**Detects and extracts text from any image using Image to Text API**

**NAME**: SANDU JAI VENKATESH

**BRANCH:** COMPUTER SCIENCE OF ENGINEERING AND TECHNOLOGY

**YEAR**: 3rd YEAR

**ROLL NO**: 18R11A05J1

**COLLEGE**: GEETHANJALI COLLEGE OF ENGINEERING AND TECNOLOGY

**TECHNOLOGY**: PYTHON PROGRAMMING

**INTRODUCTION**

Most visiting cards never see the light of day. You leave a tradeshow or a meeting with a pocketful of cards, thinking “I should really do something with these cards at some point….” Your options are bleak: spend a few hours manually inputting them into your database, go through the trouble of hiring someone to do it for you, buy an expensive app to scan the cards (and another one to integrate it with your system of record). Or you can do nothing at all. Leave those cards in your pocket to collect dust.

Despite the growing popularity of email and social media, visiting cards remains the quickest means of exchanging contact information. That says, address books have now gone digital, and no one has time ( or patience)to manually save the visiting card details to smartphones.

Fear not, for visiting card scanner apps are here to help. We can use our smartphone camera to scan visiting cards and automatically save the contact details on them to our smartphones.

**Existing System:**

The existing system is where we save the contact details manually present on the visiting cards. It needs lots of time,and if not we have to carry those visiting cards where ever we go with our pocket full.

**Disadvantages:**

* This is a time taken process.
* Unnecessary storage of visiting cards in pockets.
* There is a chance of loosing the cards which leads to lose of the details.

**Proposed System:**

The proposed system is more robust and can save the contact details present on the visiting card into its database. It recognizes the text and prints the data into it.

**ADVANTAGES:**

* This is a time saving process.
* There is no need to carry visiting cards where ever we go.
* There is no chance of loosing the details.

**LITERATURE SURVEY**

**PROJECT LITERATURE :**

The days had changed and people became busy with there lives and had no time to even save the contact details present on the visiting card manually. So there comes visiting card scanner apps to help us. These apps can be developed using python language. These apps help us to directly scan the cards through the camera and saves the contact details present on the visiting card into its database.

**INTRODUCTION TO PYTHON :**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side)
* software development,
* mathematics,
* system scripting.

### What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

### Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3.However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**PYTHON TECHNOLOGY :**

|  |  |
| --- | --- |
|  | |
|  | |
| [**Paradigm**](https://en.m.wikipedia.org/wiki/Programming_paradigm) | [Multiparadigm](https://en.m.wikipedia.org/wiki/Multi-paradigm_programming_language): [functional](https://en.m.wikipedia.org/wiki/Functional_programming), [imperative](https://en.m.wikipedia.org/wiki/Imperative_programming), [object-oriented](https://en.m.wikipedia.org/wiki/Object-oriented_programming), [structured](https://en.m.wikipedia.org/wiki/Structured_programming), [reflective](https://en.m.wikipedia.org/wiki/Reflective_programming) |
| [**Designed by**](https://en.m.wikipedia.org/wiki/Software_design) | [Guido van Rossum](https://en.m.wikipedia.org/wiki/Guido_van_Rossum) |
| [**Developer**](https://en.m.wikipedia.org/wiki/Software_developer) | [Python Software Foundation](https://en.m.wikipedia.org/wiki/Python_Software_Foundation) |
| **First appeared** | 1990; 30 years ago[[1]](https://en.m.wikipedia.org/wiki/Python_(programming_language)" \l "cite_note-guttag-1) |
|  | |
| [**Stable release**](https://en.m.wikipedia.org/wiki/Software_release_life_cycle) | 3.8.3 / 13 May 2020; 28 days ago[[2]](https://en.m.wikipedia.org/wiki/Python_(programming_language)" \l "cite_note-2) |
| [**Preview release**](https://en.m.wikipedia.org/wiki/Software_release_life_cycle) | 3.9.0b3 / 9 June 2020; 1 day ago[[3]](https://en.m.wikipedia.org/wiki/Python_(programming_language)" \l "cite_note-3) |
| [**Typing discipline**](https://en.m.wikipedia.org/wiki/Type_system) | [Duck](https://en.m.wikipedia.org/wiki/Duck_typing), [dynamic](https://en.m.wikipedia.org/wiki/Dynamic_typing), [gradual](https://en.m.wikipedia.org/wiki/Gradual_typing) (since 3.5)[[4]](https://en.m.wikipedia.org/wiki/Python_(programming_language)#cite_note-4) |
| [**OS**](https://en.m.wikipedia.org/wiki/Operating_system) | [Linux](https://en.m.wikipedia.org/wiki/Linux),macOS, [Windows](https://en.m.wikipedia.org/wiki/Microsoft_Windows) Vista (and newer) and more |
| [**License**](https://en.m.wikipedia.org/wiki/Software_license) | [Python Software Foundation License](https://en.m.wikipedia.org/wiki/Python_Software_Foundation_License) |
| [**Filename extensions**](https://en.m.wikipedia.org/wiki/Filename_extension) | .py, .pyi, .pyc, .pyd, .pyo (prior to 3.5),[[5]](https://en.m.wikipedia.org/wiki/Python_(programming_language)#cite_note-5) .pyw, .pyz (since 3.5)[[6]](https://en.m.wikipedia.org/wiki/Python_(programming_language)#cite_note-6) |
| **Website** | [www.python.org](https://www.python.org/) |

