**ID CARD SCANNER**



A mini project-1 report submitted in partial fulfillment of requirements for the award of Degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**Vuluvabeeti Saroja (199X1A05G7)**

**Under the Esteemed guidance of**

**Dr. N. Kasiviswanath**

**Head of the Department**

**Department of C.S.E.**

**Department of Computer Science and Engineering**

**G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL**

**(Affiliated to JNTUA, ANANTAPURAMU)**

**2021 - 2022**

**Department of Computer Science and Engineering**

**G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL**

**(Affiliated to JNTUA, ANANTAPURAMU)**



**CERTIFICATE**

This is to certify that the mini project-1 work entitled ‘ID CARD SCANNER’ is a bona fide record of work carried out by

**V. SAROJA (199X1A05G7)**

Under my guidance and supervision in fulfillment of the requirements for the award of degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**HEAD OF THE DEPARTMENT AND PROJECT GUIDE**

**Dr. N. Kasiviswanath,**

Professor & Head of the Department,

Department of CSE.

G. Pulla Reddy Engineering College,

Kurnool.

**DECLARATION**

We hereby declare that the project titled “ID CARD SCANNER” is the authentic work carried out by us as students of G. PULLA REDDY ENGINEERING COLLEGE (Autonomous) Kurnool, during October 2021 – February 2022 and has not been submitted elsewhere for the award of degree in part or in full to any institute.

**V. SAROJA**

**(199X1A05G7)**

**Acknowledgement**

We wish to express our deep sense of gratitude to our project guide, **Dr. N. Kasi Viswanath** Head of the Department, CSE, G. Pulla Reddy Engineering College, for his immaculate guidance, constant encouragement and cooperation which have made possible to bring out this project work.

We are grateful to our project in charge, **Sri. S. Vinay Kumar**, Asst Professor of CSE Department, G. Pulla Reddy Engineering College, for helping us and giving us the required information needed for our project work.

We are thankful to our Head of the Department **Dr. N. Kasi Viswanath**, for his whole hearted support and encouragement during the project sessions.

We are grateful to our respected Principal **Dr. B. Sreenivasa Reddy**, for providing requisite facilities and helping us in providing such a good environment.

We wish to convey our acknowledgements to all the staff members of the Computer Science Engineering Department for giving the required information needed for our project work.

Finally, we wish to thank all our friends and well-wishers who have helped us directly or indirectly during the course of this project work.

**CONTENTS**

|  |  |
| --- | --- |
| **ABSTRACT** |  |
| **LIST OF FIGURES** |  |
| 1. **Introduction** | 1 |
| 1.1 Introduction | 1 |
| 1.2 Objective of the project | 3 |
| 1.3 Organization of the report | 5 |
| **2. System Specifications** | 6 |
| 2.1 Software Specification | 6 |
| 2.2 Hardware Specification | 7 |
| **3. Design & Implementation** | 8 |
| 3.1 Introduction | 10 |
| 3.2 Source Code | 20 |
| 3.2.1 Output Screens | 26 |
| 3.3 Testing and Validation | 35 |
| **4. Conclusion** | 40 |
| **References** | 42 |

**Abstract**

An identification card is a card that contains a person's name, photograph, date of birth, and other personal information. It aids in the determination of a person's identification.

Companies issue ID cards to their employees. To get access to enter a college, students must present an ID card. This ensures the organization's safety and security by allowing only authorized personnel to access the facility. An ID Scanner alleviates the need for a human to manually check the data.

In this project, we intend to create an ID card Scanner. The ID card is scanned, and the information on it is segmented using OCR. The information scanned from the ID card can thus be used to determine whether or not a person is authenticated to enter the college.

**LIST OF FIGURES**

Fig 1.1 Scanning the information from an id card

Fig 1.2 General working of OCR

Fig 3.1.1 Overview of Google Colab

Fig 3.1.2 Anvil uplink code to connect to an app

Fig 3.1.3 Working of pillow

Fig 3.1.4 Image processing in OpenCV

Fig 3.2.1 Installing anvil uplink

Fig 3.2.2 Installing the required packages

Fig 3.2.3 Installing the requires packages

Fig 3.2.4 Importing the required packages and Establishing server connection

Fig 3.2.5 Capturing the Image

Fig 3.2.6 Authentication of the ID Card

Fig 3.2.7 ID Card Scanner App

Fig 3.2.8 Student ID card

Fig 3.2.9 Final output of the App

**1. INTRODUCTION**

1. **INTRODUCTION**
   1. **Introduction**

The development of information technology has developed quite rapidly, both in theory and application. A lot of research technology has been used to facilitate and accelerate human work. The research has been implemented on computers and is being used to optimally accomplish human tasks.

An identification card is a card that contains a person's name, photograph, date of birth, and other personal information. It aids in the determination of a person's identification.

To get access to the college, students must present an ID car. This ensures the organization's safety and security by allowing only authorized personnel to access the facility. An ID Scanner alleviates the need for a human to manually check the data.



Fig 1.1 Scanning the information from an id card

Humans can understand the contents of an image simply by looking. We perceive the text on the image as text and can read it. Computers don't work the same way. They need something more concrete, organized in a way they can understand.

This is where Optical Character Recognition (OCR) kicks in. Whether it's recognition of car plates from a camera, or hand-written documents that should be converted into a digital copy, this technique is very useful. While it's not always perfect, it's very convenient and makes it a lot easier and faster for some people to do their jobs.

Optical Character Recognition involves the detection of text content on images and translation of the images to encoded text that the computer can easily understand. An image containing text is scanned and analyzed in order to identify the characters in it. Upon identification, the character is converted to machine-encoded text.

How is it really achieved? To us, text on an image is easily discernible and we are able to detect characters and read the text, but to a computer, it is all a series of dots.

The image is first scanned and the text and graphics elements are converted into a bitmap, which is essentially a matrix of black and white dots. The image is then pre-processed where the brightness and contrast are adjusted to enhance the accuracy of the process.

The image is now split into zones identifying the areas of interest such as where the images or text are and this helps kick off the extraction process. The areas containing text can now be broken down further into lines and words and characters and now the software is able to match the characters through comparison and various detection algorithms. The final result is the text in the image that we're given.

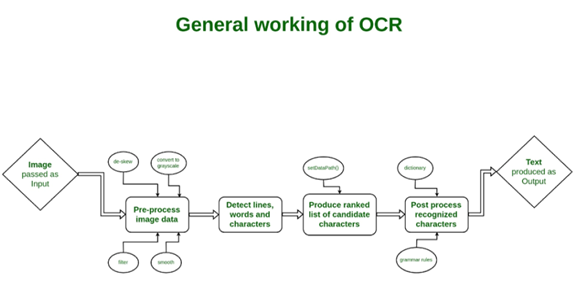


Fig 1.2 General working of OCR

Based on some research above, this study compares the result of character recognition of name and NIK (identity number) in ID Card using two different tesseract models. The first model uses the train data manually that created from five ID Card as data set and training on tesseract 3.05 with the support of software QT-box version 1.08. While the second model uses train data that already contained in tesseract 4.0, which is a data train that contains text data in Indonesian language with different fonts and using tesseract version 4.0 for OCR which in that version has implemented neural network model that is LSTM.

Character recognition using Optical Character Recognition (OCR) technique with tesseract tools. This research will compare the character recognition model using the data train manually that trained using tesseract 3.05 and data train that already contained in tesseract 4.0 which containing Indonesian language text data with different font.

The output can now be used to verify whether the student is permitted to enter the college or not.

