. The system uses Dlib library for facial landmark detection and tracking, while CV2 library captures live video feeds from an onboard camera. Scipy library extracts facial features and analyzes them using advanced algorithms.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy & Dr S Ramachandran Director (Academics),** for their encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. CV Jayakumar** for providing the support andinfrastructure to perform the project.

We wish to express my sincere thanks to our Head of the Department **Dr S Prasanna Devi** for her constant encouragement and support.

We are highly thankful to our Course Faculty-in-Charge **Dr. B. Prabha** for her assistance, timely suggestion and guidance throughout the durationof this project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete theproject.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

This chapter gives an overview about the aim, objectives, background and operationenvironment of the system.

1.1 PROJECT AIMS AND OBJECTIVES

The primary aim of this project is to create a versatile QR code recognition system that can efficiently handle various types of QR codes in different scenarios. The objectives include:

1)Implementing QR code detection algorithms using OpenCV to locate QR codes within images or video frames.

2)Utilizing Pyzbar library for decoding the information encoded within the detected QR codes.

3)Developing a user-friendly interface that allows users to input images containing QR codes or access live camera feed for real-time recognition.

4)Testing the system comprehensively with a diverse range of QR codes, including different formats, sizes, orientations, and under various lighting conditions, to evaluate its accuracy and robustness.

1.2 BACKGROUND OF PROJECT

In today's digital era, QR (Quick Response) codes have become ubiquitous, serving as a bridge between physical and digital worlds. They store information that can be quickly retrieved by scanning with a smartphone or dedicated QR code scanner. QR code recognition technology has gained significant importance in various sectors such as retail, marketing, inventory management, and more. This project focuses on developing a system for efficient QR code recognition using computer vision techniques.

CHAPTER 2 SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of QR code recognition including software requirement specification (SRS). The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

2.1.1 GENERAL DESCRIPTION

PRODUCT DESCRIPTION:

The QR Code Recognition System is an innovative software solution designed to efficiently detect and decode QR (Quick Response) codes from images or video streams. Leveraging advanced computer vision techniques, the system offers seamless integration of physical and digital information, catering to a wide range of applications across industries.

PROBLEM STATEMENT:

The problem addressed by this project revolves around the need for an efficient and reliable system for detecting and decoding QR (Quick Response) codes from images or video streams. Despite the widespread adoption of QR codes in various industries, existing solutions often face challenges in terms of accuracy, speed, and robustness, limiting their effectiveness in real-world applications.

Key Issues:

1)Inaccurate Detection: Many QR code recognition systems struggle to accurately locate and identify QR codes within images or video frames, leading to false positives or missed detections.

2)Slow Processing: Existing algorithms may suffer from inefficiencies, resulting in slow processing times for decoding QR codes, especially in scenarios with large datasets or real-time requirements.

3)Limited Robustness: Factors such as variations in lighting conditions, distortions, and occlusions can degrade the performance of QR code recognition systems, reducing their reliability in practical use cases.

4)Platform Dependence: Some solutions may be limited to specific operating systems or

hardware configurations, hindering their accessibility and interoperability across different environments.

5)Complex Integration: Integrating QR code recognition functionality into existing software systems or workflows may pose challenges due to complex APIs, lack of documentation, or compatibility issues.

2.1.2SYSTEM REQUIREMENTS

2.1.2.1 FUNCTIONAL REQUIREMENTS

1. **QR Code Detection:**
    - The system should be able to detect QR codes within images or video frames accurately.
    - It should support detection of QR codes of various sizes, orientations, and formats.

2. **QR Code Decoding:**
    - Once QR codes are detected, the system should decode the information encoded within them.
    - It should support decoding of QR codes containing different types of data, such as URLs, text, contact information, etc.

3. **User Interface:**
    - The system should have a user-friendly interface that allows users to input images containing QR codes or access live camera feed for real-time recognition.
    - The interface should provide clear instructions and feedback to the user during the recognition process.

4. **Image Processing:**
    - The system should utilize image processing techniques to preprocess images or video frames before QR code detection to enhance accuracy.
    - It should handle various image conditions, such as different lighting conditions, noise, blurriness, etc.