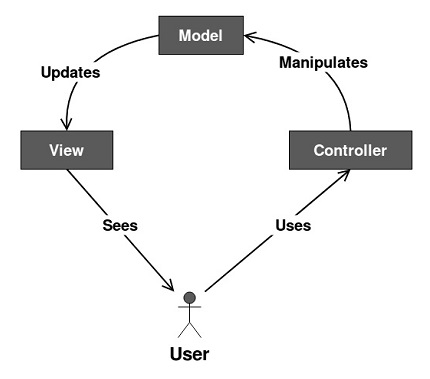
Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. ... Python can be used as the script in Microsoft's Active Server Page (ASP) technology.

Pycharm is one of the widely used Python IDE which was created by Jet Brains. It is one of the best IDE for Python. PyCharm is all a developer's need for productive Python development. With PyCharm, the developers can write a neat and maintainable code.

**MVC ARCHITECTURE:**

Model View Controller is the most commonly used design pattern. Developers find it easy to implement this design pattern.

Following is a basic architecture of the Model View Controller.



Let us now see how the structure works.

### Model

It consists of pure application logic, which interacts with the database. It includes all the information to represent data to the end user.

### View

View represents the HTML files, which interact with the end user. It represents the model’s data to user.

### Controller

It acts as an intermediary between view and model. It listens to the events triggered by view and queries model for the same.

The three components of the MVC pattern are **decoupled** and they are responsible for different things:

* the **Model** manages the data and defines rules and behaviors. It represents the [business logic](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/WIKIPEDIA) of the application. The data can be stored in the Model itself or in a database (only the Model has access to the database).
* the **View** presents the data to the user. A View can be any kind of output representation: a HTML page, a chart, a table, or even a simple text output. A View should never call its own methods; only a Controller should do it.
* the **Controller** accepts user’s inputs and delegates data representation to a View and data handling to a Model.

Since Model, View and Controller are **decoupled**, each one of the three can be extended, modified and replaced without having to rewrite the other two.

**TKinter:**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface.

Python GUI – tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.  
**To create a tkinter app:**

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete [Tcl](https://en.m.wikipedia.org/wiki/Tcl) interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

**LIBRARIES SPECIFIC TO PROJECT:**

**OpenCV:**

Before you can start learning OpenCV you first need to install the OpenCV library on your system. By far the easiest way to install OpenCV is via pip: Install OpenCV the “easy way” using pip.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

## OpenCV-Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

## OpenCV-Python Tutorials

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python. This guide is mainly focused on OpenCV 3.x version (although most of the tutorials will also work with OpenCV 2.x).

OpenCV is used for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.