**1.2 Objective of the Project**

In today's environment, security failures on college campuses are a genuine issue and an increasing one. The project's goal is to improve the security and safety of the campus by allowing only authenticated people inside the building and reducing the amount of manual effort required for humans to verify IDs.

Since every student has a unique ID, which consists of a unique number and the student's details, which are provided by the college management, we are going to scan that unique ID card to extract the details and use those details to check if the student is authenticated to enter the college or not. With the help of machine learning techniques, we are going to extract the roll number from the ID card through a camera. And check whether the roll number has authentication or not.

The major goal is to improve the college's safety and security by deploying an ID card scanner that scans each student's unique ID card and reduces the amount of time that humans spend manually verifying IDs.

* 1. **Organization of the Project**

In this project, we have developed a web application that captures a picture of the student ID card and verifies whether the student is authorized to enter the college or not.

The data of the student is extracted from the image with the help of Tesseract OCR engine. JavaScript captures the image through web camera. Image pre-processing can be done with the help of OpenCV library.

We have implemented the code in Google Colab and designed a web application as a front-end to the Google Colab notebook.

**2. SYSTEM SPECIFICATIONS**

1. **SYSTEM SPECIFICATIONS**

**2.1 Introduction**

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

**Nonfunctional Requirements**

Nonfunctional requirements are the functions offered by the system.

It includes time constraints on the development process and standards. The nonfunctional requirements are as follows:

* **Speed:**  The system should process the given input into output within appropriate time.
* **Ease of use:** The software should be user friendly. Then the customers can use easily, so it doesn’t require much training time.
* **Reliability:** The rate of failure should be less then only the system is more reliable
* **Portability:** It should be easy to implement in any system.

**Specific Requirements**

**DEPT F CSE, GPREC, KNL**

The specific requirements are:

* **User Interfaces:** The external users are clients. All the clients can use this software for indexing and searching.
* **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC’s may be laptops with wireless LAN as the internet connections provided will be wireless
* **Software Interfaces:** The operating Systems can be any versions of windows.
* **Performance Interfaces:** The PC’s used must be at least Pentium 4 machines so that they can give optimum performance of the product.

**2.2 Software Specifications**

Software specification deal with defining software resources requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Software Requirements:**

Operating System : Windows, Linux, macOS

IDE : Google Colab

Languages : Python, JavaScript

Libraries : Pillow, Tesseract, OpenCV, Anvil

**2.3 Hardware Specifications**

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

All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the central processing unit (CPU) is a fundamental System requirement for any software. Most software running on x86 architecture define processing power as the model land the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMDAthlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds.

**Hardware Requirements:**

* RAM : 8GB
* Processor : Intel I5
* Hard Disk : 2TB
* PC or Laptop with WebCam

**3. DESIGN AND IMPLEMENTATION**

1. **DESIGN AND IMPLEMENTATION**
   1. **Introduction**

**JavaScript**

JavaScript (JS) is a lightweight, interpreted, or just in time compiled programming language with first-class function. While it is most well-known as the scripting

language for Web pages, many non-browser environments also use it, such as Node Js, Apache CouchDB and Adobe Acrobat.

JavaScript is a protype based, multi-paradigm,

single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (functional programming) styles

It is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

• JavaScript is a light weight, interpreted programming language.

• Designed for creating network-centric applications.

• Complementary to and integrated with Java.

• Complementary to and integrated with HTML.

• Open and cross-platform.

The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly

**Python**

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

**Google Colab**

Google Colab was developed by Google to provide free access to GPU’s and TPU’s to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook**.**

**Google Colab features**

Google Colab provides tons of exciting features that any modern IDE offers, and much more. Some of the most exciting features are listed below.

* Interactive tutorials to learn machine learning and neural networks.
* Write and execute Python 3 code without having a local setup.
* Execute terminal commands from the Notebook.
* Import datasets from external sources such as Kaggle.
* Save your Notebooks to Google Drive.
* Import Notebooks from Google Drive.
* Free cloud service, GPUs and TPUs.
* Integrate with PyTorch, Tensor Flow, Open CV.
* Import or publish directly from/to GitHub.

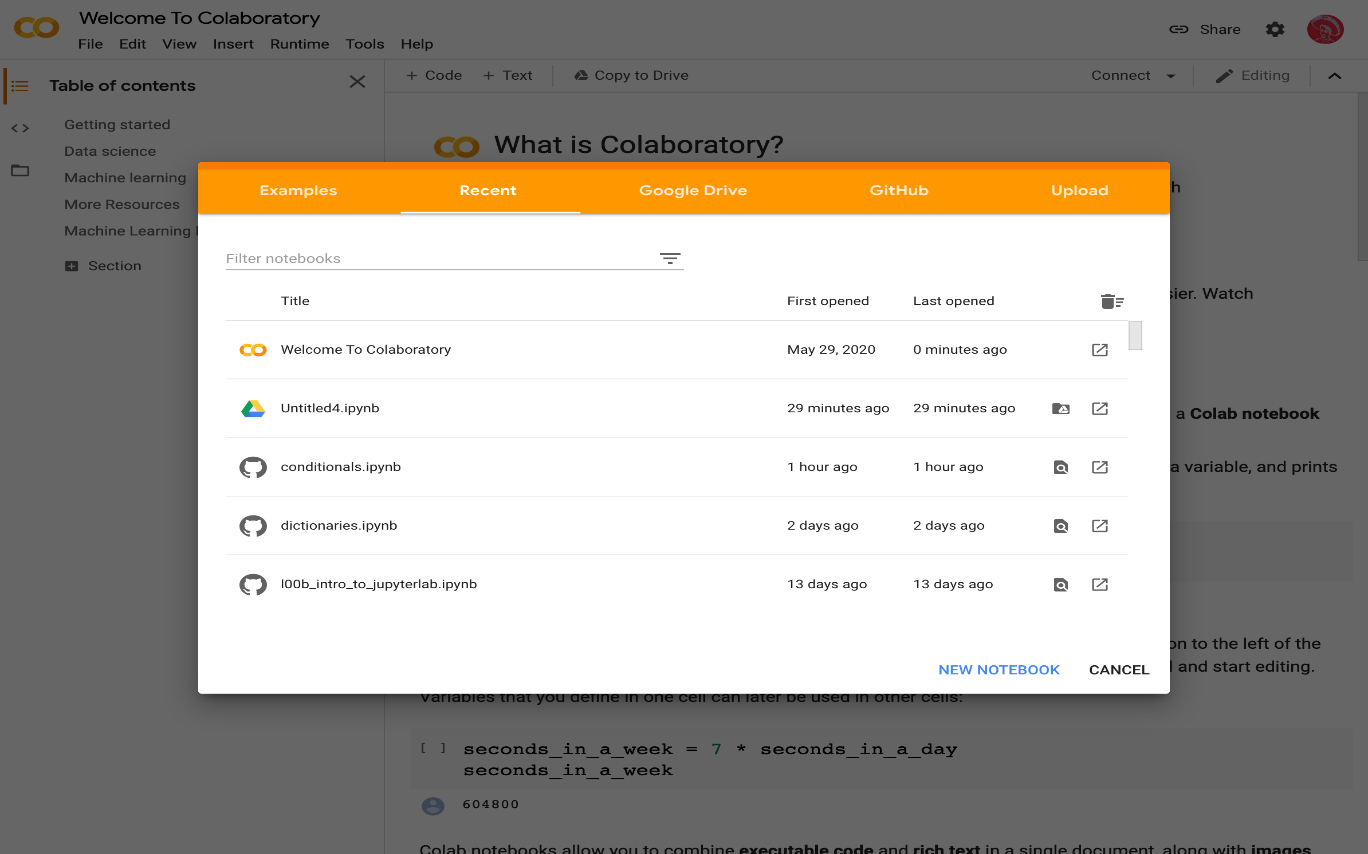


Fig 3.1.1 Overview of Google Colab

**Anvil**

Anvil is a new way to build web apps, with nothing but Python. This documentation will tell you all about how it works

**Uplink: Code outside Anvil**

Sometimes you want to write some server-side code that can’t run on our servers. Perhaps it needs resources that are only available on your company network, or on your computer.

