BARCODE/QR CODE SCANNER

Team members:

1.G.Chetna Varma

2. Sree VishnuPriya

3. T.Sahiti Sowmya

INDEX :

1.INTRODUCTION

Modern hand held devices such as PCs and PDAs have become increasingly powerful in recent years. Dramatic breakthroughs in processing power along with the number of extra features included in these devices has opened the doors to a wide range of commercial possibilities. However, even with all these added abilities, there are few applications that allow much passing or decryption of environmental information.one such application is barcode scanner. A barcode is a machine-readable strip of data printed in parallel lines, used to represent a multitude of information. Now a days Every type of industry is using the barcode technology because it is much efficient,accurate and secured in storing the real time data.so This application helps the organization to scan a barcode or qrcode of an object and displays the information about that object. It uses webcam video proctoring AI based barcode detection algorithm to detect the object in the frame. Once when the barcode image is captured System will analyze the image and then display the information about that object

1.1 Existing system:

The existing system is a barcode scanner that uses laser or infrared technology to scan a barcode,this may lead to the costing issue where the scanners are very expensive.

Disadvantages:

- It is very expensive
- It is not secured
- It is not user friendly

1.2 Proposed system:

The proposed system uses a web cam video based system for barcode scanning of an object

Advantages:

- Provides better security
- Maintenance of the system is easy and cost effective
- Generates the results quickly
- Provides accurate and efficient data
- User friendly

2.LITERATURE SURVEY

2.1 Project literature:

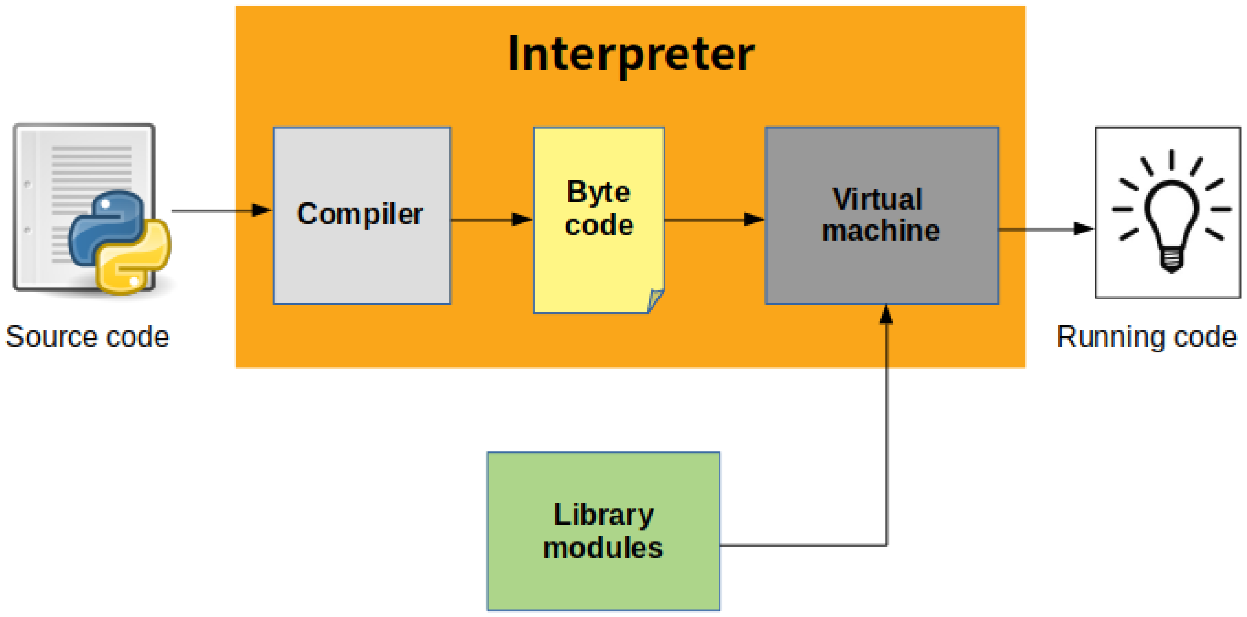
iterature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

2.2 Introduction to python:

2.2.1 Python technology:

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.

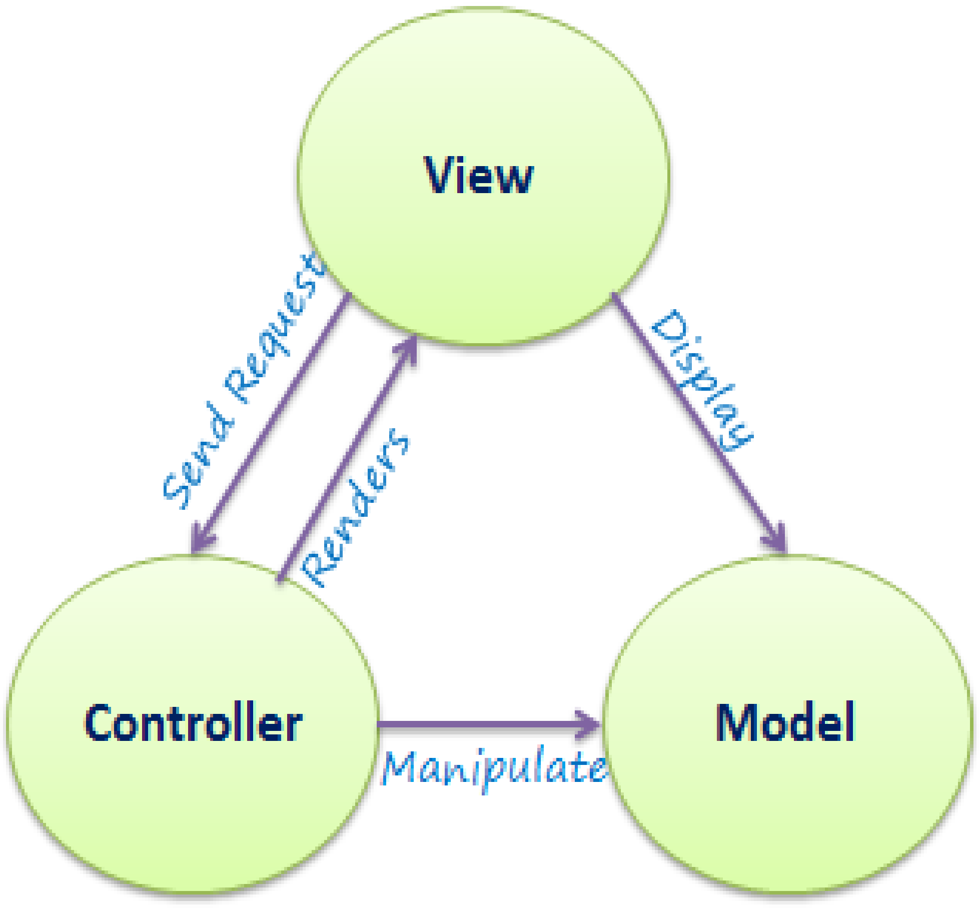
Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code.