**OS:**

The OS module in python provides functions for interacting with the operating system. OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

Following are some functions in OS module:

1. **os.name():** This function gives the name of the operating system dependent module imported. The following names have currently been registered: ‘posix’, ‘nt’, ‘os2’, ‘ce’, ‘java’ and ‘riscos’

2. **os.getcwd():**Function os.getcwd(), returns the Current Working Directory(CWD) of the file used to execute the code, can vary from system to system.

3. **os.error():**All functions in this module raise OSError in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system. os.error is an alias for built-in OSError exception.

4. **os.popen():**This method opens a pipe to or from command. The return value can be read or written depending on whether mode is ‘r’ or ‘w’.

5. **os.close():**Close file descriptor fd. A file opened using open(), can be closed by close()only. But file opened through os.popen(), can be closed with close() or os.close(). If we try closing a file opened with open(), using os.close(), Python would throw TypeError.

6. **os.rename():** A file old.txt can be renamed to new.txt, using the function os.rename(). The name of the file changes only if, the file exists and user has sufficient privilege permission to change the file.

**SYSTEM ANALYSIS AND REQUIREMENTS**

**FEASIBILITY STUDY:**

The feasibility of the project is analyzed in this phase and business proposal put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY:**

### This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SOFTWARE AND HARDWARE REQUIREMENTS:**

**SOFTWARE REQUIREMENTS :**

Operating Systems: Any Graphical OS

Language : Python

Data base : SQLite

**HARDWARE REQUIREMENTS :**

Processor : Core 2 duo or higher

Ram : 1GB

Hard disk : 10GB

**PERFORMANCE REQUIREMENTS:**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system.
* The system should be accurate.
* The system should be better than the existing system.

The existing system is completely dependent on the user to perform all the duties.

**SOFTWARE DESIGN**

**INTRODUCTION:**

software design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

**UML DIAGRAMS:**

The **Unified Modeling Language** allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

1.User Model View

1. This view represents the system from the user’s perspective.
2. The analysis representation describes a usage scenario from the end-user’s perspective.

2.Structural model view

1. In this model, the data and functionality are arrived from inside the system.
2. This model view models the static structures.

3.Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

4.Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

5.Environmental Model View

In this the structural and behavioral aspect of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

1. UML Analysis modeling, this focuses on the user model and structural model views of the system.
2. UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**CLASS DIAGRAM:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

A class exists with three sections. In the diagram, classes are represented with boxes which contain three parts:

1. The upper part holds the name of the class
2. The middle part contains the attributes of the class
3. The bottom part gives the methods or operations the

The class can take or undertake.

My App Auth View

Auth View Transfer Control

Detection View Auth Controller

Variables Load( )

Importing Packages Login( )

Run( ) Login Controller( )

Detection( ) Register( )

Auth Controller Register Controller()

Auth Model

Login( )

Register( ) DataBase

Auth Model Insertion( )

Data Base Deletion( )

Create User( ) Updation( )

Get User( ) Select(

**USE CASE DIAGRAM:**

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a [use case](http://en.wikipedia.org/wiki/Use_Case). A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual [use case](http://en.wikipedia.org/wiki/Use_Case) and will often be accompanied by other types of diagrams as well.

Login

Authentication

Authentica

Register

Start

Stop

Detection

Detection

Capture

User

Exit

**SEQUENCE DIAGRAM:**

A sequence diagram is a kind of [interaction diagram](http://en.wikipedia.org/wiki/Interaction_diagram) that shows how processes operate with one another and in what order. It is a construct of a [Message Sequence Chart](http://en.wikipedia.org/wiki/Message_Sequence_Chart). A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical.

User Auth

Detection

Authentication

User

UserName/PassWord

Logged in

Start Camera

Webcam VideoStream

**ACTIVITY DIAGRAM:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential branched or concurrent.

Authentication

No

Login

Login

Yes

Detection App

Registration

Register

Popping the error message

No

Popping the success window

Yes

Pppp.

**MODULE DESCRIPTION**

**LOGIN MODULE:**

The Login Module is a portal module that allows users to type a user name and password to login.You can add this module on any module tab to allow users to log in to the application.