The Uplink makes any Python code behave like a Server Module: you can call functions in it from your app using *anvil.server.call*. The connection goes both ways; you can *anvil.server.call* into your app from the Uplink.

Anything you can do in a Server Module can also be done from the Uplink - using *app\_tables* to search and modify your Data Tables, or using *anvil.users.get\_user()* to check who’s logged in, for example.

It’s common to use the Uplink to connect to a Jupyter Notebook to create an Anvil front-end for your Data Science models.

Here’s a script that you can run on your own machine and connect to your app.

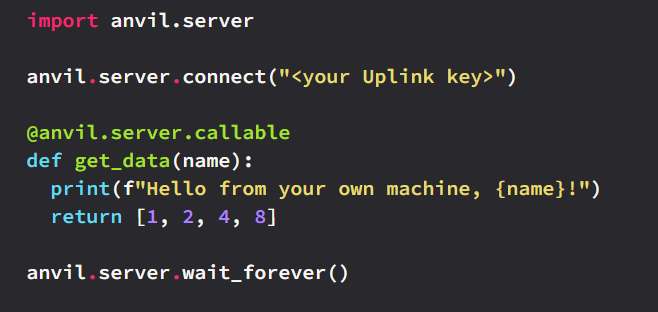


Fig 3.1.2 Anvil uplink code to connect to an app

**Python Imaging Library (PILLOW)**

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors. As of 2019, Pillow development is supported by Tidelift.

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

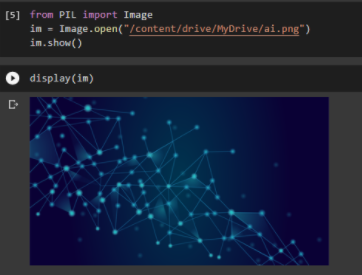


Fig 3.1.3 working of pillow

**OpenCV**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.



Fig 3.1.4 Image processing in OpenCV

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start-ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

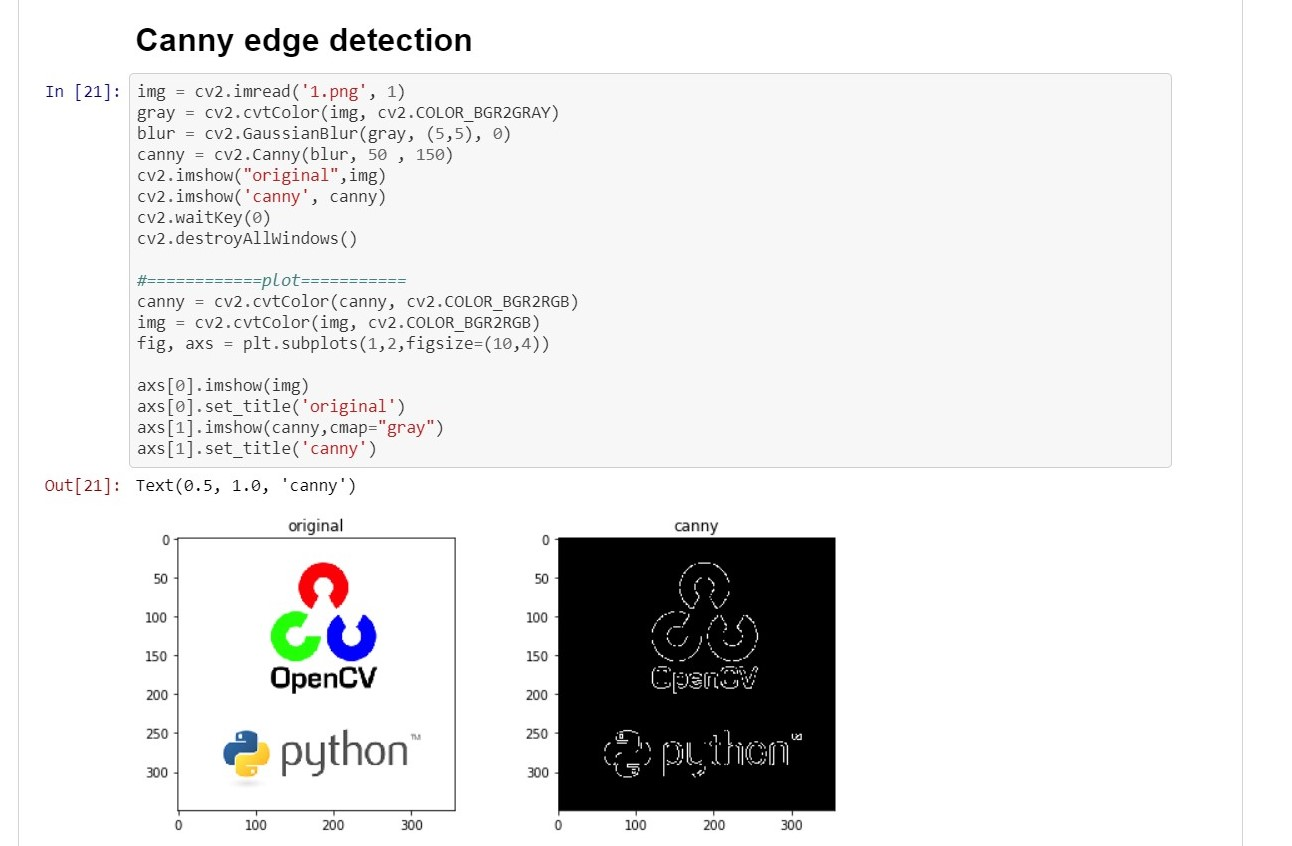


Fig 3.1.5 Working of OpenCV

**Tesseract**

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

**Functions**

* *get\_languages* Returns all currently supported languages by Tesseract OCR.
* *get\_tesseract\_version* Returns the Tesseract version installed in the system.
* *image\_to\_string* Returns unmodified output as string from Tesseract OCR processing
* *image\_to\_boxes* Returns result containing recognized characters and their box boundaries
* *image\_to\_data* Returns result containing box boundaries, confidences, and other information. Requires Tesseract 3.05+. For more information, please check the Tesseract TSV documentation
* *image\_to\_osd* Returns result containing information about orientation and script detection.
* *image\_to\_alto\_xml* Returns result in the form of Tesseract’s ALTO XML format.
* *run\_and\_get\_output* Returns the raw output from Tesseract OCR. Gives a bit more control over the parameters that are sent to tesseract.