The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

2.2.2 MVC Architecture:

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.



2.2.2.1 MVC architecture image

Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View:

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller:

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

2.2.3 Tkinter:

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

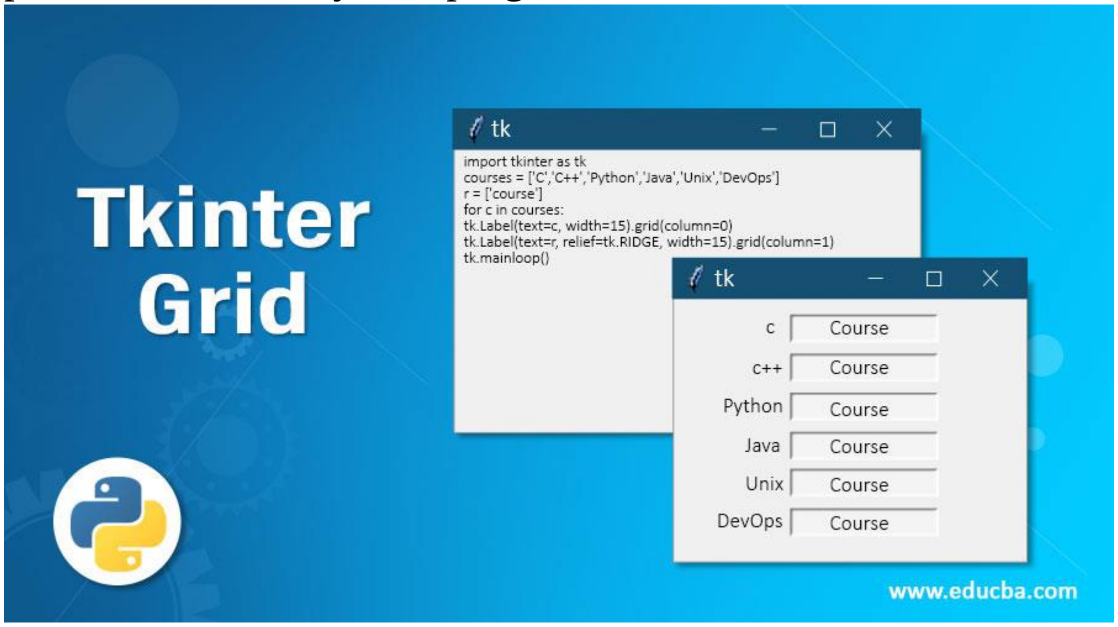
Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.

- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer. .



```
import tkinter as tk
courses = ['C','C++','Python','Java','Unix','DevOps']
r = ['course']
for c in courses:
tk.Label(text=c, width=15).grid(column=0)
tk.Label(text=r, relief=tk.RIDGE, width=15).grid(column=1)
tk.mainloop()
```

```
     c   Course
   c++   Course
Python   Course
  Java   Course
  Unix   Course
DevOps   Course
```

2.2.3.2Tkinter grid image

2.2.4 LIbraries specific to project:

2.2.4.1 Open cv:

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.

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.

2.2.4.2 Pyzbar:

pyzbar is a QR code and Barcode decoder library available in python. It can detect multiple codes from one image and return the code type. OpenCV 4.1 also has QR code detector called QRCodeDetector but it can only detect one QR code from image and barcode is not supported, which is not as powerful as pyzbar.

2.2.4.3 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.

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

3. SYSTEM ANALYSIS AND REQUIREMENTS

3.1 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is 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

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.2 SOFTWARE AND HARDWARE REQUIREMENTS:

3.2.1 Hardware Requirements:

- System : Pentium IV 2.4 GHz or Above
- Hard Disk : 80 GB.
- Web camera : 1080p.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 2 GB

3.2.2 Software Requirements:

- OS : Windows XP Professional/Vista/7/8/8.1 or Linux
- Front End: python
- Tool :pycharm

3.3 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.

4.SOFTWARE DESIGN:

4.1 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.

4.2 UML diagram:

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:

- 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.

- 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.