**REGISTER MODULE:**

The register module provides a conceptual framework for entering data on those patients in a way that: eases data entry & accuracy by matching the OpenMRS entry to the data source (usually paper files created at point of care), ties easily back to individual patient records to connect registers to patient data, and collects data elements to enable better supervision of treatment programs.

**DETECTION MODULE:**

The detection module is a module where we can manually detect the text from the visiting card through the webcamera and print the details present on it into its database.

This help us to save the contact details and use whenever needed.

**CODING TEMPLATES/CODE**

**APP CODE:**

Class MyApp:

def run(self):

av=AuthView( )

av. next =self.detection

av. Load( )

def detection(self):

dv=DetectionView( )

dv. load( )

def PrintSomething(self):

Print(“This is my app print function”)

app=Myapp( )

app. run( )

**CONTROLLER CODE:**

Class AuthController:

def login(self,username,password)

If len(username) ==0:

message = “Username cannot be Empty”

return message

If len(username) ==0:

message = “Username cannot be Empty”

return message

am = AuthModel( )

result = am. getUser(username,password)

If result:

message = 1

return message

else:

message = “User not found”

return message

def register(self,name,phone,email,username,password):

If len(name) == 0

message = “Name cannot be empty”

return message

am = AuthModel( )

result = am. createUser(self,name,phone,email,username,password)

If result == 1:

message = ‘user is successfully registered. You

can login now'

else:

message = ‘Some database error.Please kindly

retry'

return message

**MODEL CODE:**

Class AuthModel:

def \_\_init \_\_(self) :

self. conn = connect(‘app. db’)

def getUser(self,username,password):

query\_\_=f“SELECT \* FROM users WHERE username =

‘{username}' and password = ‘{password}’ ”

result = \_\_fetchone(self.conn, query)

print(result)

return result

def create User(self,name,phone,email,username,password):

query = f “ INSERT INTO users (name,phone,email,username,password)

VALUES (‘{name}’,{phone},‘{email}’,‘{username}’,‘{password}’)

try:

insert(self.conn,query)

return 1

except:

print(“Some database error”)

return 0

if \_\_name\_\_ = = ‘\_\_main\_\_’:

am=AuthModel( )

am. createUser(‘Rajesh’,7777777777,‘rajesh@gmail.

com’,‘rajesh’,‘rajesh123’)

**VIEW CODE:**

**AUTHVIEW:**

Class AuthView:

next = None

def load(self):

self. Window = Tk( )

self. window.title(“VISITING CARD

DETECTION APPLICATION”)

tab\_control = try. Notebook(self.window)

self. login\_tab = Frame(tab\_control,

bg = “white” ,padx = 20,pady = 20)

self. register\_tab = Frame(tab\_control,

bg = “white” ,padx = 20,pady = 20)

tab\_control.add(self.login\_tab,text=“login”)

tab\_control.add(self.register\_tab,text=“Register”)

self. Login( )

self. register( )

tab\_control.grid( )

self. window.mainloop( )

def register(self):

window = self. Register\_tab

nl = label(window,text = “Name”)

nl. grid(row=0,column=0)

ne = Entry(window, width=10)

ne. grid(row=0,column=1)

ul = label(window,text = “Username”)

ul. grid(row=1,column=0)

ue = Entry(window, width=10)

ue. grid(row=1,column=1)

pl = label(window,text = “Password”)

pl. grid(row=2,column=0)

pe = Entry(window, show=“\*”,width=10)

pe. grid(row=2,column=1)

el = label(window,text = “Email”)

el. grid(row=3,column=0)

ee = Entry(window, width=10)

ee. grid(row=3,column=1)

phl = label(window,text = “Phone”)

phl. grid(row=4,column=0)

phe = Entry(window, width=10)

phe. grid(row=4,column=1)

b1=Button(window,text =“Register”, command=lambda:self.registerControl(

ne. get( ),phe. get( )we. get( ),ue. get( ),pe. get( )),padx=5,pady=5)

b1. grid(row=5,column=1,pady=5)

def login(self):