**Parameters**

* *image* Object or String - PIL Image/NumPy array or file path of the image to be processed by Tesseract. If you pass object instead of file path, pytesseract will implicitly convert the image to [RGB mode](https://pillow.readthedocs.io/en/stable/handbook/concepts.html#modes).
* *lang* String - Tesseract language code string. Defaults to eng if not specified! Example for multiple languages: lang='eng+fra'
* *config* String - Any **additional custom configuration flags** that are not available via the pytesseract function. For example: config='--psm 6'
* *nice* Integer - modifies the processor priority for the Tesseract run. Not supported on Windows. Nice adjusts the niceness of unix-like processes.
* *output\_type* Class attribute - specifies the type of the output, defaults to string. For the full list of all supported types, please check the definition of [*pytesseract.Output*](https://github.com/madmaze/pytesseract/blob/master/pytesseract/pytesseract.py) class.
* *timeout* Integer or Float - duration in seconds for the OCR processing, after which, pytesseract will terminate and raise RuntimeError.
* *pandas\_config* Dict - only for the *Output.DATAFRAME* type. Dictionary with custom arguments for [*pandas.read\_csv*](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html#pandas-read-csv)*.* Allows you to customize the output of *image\_to\_data.*

**Reading Text from the Image using Tesseract**

Here, we will use the tesseract package to read the text from the given image.

Mainly, 3 simple steps are involved here as shown below

* Loading an Image saved from the computer or download it using a browser and then loading the same. (Any Image with Text).
* Binarizing the Image (Converting Image to Binary).
* We will then Pass the Image through the OCR system.

**3.2 Source Code**

INSTALLING PACKAGES

!pip install anvil-uplink

!sudo apt install tesseract-ocr

!pip install pytesseract

IMPORTING PACKAGES

import cv2

import numpy as np

import pytesseract

import matplotlib.pyplot as plt

from PIL import Image

import anvil.server

anvil.server.connect("RWFIVIQ4QFHCNNSUGNW5A75Z-SVRWJCQTMVBAYHTV")

CAMERA CAPTURE

from IPython.display import display, Javascript

from google.colab.output import eval\_js

from base64 import b64decode

def take\_photo(filename='photo.jpg', quality=0.8):

js = Javascript('''

async function takePhoto(quality) {

const div = document.createElement('div');

const capture = document.createElement('button');

capture.textContent = 'Capture';

div.appendChild(capture);

      const video = document.createElement('video');

      video.style.display = 'block';

      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);

      div.appendChild(video);

      video.srcObject = stream;

      await video.play();

      // Resize the output to fit the video element.

      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.

      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');

      canvas.width = video.videoWidth;

      canvas.height = video.videoHeight;

      canvas.getContext('2d').drawImage(video, 0, 0);

      stream.getVideoTracks()[0].stop();

      div.remove();

      return canvas.toDataURL('image/jpeg', quality);

    }

    ''')

  display(js)

  data = eval\_js('takePhoto({})'.format(quality))

  binary = b64decode(data.split(',')[1])

  with open(filename, 'wb') as f:

    f.write(binary)

  return filename

IMPORTING THE IMAGE

from IPython.display import Image

try:

  filename = take\_photo()

  print('Saved to {}'.format(filename))

  # Show the image which was just taken.

  display(Image(filename))

except Exception as err:

  # Errors will be thrown if the user does not have a webcam or if they do not

  # grant the page permission to access it.

  print(str(err))

ROLL NUMBER AUTHENTICATION

#read img

def authenticate(file):

imgUMat = cv2.imread(file)

gray = cv2.cvtColor(imgUMat, cv2.COLOR\_BGR2GRAY)

## (2) Threshold

th, threshed = cv2.threshold(gray, 127, 255, cv2.THRESH\_TRUNC)

## (3) Detect

result = pytesseract.image\_to\_string((threshed), lang="ind")

lst = result.split()

if "Roll" in lst:

ind = lst.index("Roll")

elif "Roli" in lst:

ind = lst.index("Roli")

else:

return “Not Authenticated”

roll\_numbers = ["199X1A0575", "199X1A0123"]

if len(lst[ind + 1]) == 3:

req = lst[ind + 2]

else:

req = lst[ind + 1][3:]

print(req)

if req in roll\_numbers:

return "Authenticated"

return "Not Authenticated"

ANVIL

@anvil.server.callable

def auth():

  file = take\_photo()

  return authenticate(file)

anvil.server.wait\_forever()