- 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
- Implementation Model View

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

- Environmental Model View

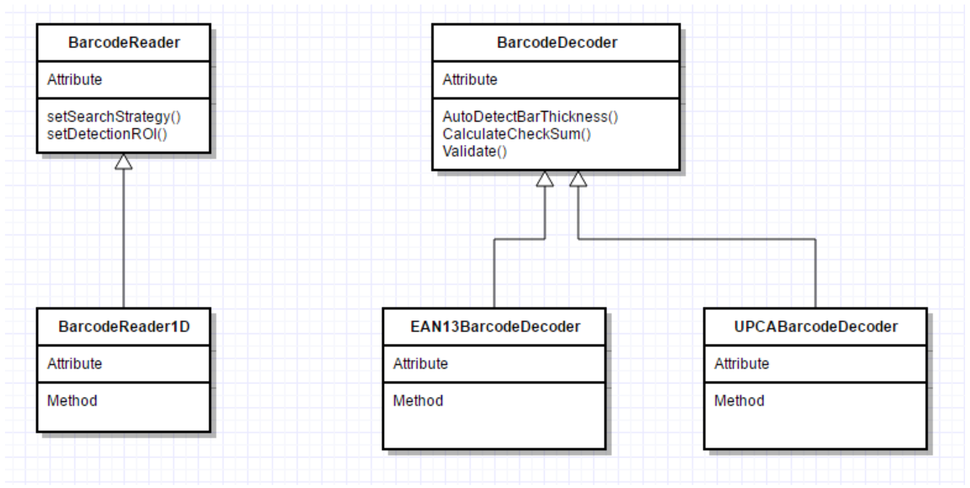
In this the structural and behavioral aspect of the environment in which the system is to be implemented are represented.

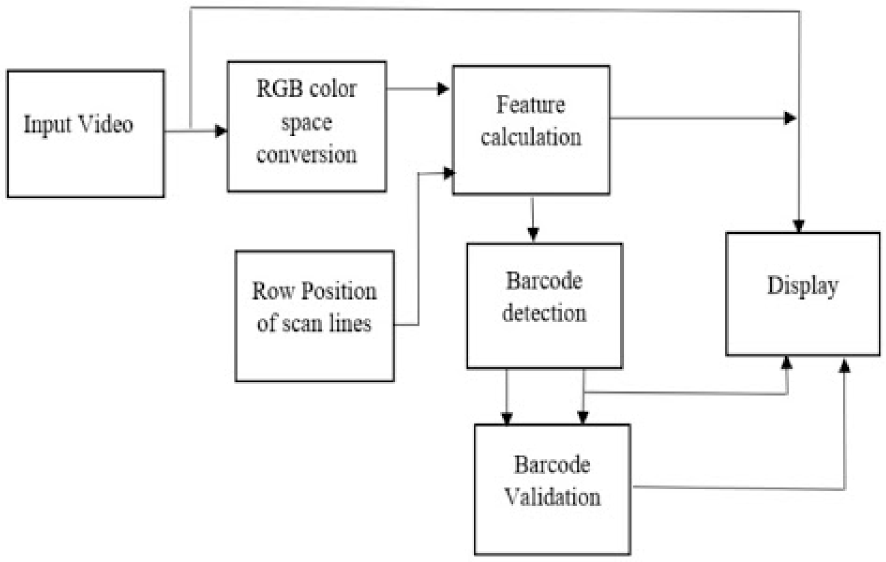
UML is specifically constructed through two different domains they are:

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

4.2.1 Class diagrams:

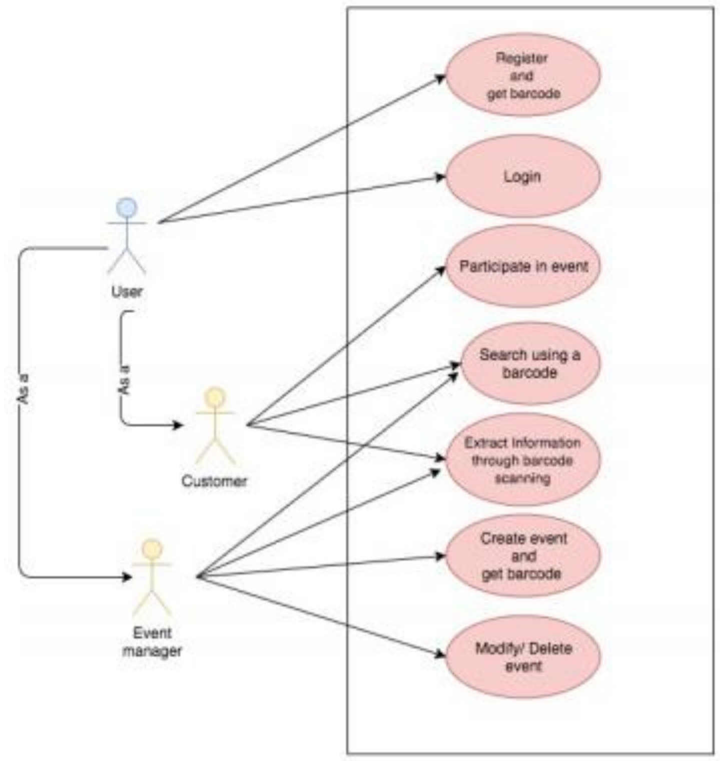
The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class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