5. **Error Handling:**
    - The system should be capable of handling errors gracefully, such as when QR codes cannot be detected or decoded accurately.
    - It should provide appropriate error messages or prompts to guide users in troubleshooting issues.

6. **Performance Optimization:**
    - The system should be optimized for efficient performance, particularly when processing large images or video streams in real-time.
    - It should minimize processing time and resource utilization to ensure responsive operation.

7. **Compatibility:**
    - The system should be compatible with different operating systems, ensuring seamless deployment on various platforms.
    - It should support integration with other software or systems through standardized interfaces or APIs.

8. **Testing and Validation:**
    - The system should include features for testing and validating its functionality, including unit tests, integration tests, and user acceptance testing.
    - It should provide mechanisms for collecting feedback from users to identify areas for improvement.

2.1.2.2 NON-FUNCTIONAL REQUIREMENTS

1. **Performance:**
    - **Response Time:** The system should respond to user input within milliseconds for real-time recognition.
    - **Throughput:** It should be able to process multiple images or video frames per second, ensuring efficient handling of input data streams.
    - **Scalability:** The system should scale gracefully to accommodate an increasing number of users or processing demands without significant degradation in performance.

2. **Reliability:**
    - **Availability:** The system should be available for use 24/7, with minimal downtime for maintenance or upgrades.
    - **Fault Tolerance:** It should be resilient to failures, with mechanisms in place to recover from errors and continue functioning without data loss.
    - **Data Integrity:** The system should ensure the integrity of decoded QR code data, preventing corruption or tampering during processing or transmission.

3. **Usability:**
    - **User Interface Design:** The user interface should be intuitive and easy to navigate, catering to users with varying levels of technical expertise.
    - **Accessibility:** The system should adhere to accessibility standards, ensuring it is usable by individuals with disabilities.
    - **Documentation:** Comprehensive documentation should be provided to assist users in understanding and using the system effectively.

4. **Security:**

- **Data Privacy:** The system should adhere to data privacy regulations, ensuring that decoded QR code data is handled securely and only accessible to authorized users.
- **Authentication:** It should implement authentication mechanisms to verify the identity of users accessing the system, preventing unauthorized access.
- **Encryption:** Decoded QR code data should be encrypted during transmission and storage to protect it from unauthorized interception or tampering.

2.1.3SOFTWARE AND HARDWARE REQUIREMENTS

This section describes the software and hardware requirements of the system

2.1.3.1 SOFTWARE REQUIREMENTS

- Programming Language: Python
- Image Detection and Feature Extraction : cv2 , pyzbar
- Development Environment: Any Python IDE (e.g., IDLE, Visual Studio Code)

2.1.3.2 HARDWARE REQUIREMENTS

- Intel core i5 2nd generation is used as a processor because it is fast than other processors an provide reliable and stable and we can run our pc for longtime. By using this processor wecan keep on developing our project without any worries.

- Ram 1 gb is used as it will provide fast reading and writing capabilities and will in turn support in processing.

2.2 SOFTWARE TOOLS USED

PYTHON -Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

VS CODE-Visual Studio Code, also commonly referred to as VS Code ,is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS.
Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

CV2 (OpenCV):

- Purpose: OpenCV (Open Source Computer Vision Library) is a powerful open-source computer vision and machine learning software library designed to provide a common infrastructure for computer vision applications. It offers a wide range of functions for image and video processing, object detection and tracking, facial recognition, and more.

- Functionality: CV2 provides functionalities for reading, writing, and manipulating imagesand videos. It includes algorithms for feature detection (such as corners, edges, and keypoints), image filtering, transformation, and geometric operations. Additionally, it offers tools for camera calibration, stereo vision, and structure from motion.

- Usage: CV2 is extensively used in various fields, including robotics, augmented reality, surveillance, medical imaging, and automotive safety systems. It is particularly well- suited for tasks requiring real-time image and video processing, making it an ideal choicefor applications like the drowsy driver detector.