**3.2.1 Output Screens**

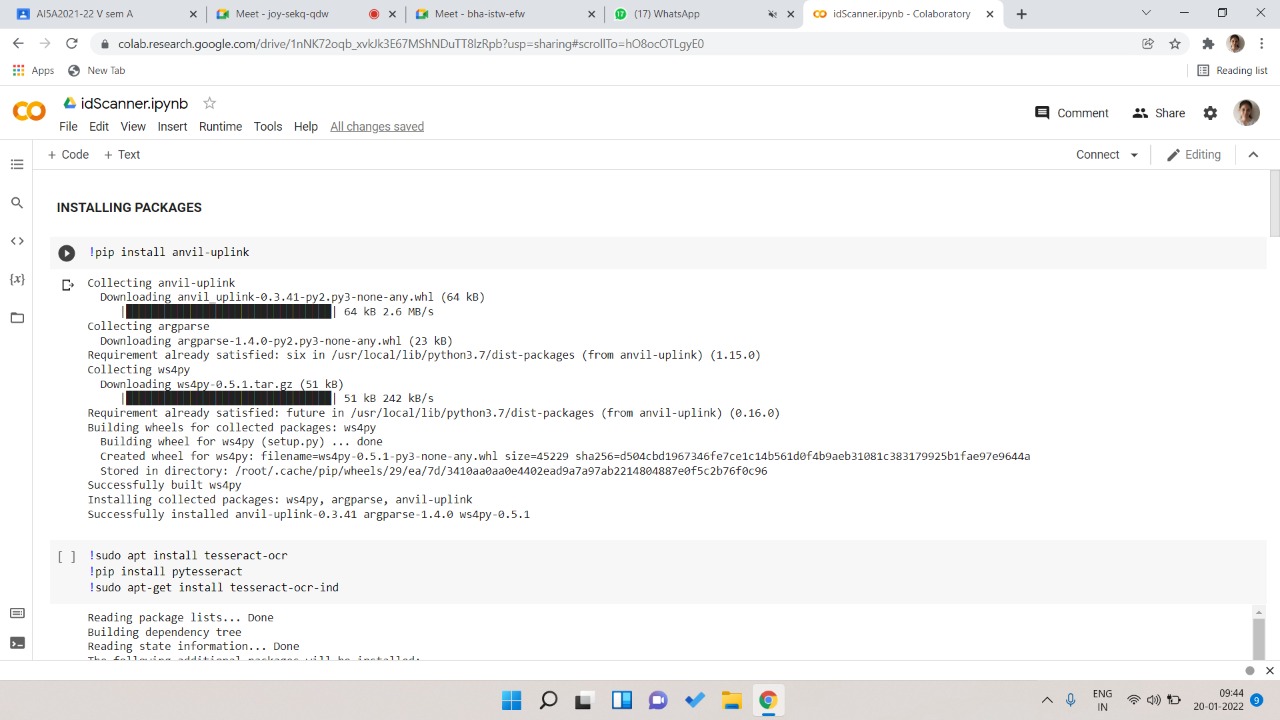


Fig 3.2.1 Installing anvil uplink

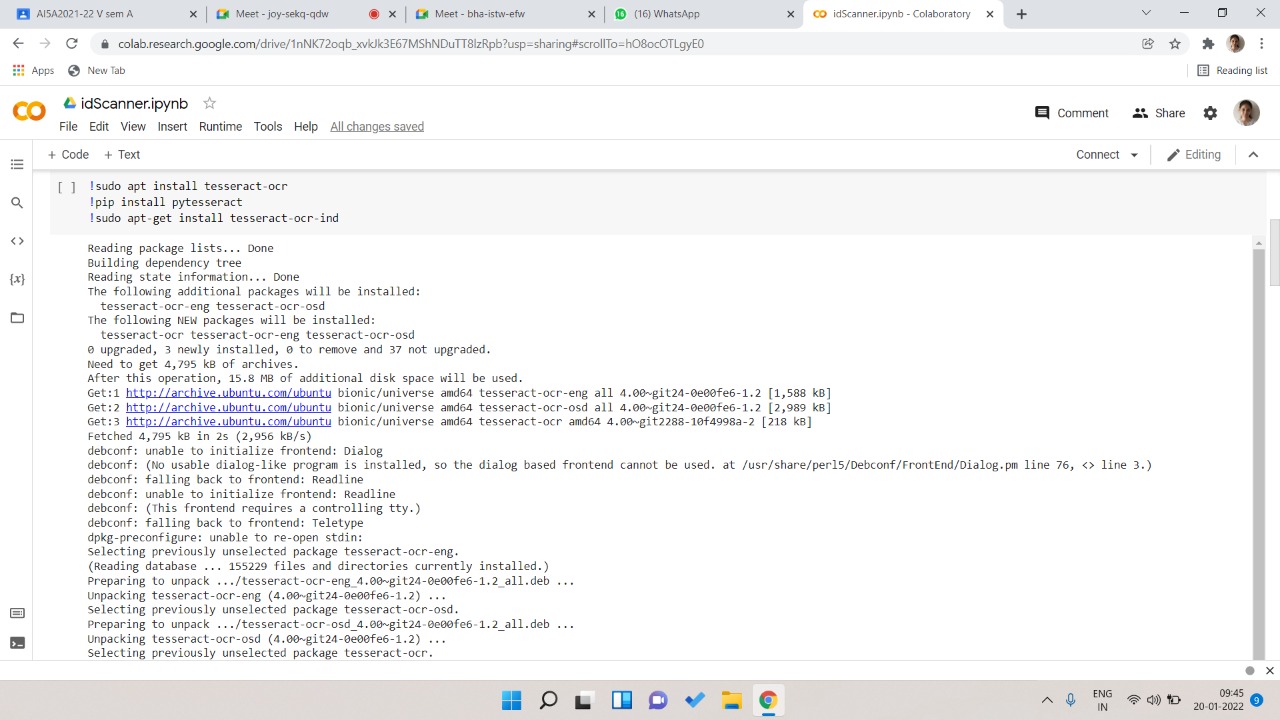


Fig 3.2.2 Installing the required packages