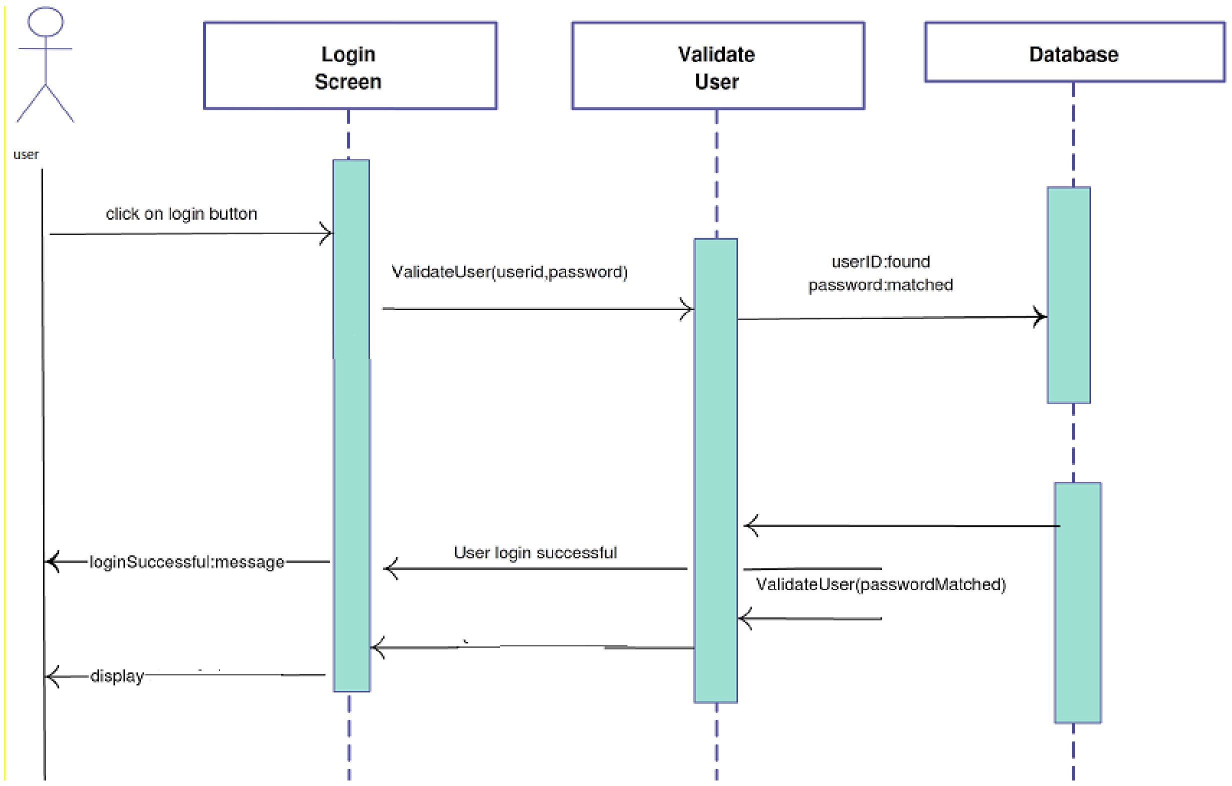
4.2.2 USE CASE DIAGRAM:

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. 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 and will often be accompanied by other types of diagrams as well.

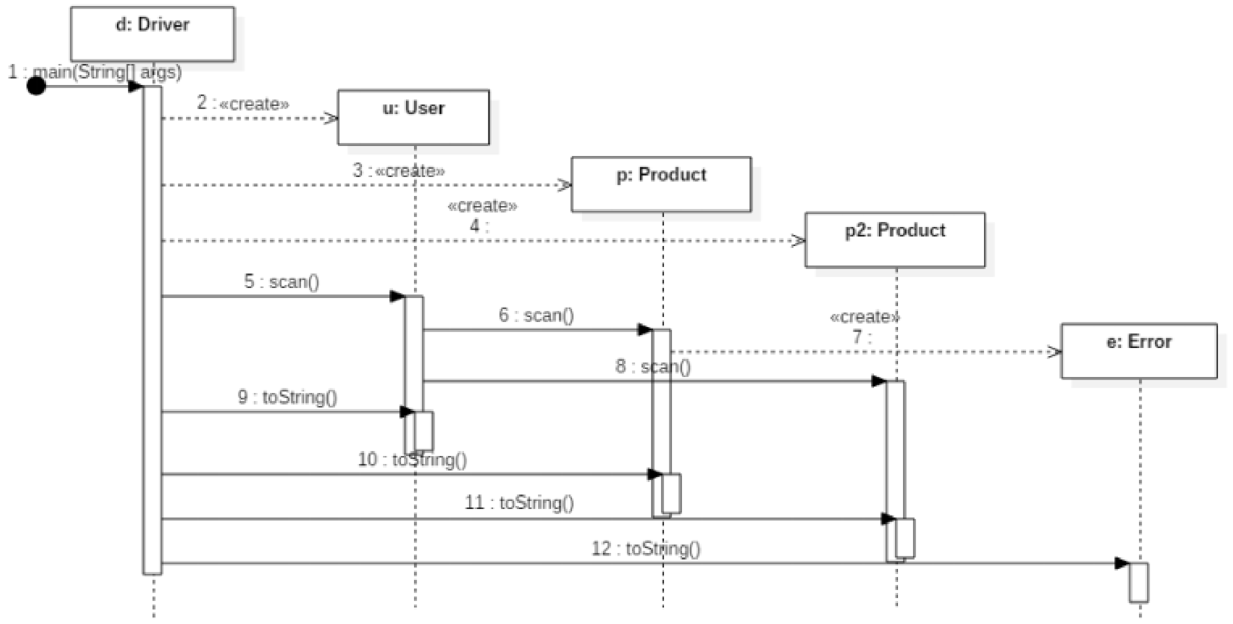


4.2.3 SEQUENCE DIAGRAMS:

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a 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.