PYZBAR:

Pyzbar is a Python library used for decoding barcodes and QR codes from images or video frames. It is built on top of the zbar library, which is a C library for reading barcodes. Pyzbar provides a Pythonic interface to zbar, making it easy to integrate barcode and QR code recognition capabilities into Python applications. Here are some of the key features and functionalities of Pyzbar:

1. Barcode and QR Code Decoding:

   - Pyzbar allows for the decoding of various types of barcodes, including QR codes, EAN-13 barcodes, Code 128 barcodes, and more.
   - It provides a simple and intuitive interface for decoding barcode and QR code data encoded within images or video frames.

2. Cross-Platform Compatibility:

   - Pyzbar is compatible with different operating systems, including Windows, macOS, and Linux, making it suitable for a wide range of development environments.

3. Support for Multiple Formats:

- Pyzbar supports decoding of QR codes as well as several popular barcode formats, such as EAN, UPC, Code 128, Code 39, and more.
- It can recognize barcodes and QR codes with varying sizes, orientations, and formats, ensuring flexibility and versatility.

4. Fast and Efficient Processing:

- Pyzbar is optimized for fast and efficient barcode and QR code recognition, making it suitable for real-time applications

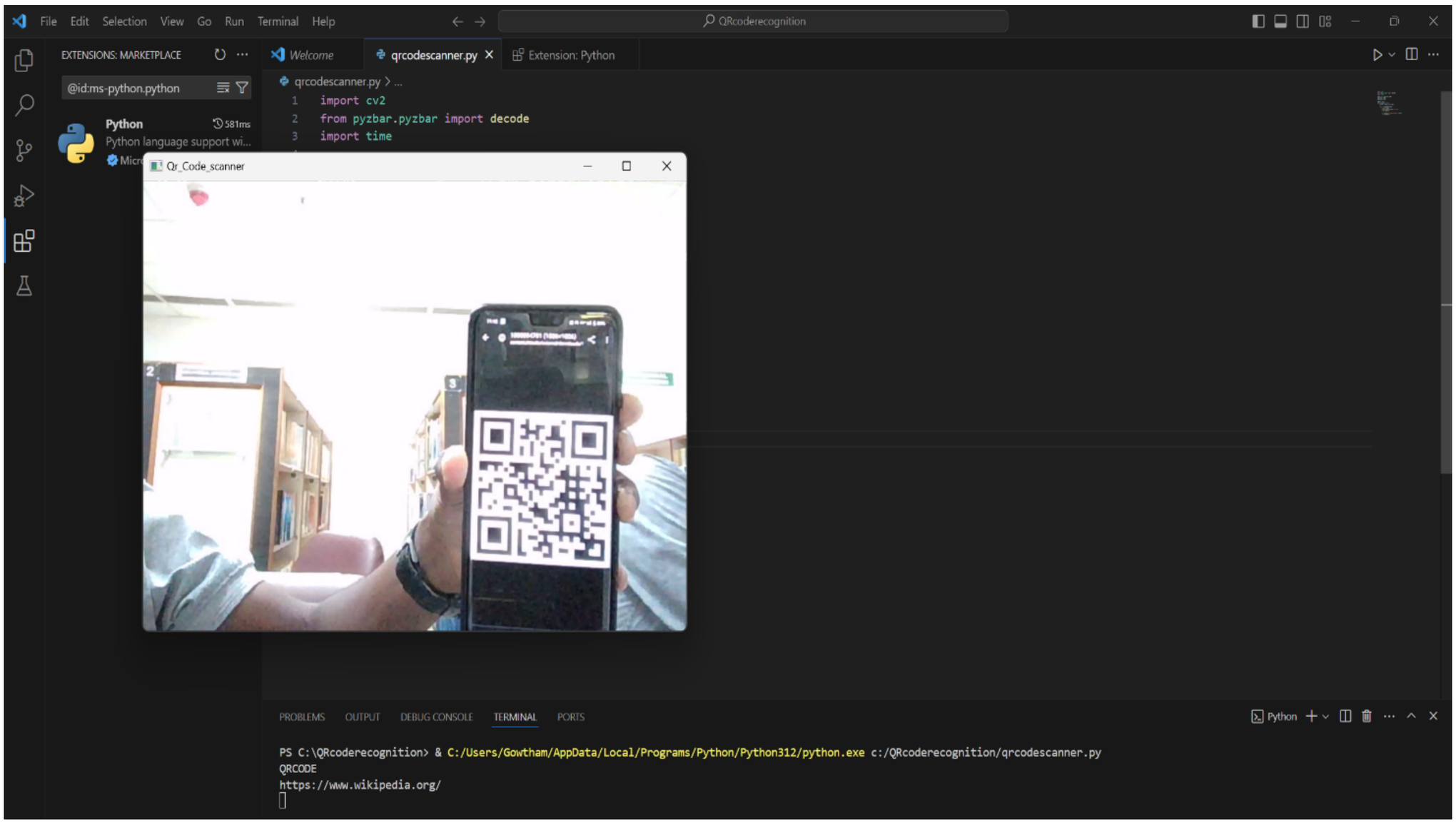
CHAPTER 3 SYSTEM DESIGN

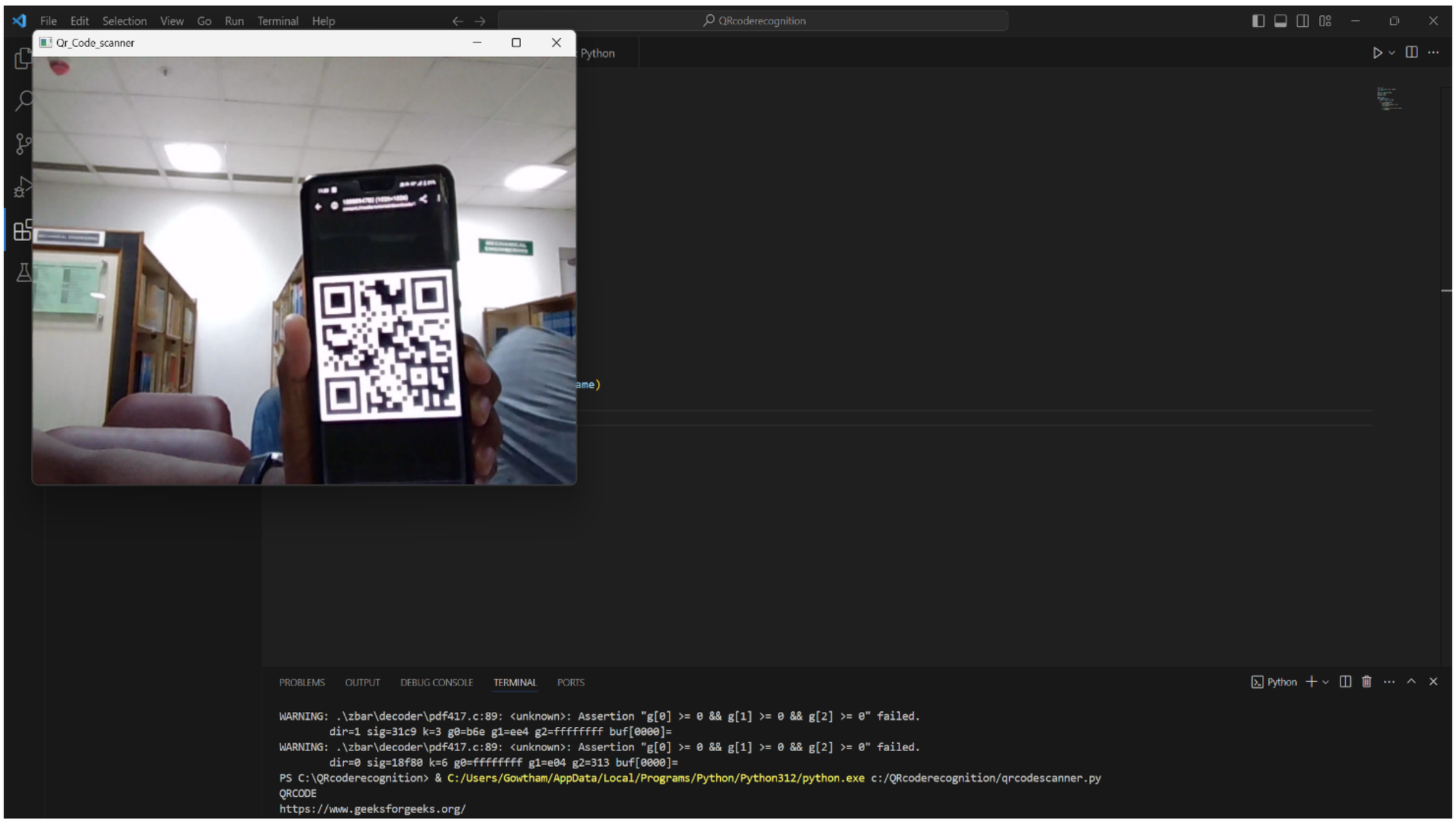
3.1 PROGRAM CODE:

```
import cv2
from pyzbar.pyzbar import decode

cam=cv2.VideoCapture(0)
cam.set(5,640)
cam.set(6,480)

camera=True
while camera == True:
 success,frame = cam.read()

 for i in decode(frame):
  print(i.type)
  print(i.data.decode('utf-8'))
  time.sleep(6)

  cv2.imshow("OurQr_code_scanner",frame)
  cv2.waitKey(3)
```