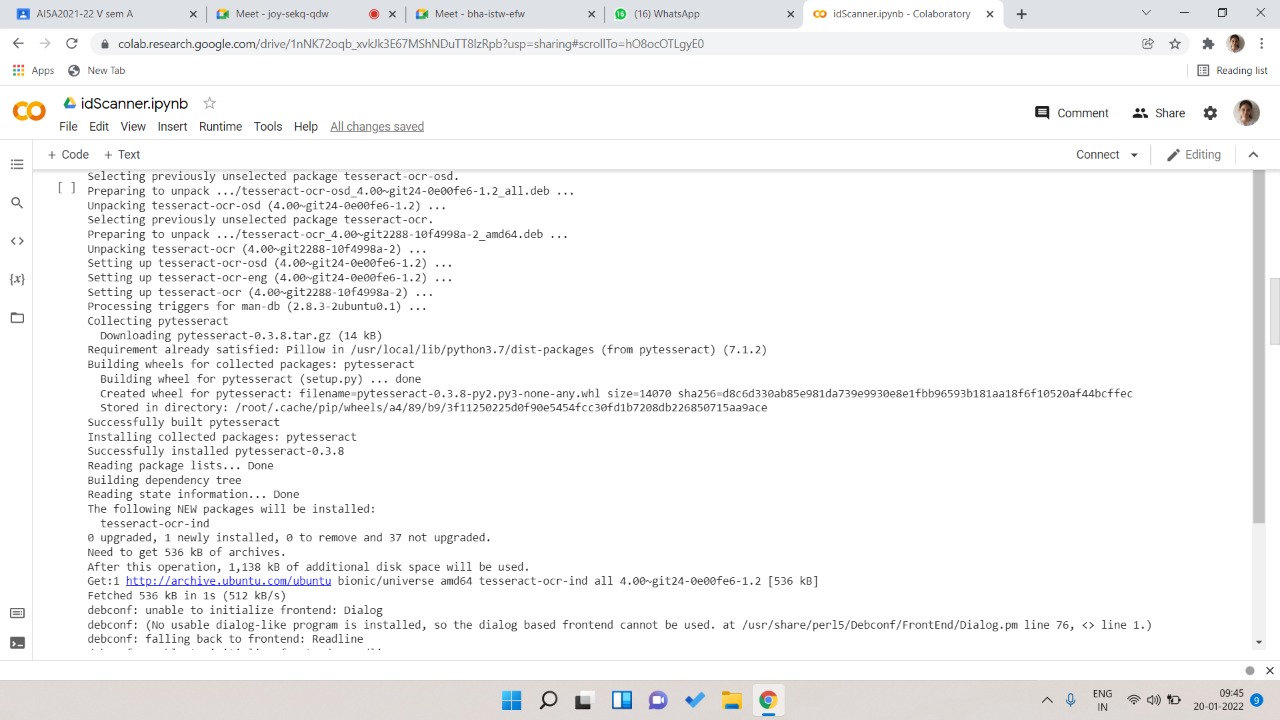


Fig 3.2.3 Installing the requires packages

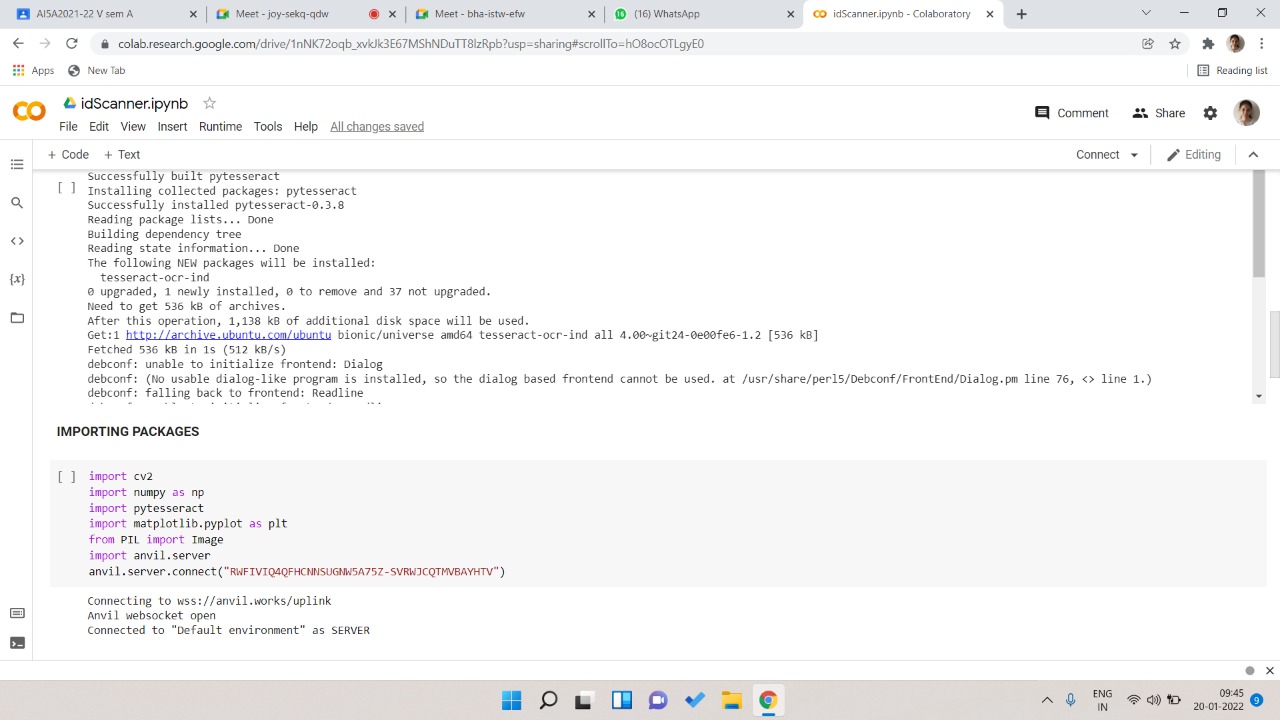


Fig 3.2.4 Importing the required packages and Establishing server connection