4.2.3.1.Login sequence diagram

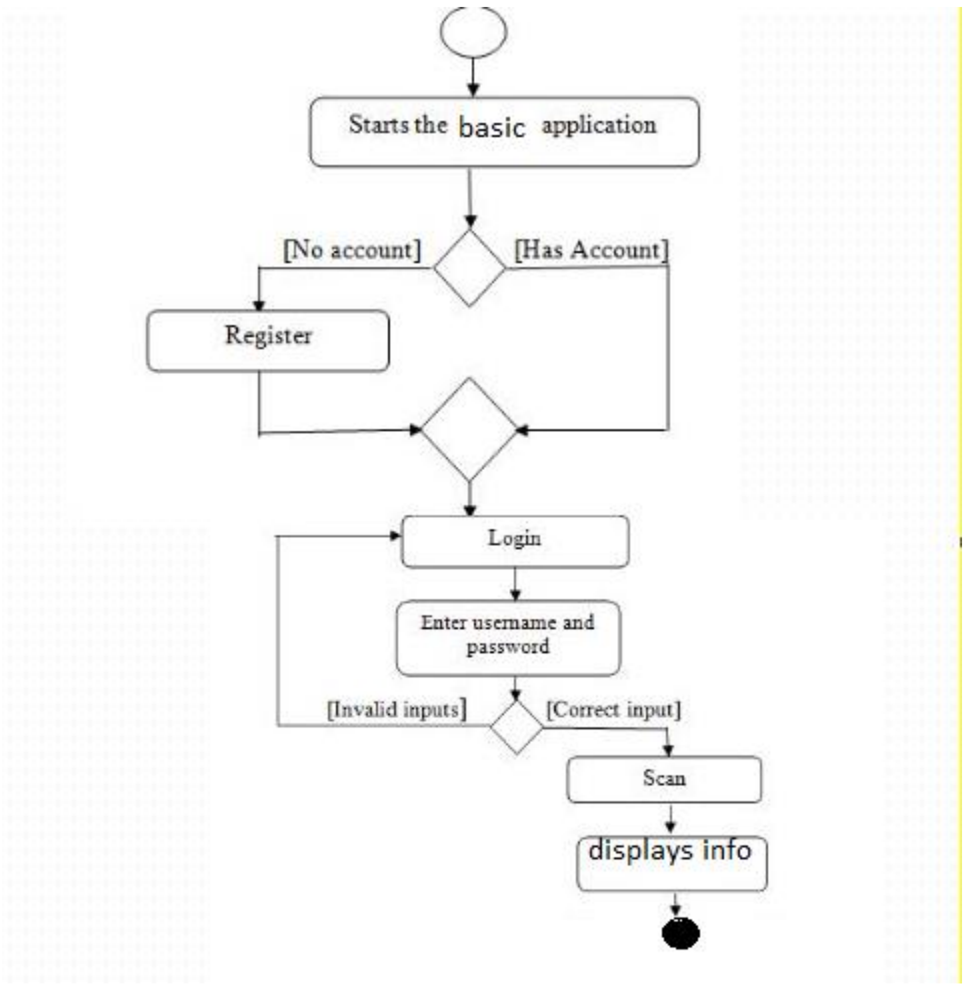


4.2.3.2. Barcode scanner sequence diagram

4.2.4 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.



4.3 MODULE DESCRIPTION:

4.3.1 login module:

The login module is designed in such a way that it requires username and password of the user.If the details entered by the user are valid and correct then it allows the user to access the web camera.

4.3.2 Register module:

The register module is designed in such a way that it requires the basic details of the user like name, phone number,email,username and password. after successful regestration by the user the user can now login the application

4.3.3 Scan module:

The Scan module is designed in such a way that user can scan the object using barcode or QR code and can get the details of the scanned object.

5.CODING TEMPLATES/CODE:

5.1 App code:

```
from views.AuthView import AuthView
from views.BarView import BarView
class MyApp:
    def run(self):
        av = AuthView()
        av.transfer_control = self.Barcode
        av.load()

def Barcode(self):
        bv = BarView()
        bv.load()

app = MyApp()

app.run()
```

5.2 Controller Code:

Bar Controller:

```
from models.BarModel import AuthModel

class AuthController:
    def login(self,username,password):

        if len(username) == 0 :
            message = "Username cannot be empty"
            return message

        if len(password) == 0:
            message = "Password cannot be empty"
            return message

        am = AuthModel()
        result = am.getUser(username,password)
        if result:
            message = 1
        else:
            message = f'User not found'

        return message

    def register(self,name,phone,email,username,password,role):
        am = AuthModel()
        result = am.createUser(name,phone,email,username,password,role)
        if result:
            print("Successfully inserted")
            message = 'Successfully created the user. You can login now'
            return message
        else:
            print("Some problem")
            message = 'There is some issue in storing the data, kindly retry'
            return message
```

5.3 Model Code:

Bar Model:

```
from lib.db import *

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)
        return result
    def createUser(self,name,phone,email,username,password,role):
        query = f"INSERT INTO users (name,phone,email,username,password,role)" \
                f" VALUES
('{name}',{phone},'{email}','{username}','{password}','{role}')"

        result = insert(self.conn,query)
        return result

if __name__ == '__main__':
    am = AuthModel()
```