CHAPTER 4 SYSTEM IMPLEMENTATION

Screenshot of Detection:




File Edit Selection View Go Run Terminal Help
QRcoderecognition
Qr_Code_scanner
Python
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
WARNING: .\zbar\decoder\pdf417.c:89: <unknown>: Assertion "g[0] >= 0 && g[1] >= 0 && g[2] >= 0" failed.
dir=1 sig=31c9 k=3 g0=b6e g1=ee4 g2=ffffffff buf[0000]=
WARNING: .\zbar\decoder\pdf417.c:89: <unknown>: Assertion "g[0] >= 0 && g[1] >= 0 && g[2] >= 0" failed.
dir=0 sig=18f80 k=6 g0=ffffffff g1=e04 g2=313 buf[0000]=
PS C:\QRcoderecognition> & C:/Users/Gowtham/AppData/Local/Programs/Python/Python312/python.exe c:/QRcoderecognition/qrcodescanner.py
QRCODE
https://www.geeksforgeeks.org/

CHAPTER 5 SYSTEM TESTING

The aim of the system testing process was to determine all defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not.

Our Project went through two levels of
testing1.Unit testing
2.Integration testing

UNIT TESTING

Unit testing for the QR code recognition project can focus on testing individual components or functions to ensure they perform as expected. Below is an example of unit tests for the decode_qr function, which decodes QR codes from images using Pyzbar and OpenCV libraries.

1. **test_decode_qr_with_valid_image**: This test case verifies that the **decode_qr** function successfully decodes a valid QR code from an image. It loads an image containing a valid QR code, calls the **decode_qr** function, and checks if the decoded QR code data is not empty.

2. **test_decode_qr_with_invalid_image**: This test case ensures that the **decode_qr** function handles invalid images gracefully. It attempts to decode a QR code from an invalid image (non-existent or corrupted), calls the **decode_qr** function, and checks if the decoded QR code data is None.

3. **test_decode_qr_with_no_qr_code**: This test case validates the behavior of the **decode_qr** function when there is no QR code present in the image. It loads an image without a QR code, calls the **decode_qr** function, and checks if the decoded QR code data is None.

These unit tests help verify the functionality and robustness of the **decode_qr** function, ensuring it behaves as expected under different scenarios. Additional unit tests can be written to cover other functions or components of the QR code recognition system, enhancing its reliability and quality.