window = self.login\_tab

ul = label(window,text = “Username”)

ul. grid(row=0,column=0)

ue = Entry(window, width=10)

ue. grid(row=0,column=1)

pl = label(window,text = “Password”)

pl. grid(row=1,column=0)

pe = Entry(window,show=“\*” ,width=10)

pe. grid(row=1,column=1)

b1=Button(window,text =“Login”, command=lambda:self.loginControl(

ue. get( ),pe. get( )),padx=5,pady=5)

b1. grid(row=2,column=1,pady=5)

def loginControl(self,username,password):

ac=AuthController( )

message = ac.login(username,password)

if message == 1:

self. window.destroy( )

self. next( )

else:

messagebox.showinfo(‘Message’,

message)

def registerController(self,name,phone,email,username,password):

am = AuthController( )

message = ac.register(name,phone,email,username,password)

messagebox.showinfo(‘Message’,message)

if \_\_name\_\_==‘\_\_main\_\_’:

av = AuthView( )

**DETECTION VIEW:**

Class DetectionView:

Stop=false

def load(self):

window =Tk()

window.title(“VISITING CARD DETECTION APP”)

frame=Frame(window,padx=20,pa dy=20,

bg=“#ffeaa7”)

frame. grid(row=0,column=0,padx=10,pady=10)

self.l1=Label(frame)

self. L1. grid(row=1,column=0,column=3)

b1=Button(frame,text=“start”,command=

self.startCamera,pady=20)

b1. grid(row=2,column=0,sticky=‘nesw’)

b2=Button(frame,text=“stop”,command=

self.stopCamera,pady=20)

b2. grid(row=2,column=1,sticky=‘nesw’)

b3=Button(frame,text=“capture”,command=

self.saveImage,pady=20)

b3. grid(row=2,column=2,sticky=‘nesw’)

self.l2=label(frame,text=‘STATUS – CAMERA

STARTED’, font=(“Courier”,35),padx=10,pady=10)

Self.l2.grid(row=3,column=0,column=3,sticky=

‘news’,pady=(20,0))

self. startCamera()

window. mainloop()

def startCamera(self):

self. stop=false

self. cascade=cv2.CascadeClassifier(‘lib/text.xml’)

self. cap=cv2.VideoCapture(0)

t=threading. Thread(target=self.webcam,args=())

t. start()

def webcam(self):

try:

ret,image\_frame=self.cap.read()

image-frame=cv2.resize(image\_frame,None,

fx=1,fy=0.5,interpolation=cv2.INTER\_AREA)

self.img=Image.fromarray(image-frame)

colorimage=cv2.cvtColor(image\_frame,

cv2.COLOR\_BGR2RGB)

grayimage=cv2.cvtColor(image\_frame,

cv2.COLOR\_BGR2GRAY)

self. img=Image.fromarray(colorimage)

img=ImageTK.PhotoImage(self.img)

self. l1.configure(image=img)

self. l1. Image=img

if self. stop == False:

self.l1.after(10,self.webcam)

else:

self. l1. Image=None

except:

print(“Some error”)

def saveImage(self):

if self. stop !=True:

try:

self.image.save(‘images/1.jpg’)

messagebox.showinfo(‘Alert’,”Image saved”)

except:

message box. Showinfo(‘Alert’,” Unable to

save image”)

else:

self. l2.config(text=“Enable the camera to

capture”)

def stopCamera(self):

self. stop=True

**RESULTS AND VALIDATION**



### **SYSTEM TESTING**

**INTRODUCTION:**

 The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS:**

**Unit testing:**

 Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program.  Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at   exposing the problems that arise from the combination of components.

**Functional test:**

 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:

Valid Input             :  identified classes of valid input must be accepted.

Invalid Input              : identified classes of invalid input must be rejected.

Functions                 : identified functions must be exercised.

Output              : identified classes of application outputs must be exercised.

Systems/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 testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing:**

    White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

#### **TEST APPROACH:**

**Testing can be done in two ways:**

1. Bottom up approach

2. Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

**CONCLUSION**

By using this application we can easily save the details and save them for later usage .