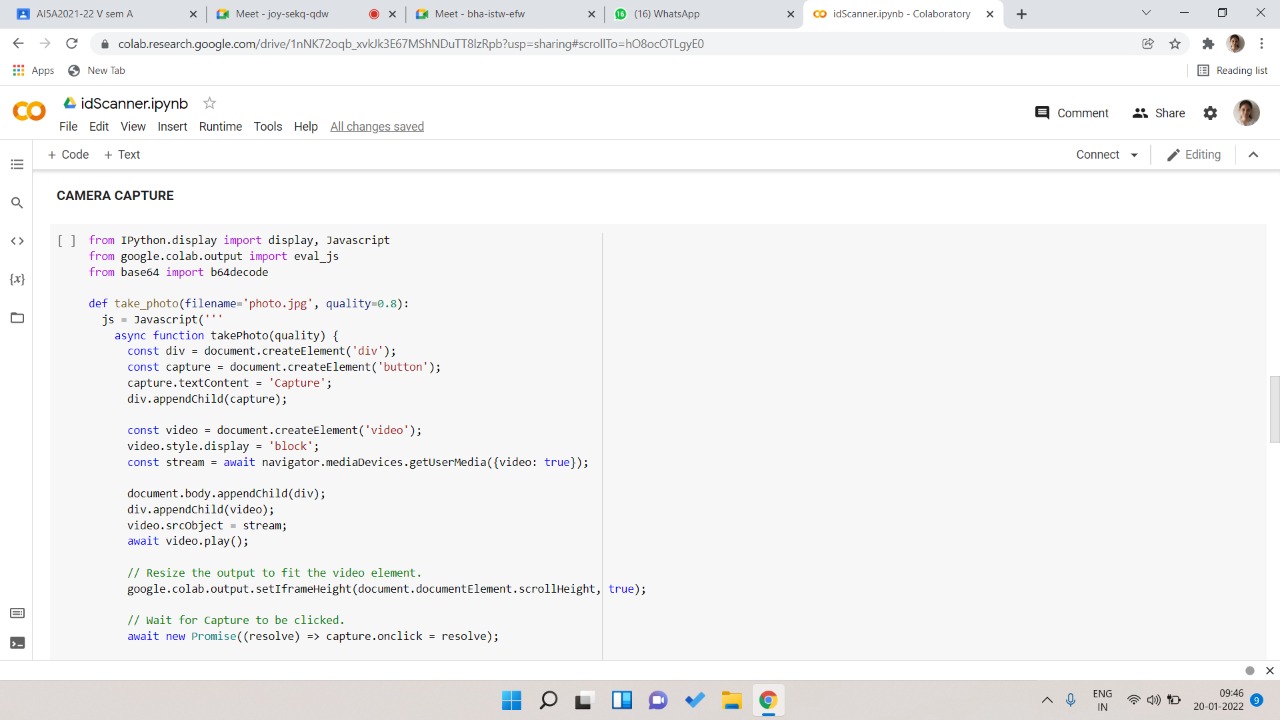




Fig 3.2.5 capturing the image

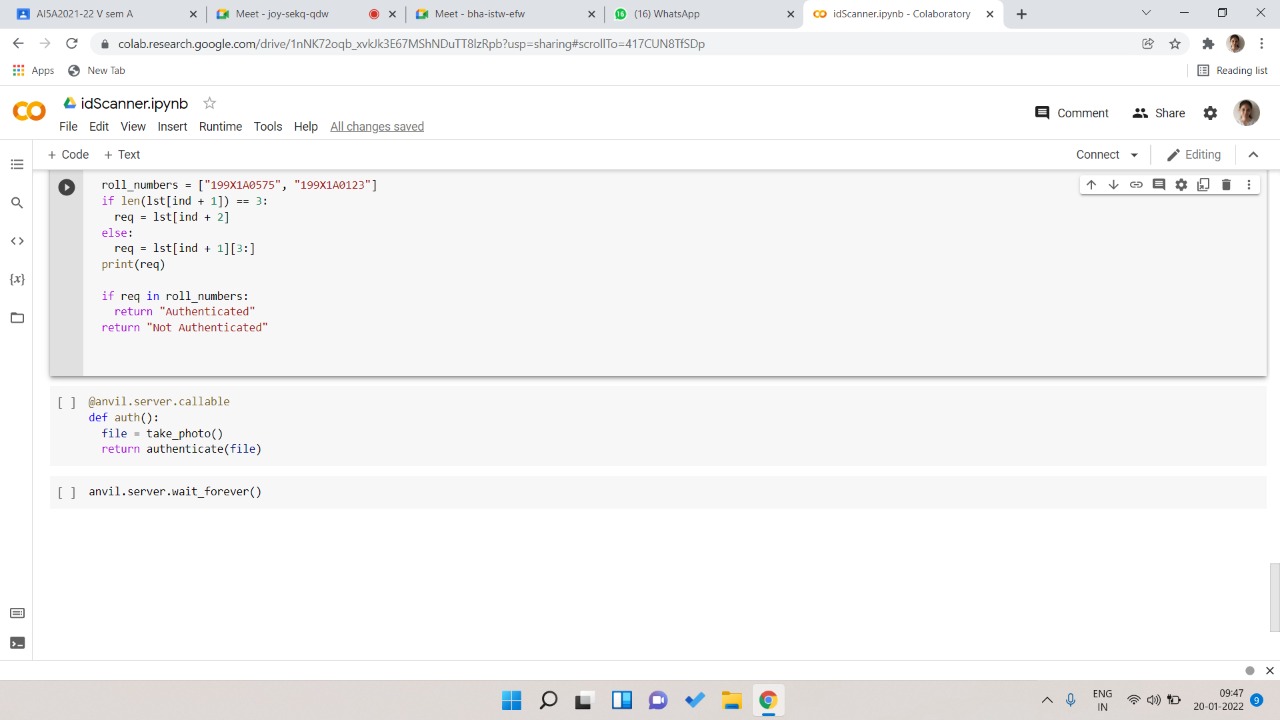
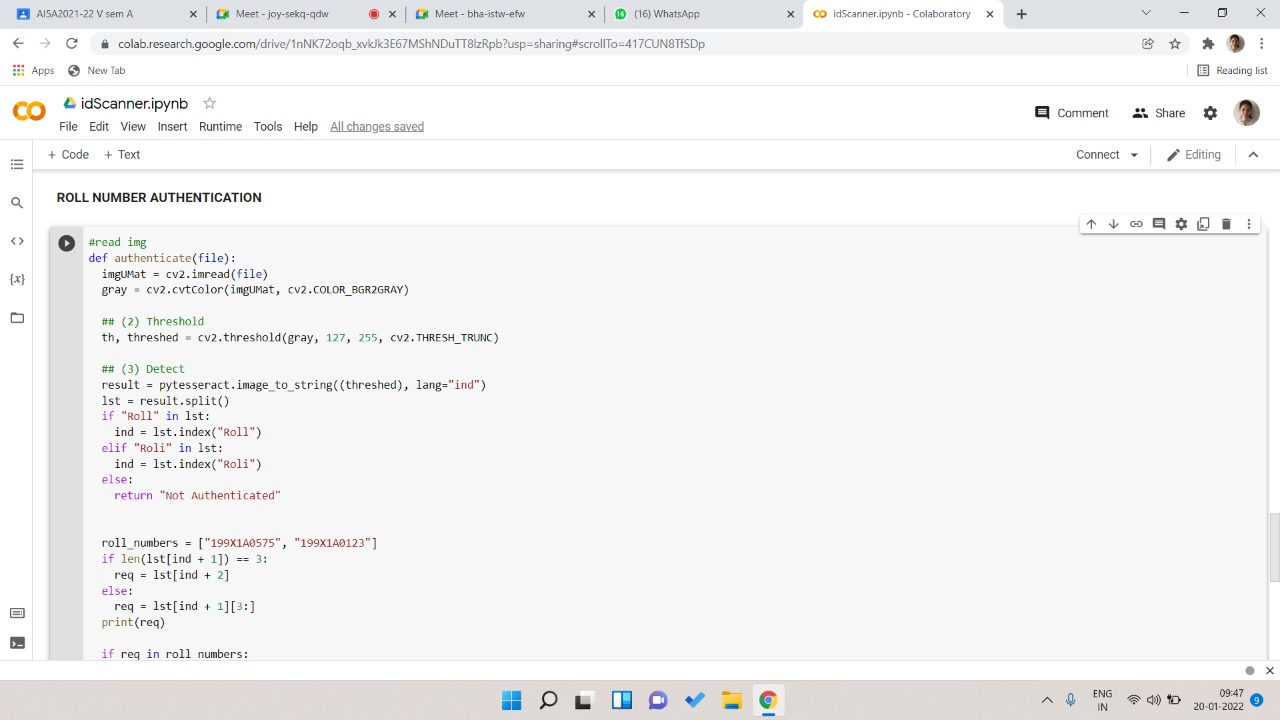