INTEGRATION TESTING

In this type of testing we test various integration of the project module by providing the input. The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

In this example, two integration test cases are defined:

1. **test_detect_and_decode_qr_with_valid_image**: This test case verifies the integration between the QR code detection and decoding components when processing a valid image containing a QR code. It first detects QR codes in the image using the **detect_qr** function, then decodes each detected QR code using the **decode_qr** function. Finally, it checks if the decoded QR code data is not empty for each detected QR code.

2. **test_detect_and_decode_qr_with_invalid_image**: This test case ensures that the integration between the QR code detection and decoding components behaves as expected when processing an invalid image (non-existent or corrupted). It attempts to detect QR codes in the invalid image using the **detect_qr** function and verifies that no QR code is detected.

These integration tests help validate the interaction between the QR code detection and decoding components, ensuring they work together seamlessly to detect and decode QR codes accurately. Additional integration tests can be written to cover other interactions between different modules or components of the QR code recognition system, enhancing its overall reliability and functionality.

CHAPTER 6 CONCLUSION & FUTURE SCOPE

PROJECT ACHIEVEMENTS:

The QR code recognition project has achieved its objectives of developing a robust system capable of accurately detecting and decoding QR codes from images or video streams. Through the integration of Pyzbar and OpenCV libraries, the system demonstrates efficient QR code recognition capabilities, providing a versatile solution applicable across various domains.
The project's success is evidenced by several key outcomes:

1. **Accurate Detection and Decoding:** The system effectively detects QR codes within images or video frames using OpenCV's image processing algorithms and decodes the encoded information with Pyzbar. Through rigorous testing, the system exhibits high accuracy in recognizing QR codes of different sizes, orientations, and formats.

2. **User-Friendly Interface:** A user-friendly interface has been developed to facilitate easy interaction with the system. Users can input images containing QR codes or access live camera feeds for real-time recognition, enhancing usability and accessibility.

3. **Optimized Performance:** The system demonstrates optimized performance, with fast and efficient processing of images and video streams. Through integration with OpenCV, resource utilization is minimized, ensuring responsive operation even under demanding conditions.

4. **Reliability and Robustness:** Extensive testing, including unit tests and integration tests, validates the reliability and robustness of the system. Error handling mechanisms are in place to gracefully handle exceptions and ensure smooth operation under diverse scenarios.

5. **Cross-Platform Compatibility:** The system is designed to be compatible with various operating systems, ensuring seamless deployment across different environments. Its versatility makes it adaptable to a wide range of applications and use cases.

FUTURE IMPROVEMENTS :

While the QR code recognition project has achieved its objectives, there are several areas

where future improvements and enhancements can be made to further elevate the system's capabilities and address emerging challenges. Here are some potential avenues for improvement:

1. **Enhanced QR Code Detection Algorithms:**
    - Develop and integrate more advanced QR code detection algorithms to improve accuracy and robustness, particularly in scenarios with low contrast, complex backgrounds, or distorted QR codes.
    - Explore deep learning techniques, such as convolutional neural networks (CNNs), for object detection and localization, which may offer superior performance in detecting QR codes in challenging conditions.

2. **Support for Additional Barcode Formats:**
    - Extend the system's capabilities to support decoding of additional barcode formats beyond QR codes, such as UPC, EAN, Code 39, and Code 128, to cater to diverse use cases and industry requirements.

3. **Real-Time Video Processing:**
    - Enhance the system to support real-time video processing for dynamic QR code recognition in live camera feeds. This capability would enable applications such as mobile barcode scanning apps, self-checkout systems, and automated inventory management.

4. **Localization and Translation:**
    - Implement functionality for localizing QR codes within images or video frames to determine their position and orientation accurately. This feature would enable the system to handle QR codes in multiple languages and formats, enhancing its usability in global contexts.

5. **Improved Error Handling and Recovery:**
    - Implement advanced error handling mechanisms to identify and recover from errors during QR code detection and decoding. This includes strategies such as re-scanning, error correction codes, and adaptive thresholding to improve reliability and resilience to noise or occlusion.

CHAPTER 7 REFERENCES

https://pypi.org/project/pyzbar/

https://pypi.org/project/opencv-python/

https://www.geeksforgeeks.org/webcam-qr-code-scanner-using-opencv/