Fig 3.2.6 Authentication of the ID Card

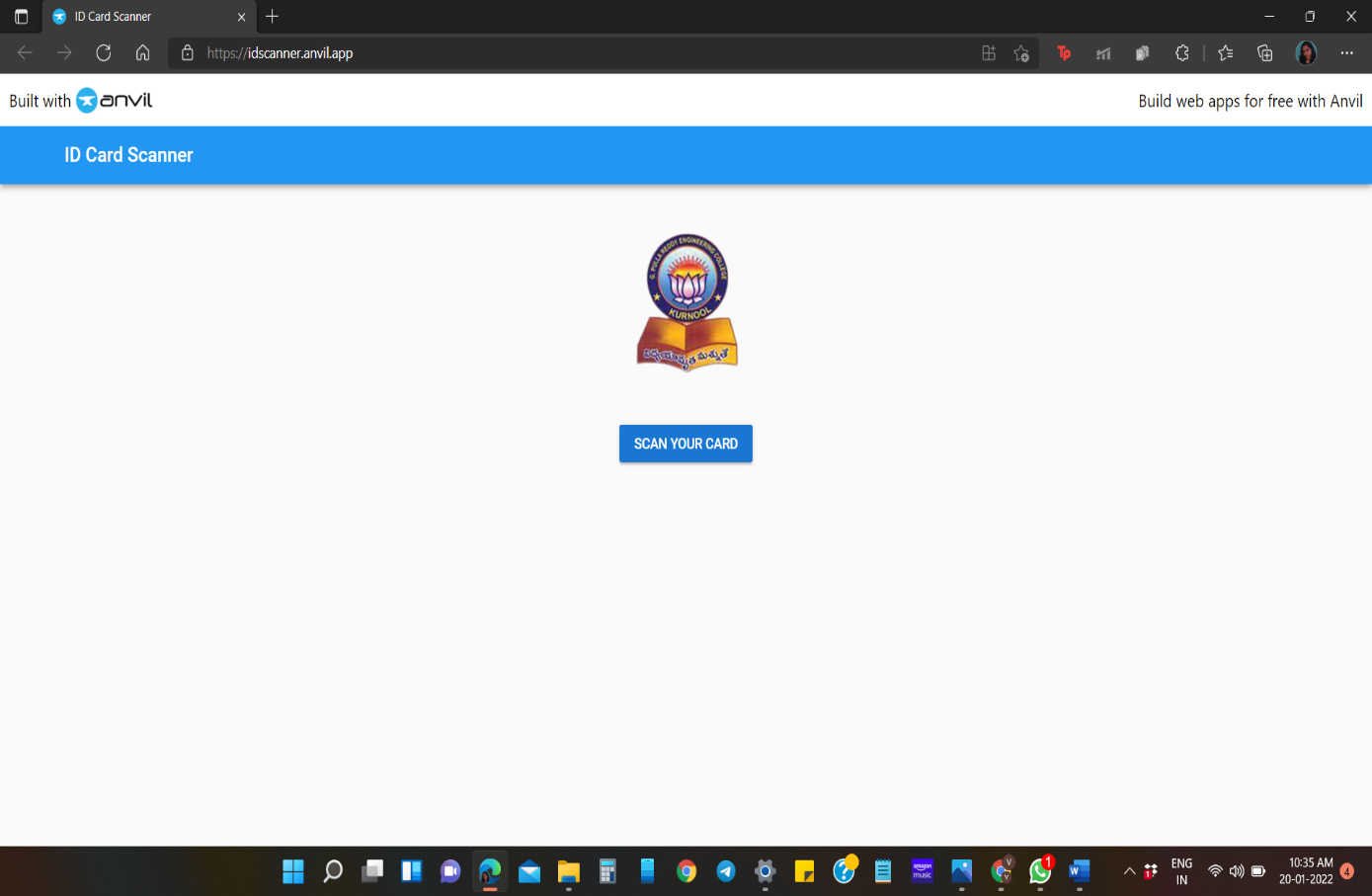


Fig 3.2.7 ID Card Scanner App

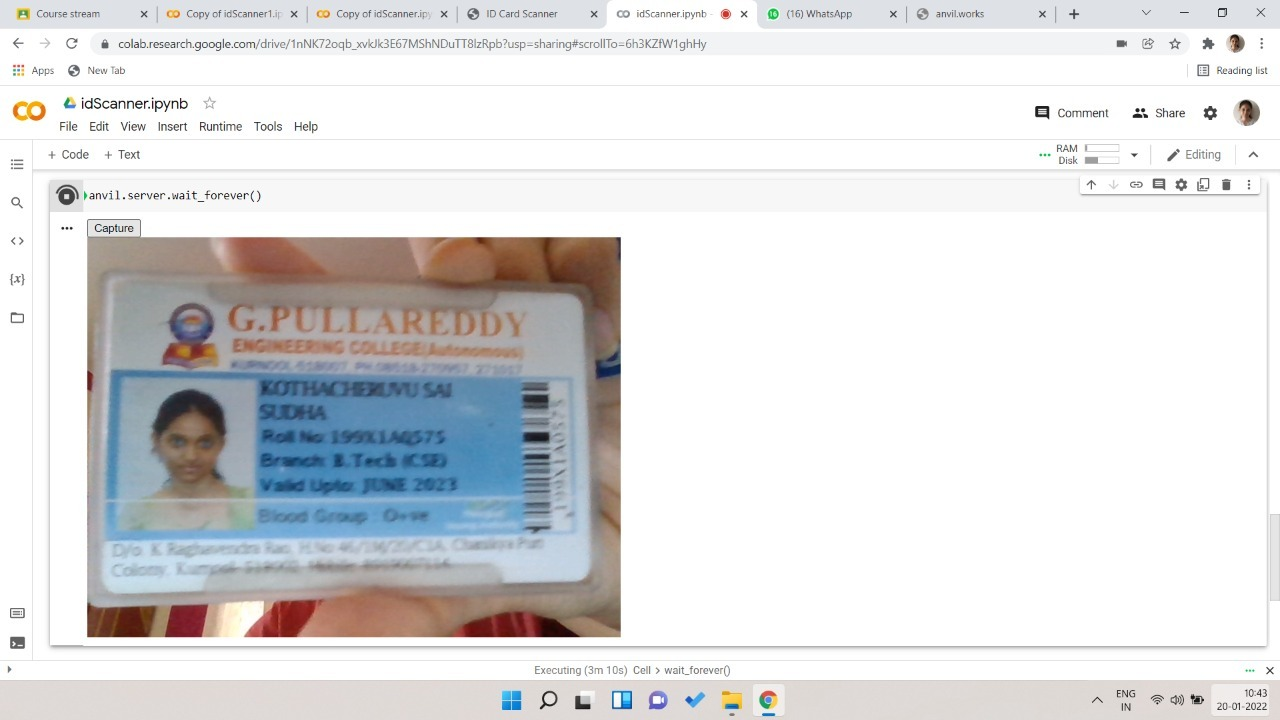


Fig 3.2.8 Student ID card

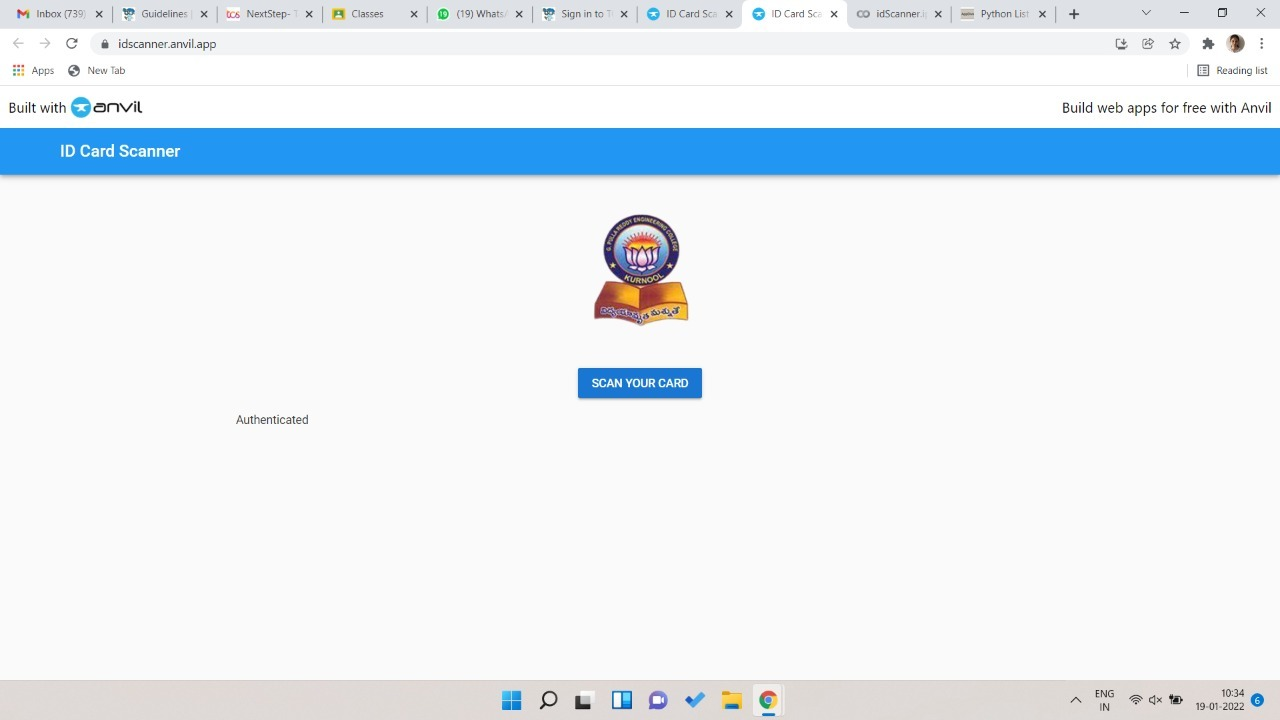


Fig 3.2.9 Final output of the App

**3.3 Testing and Validation**

**Levels of testing**

Testing can be done in different levels of SDLC. They are:

**Unit Testing**

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs the module. The unit test is always white box oriented. In this, different modules are tested against specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Due to its close association with coding, the coding phase is frequently called “coding and unit testing”. The unit test can be conducted in parallel for multiple modules.

**Integration Testing**

The second levels of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integrating testing:

**Top-down Integration**: Top-down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards throw the control hierarchy beginning with the main control module.

**Bottom-up Integration**: bottom-up integration as its name implies, begins construction and testing with automatic modules.

**Regression Testing:** In this contest of an integration test strategy, regression testing is the execution of some subset of test that have already been conducted to ensure that changes have already been conducted to ensure that changes have not propagated unintended side effect.

**Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

System/procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional is complete, additional tests are identified and the effective value of current tests is determined.

**Validation**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers.

Reasonable expectation is defined in the software requirement specification a document that describes all user visible attributes of the software. The specification contains a section title “validation criteria”. Information contained in that section forms the basis for validation testing approach.

**Alpha testing**

It is virtually impossible for a software developer to see how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

**Beta testing**

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

**System Testing and Acceptance Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

**4. CONCLUSION**

**4. CONCLUSION**

An ID Scanner which determines whether a person is authenticated to enter the college or not has been designed. The roll number is extracted from the ID card using Pytesseract OCR engine. It is validated against the roll numbers in the college database. If the roll number is present, the person is allowed to enter the college.

**Future Scope**

Our implementation can be further extended to bar code scanning. This prevents duplication of ID cards thereby, improving the security. It can also be used for collecting attendance in examination halls.

**REFERENCES**

<https://nanonets.com/blog/ocr-with-tesseract/>

<https://opencv24-python-tutorials.readthedocs.io/en/latest/>

<https://anvil.works/blog>

<https://pillow.readthedocs.io/en/stable/reference/index.html>

<https://usefulangle.com/post/352/javascript-capture-image-from-camera>

[1] Mithe R, Indalkar S and Divekar N 2013 Optical Character Recognition Int. J. Recent Technol.

Eng. 2 72–5

[2] Wang X, Ding X and Liu C 2005 Gabor filters-based feature extraction for character recognition

Pattern Recognit. 38 369–79