5.4 View code:

Bar view:

```
from tkinter import *
from tkinter import messagebox
from cv2 import cv2
from PIL import Image           # pip install pillow
from PIL import ImageTk
import threading
from pyzbar import pyzbar  # pip install pyzbar
import time
import os
import sys
sys.path.append('../')
import config



class BarView:
    stop = False

    def load(self):

        window = Tk()
        window.title("Barcode Detector App")
        frame = Frame(window,bg="#63cdda",padx=20,pady=20)
        frame.grid(row=0,column=0,padx=10,pady=10)

        self.l1 = Label(frame)
        self.l1.grid(row=0,column=0,columns=3)

        b1 = Button(frame, text="start",command= self.startCamera,pady=15)
        b1.grid(row=1,column=0,sticky="nesw",pady=10)

        b2 = Button(frame, text="stop", command=self.stopCamera,pady=15)
        b2.grid(row=1, column=1,sticky="nesw",pady=10)
        b3 = Button(frame, text="capture", command=self.captureImage, pady=15)
        b3.grid(row=1, column=2, sticky="nesw", pady=10)

        self.l2 = Label(frame,text="Camera started",font=("Courier", 30),pady=20)
        self.l2.grid(row=2,column=0,columns=3,sticky="nesw")

        self.startCamera()
        window.mainloop()

    def startCamera(self):

        self.stop = False

        # create an instance for video capture via webcam
        self.cap = cv2.VideoCapture(0)

        #  start the process by threading - to run processes parallely
        t = threading.Thread(target=self.webcam,args=())

        t.start()



    def webcam(self):

        # capture each frame (image)
        ret, frame = self.cap.read()
        frame = cv2.resize(frame,None,fx=0.5,fy=0.5,interpolation=cv2.INTER_AREA)
        # change the color to RBG
        colorimg = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        grayimg = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

        # The core functionality
        barcodes = pyzbar.decode(colorimg)
        for barcode in barcodes:
            (x, y, w, h) = barcode.rect
            cv2.rectangle(colorimg, (x, y), (x + w, y + h), (0, 0, 255), 2)
            barcodeData = barcode.data.decode("utf-8")
            barcodeType = barcode.type
            localtime = time.localtime()
            capturetime = time.strftime("%%Y%m%d%H%M%S", localtime)



            text = "{} ({})".format(barcodeData, barcodeType,capturetime)
            cv2.putText(colorimg, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.5, (0, 0, 255), 2)
            # print the barcode type and data to the terminal
            print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData,
capturetime))
```

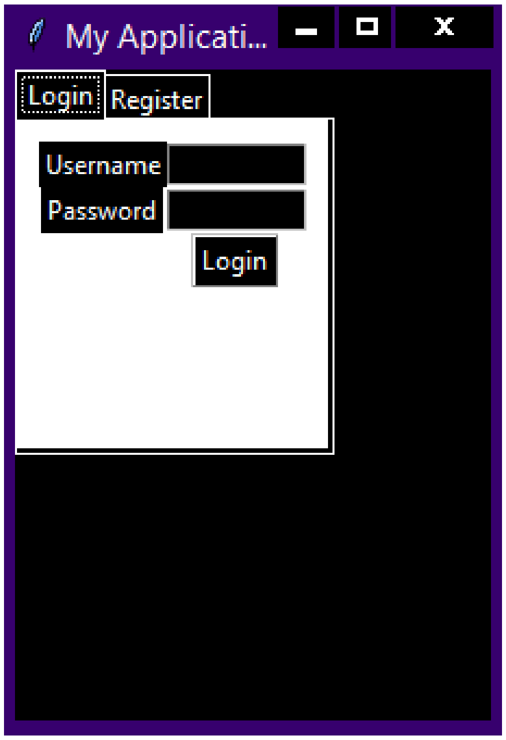
```
        # converting the image to tkinter compatible image
        self.img = Image.fromarray(colorimg)
        imgtk = ImageTk.PhotoImage(self.img)

        # push the image into the tkinter label
        self.l1.config(image=imgtk)
        self.l1.image = imgtk

        # loop this process for every 10ms
        if self.stop == False:
            self.l1.after(10,self.webcam)
        else:
            self.l1.image = None

        localtime = time.localtime()
        capturetime = time.strftime("%Y%m%d%H%M%S", localtime)
        print(capturetime)

    def stopCamera(self):
        self.stop = True

    def captureImage(self):
        image = self.img
        try:
            image.save('images/1.jpg')
            messagebox.showinfo('Alert','Image is saved')
        except:
            messagebox.showinfo('Alert', 'Some error in saving image')
```

Auth view:

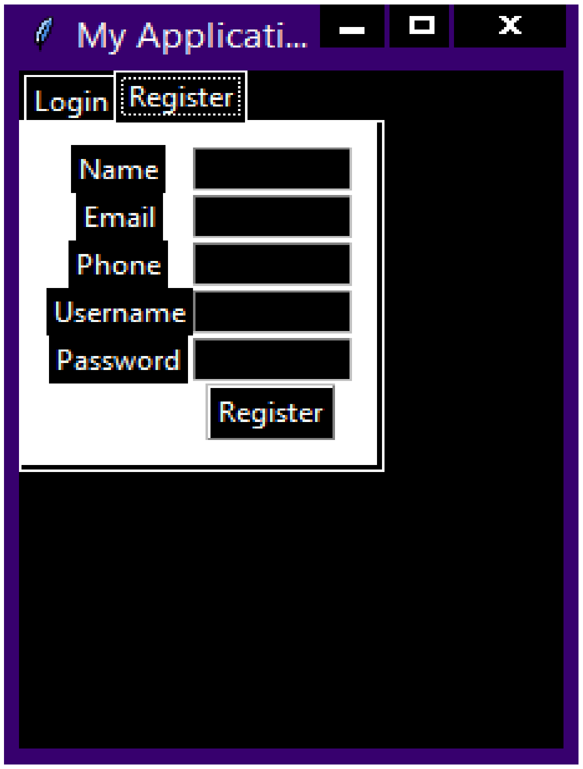
```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from controllers.BarController import AuthController

class AuthView:

    def load(self):
        self.window = Tk()
        self.window.title("My Application")

        tab_control = ttk.Notebook(self.window)

        login_tab = Frame(tab_control,bg="white",padx=10,pady=10)
        register_tab =  Frame(tab_control,bg="white",padx=10,pady=10)
        tab_control.add(login_tab, text="Login")
        tab_control.add(register_tab, text = "Register")

        self.login(login_tab)
        self.register(register_tab)

        tab_control.grid()

        self.window.mainloop()

    def login(self,login_tab):

        window = 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)

        b = Button(window, text="Login",command=lambda:
self.loginControl(ue.get(),pe.get()))
        b.grid(row=2,column=1)

    def loginControl(self,username,password):
        ac = AuthController()
        print('Username',username)
        print('Password',password)
        message = ac.login(username,password)

        if message == 1:
            self.window.destroy()
            self.transfer_control()
        else:
            messagebox.showinfo('Alert',message)

    def register(self,register_tab):

        window = register_tab

        # Create name label and entry
        nl = Label(window, text="Name")
        nl.grid(row=0, column=0)
        ne = Entry(window, width=10)
        ne.grid(row=0, column=1)

        #  create email label and entry
        el = Label(window, text="Email")
        el.grid(row=1, column=0)

        ee = Entry(window, width=10)
        ee.grid(row=1, column=1)

        #  create phone label and entry
        phl = Label(window, text="Phone")
        phl.grid(row=2, column=0)

        phe = Entry(window, width=10)
        phe.grid(row=2, column=1)

        # Create username label and entry
        ul = Label(window, text="Username")
        ul.grid(row=3, column=0)

        ue = Entry(window, width=10)
        ue.grid(row=3, column=1)

        # Create password label and entry
        pl = Label(window, text="Password")
        pl.grid(row=4, column=0)

        pe = Entry(window, show='*', width=10)
        pe.grid(row=4, column=1)

        #  create a button register
        b = Button(window, text="Register", command=lambda:
self.registerControl(ne.get(),phe.get(),ee.get(),

ue.get(),pe.get()))
        b.grid(row=5, column=1)

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

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

        if message:
            messagebox.showinfo('Alert',message)


av = AuthView()
```

6.Results and validation:

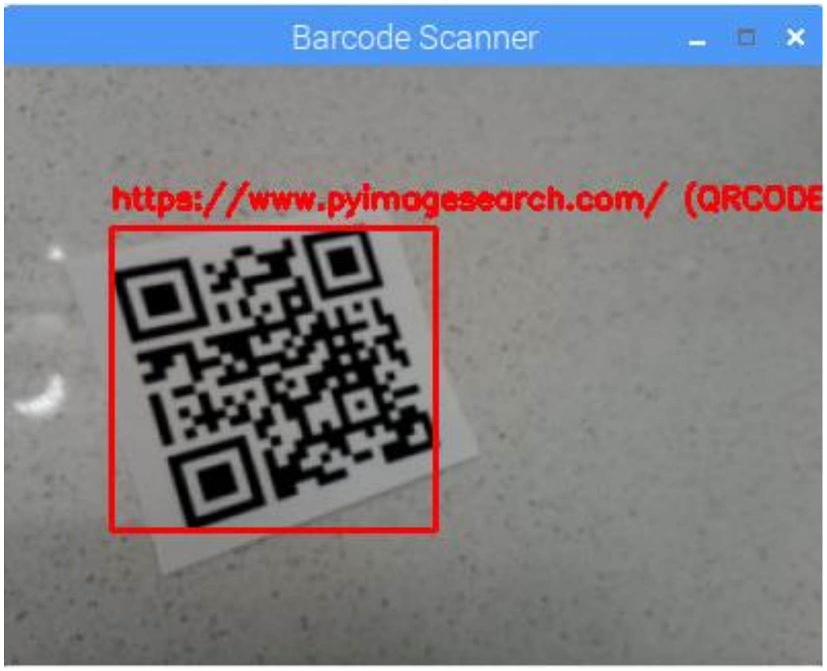
6.1 Output screen



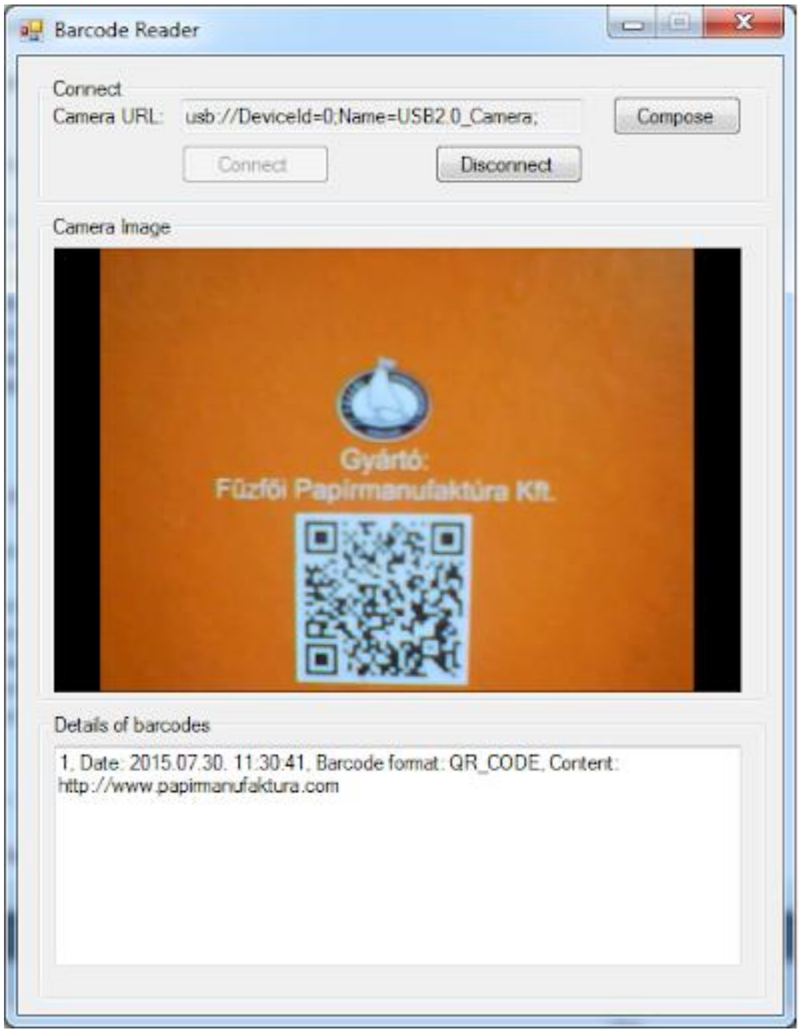
6.1.1Login screen



6.1.2.Register screen



6.1.3.Barcode scan

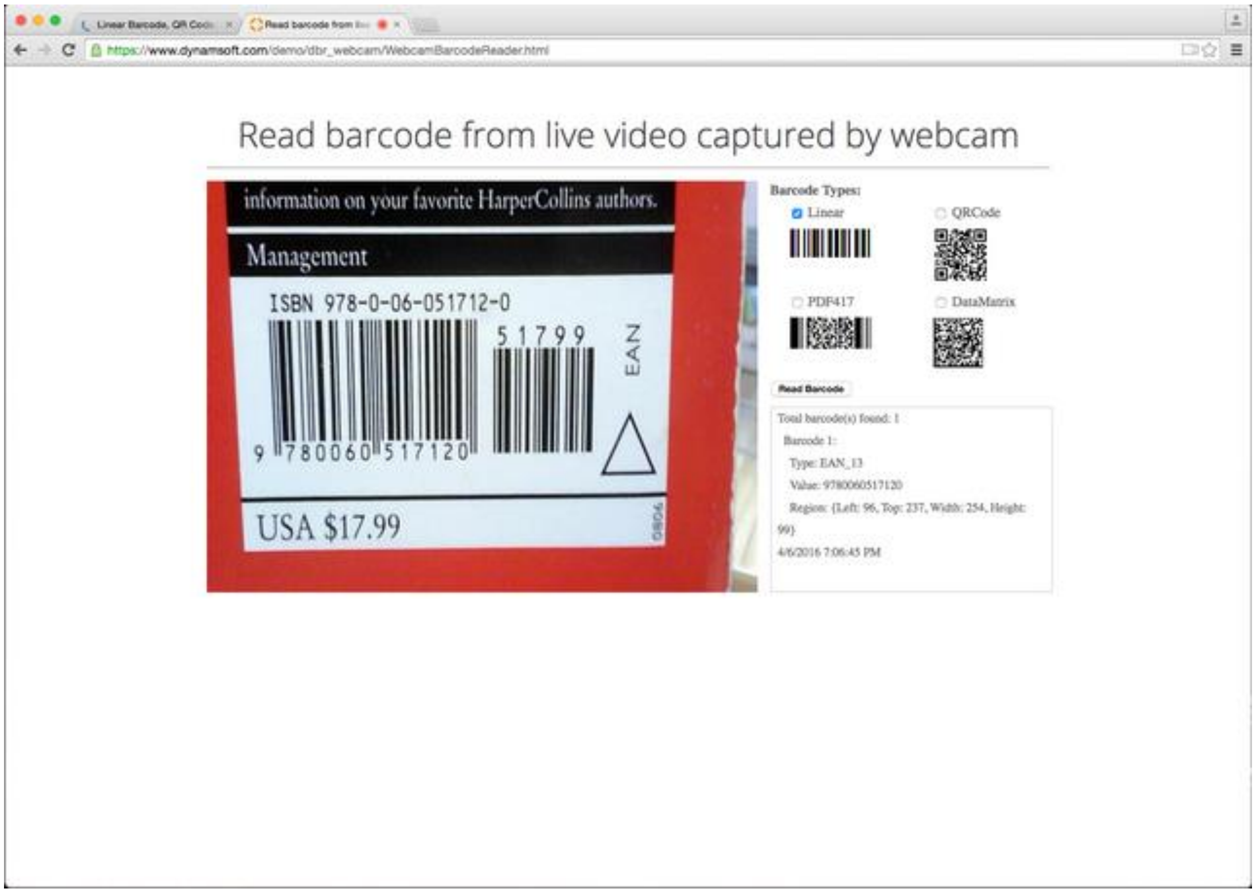


```
1, Date: 2015.07.30. 11:30:41, Barcode format: QR_CODE, Content:
http://www.papirmanufaktura.com
```

6.1.4. qr code scan image

Type : QR-Code
Data : http://LearnOpenCV.com
LearnOpenCV.com
Type : CODE-128
Data : LearnOpenCV.com
BARCODE & QR CODE SCANNER

6.1.5 Bar code and QR code scanner



```
Total barcode(s) found: 1
  Barcode 1:
    Type: EAN_13
    Value: 9780060517120
    Region: {Left: 96, Top: 237, Width: 254, Height:
99}
4/6/2016 7:06:45 PM
```

6.1.6 Scanning of barcode

7.SYSTEM TESTING:

7.1 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.

7.2 TYPES OF TESTS:

7.2.1 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.

7.2.2 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.

7.2.3 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.

7.2.4 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.

7.2.5 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.

7.2.6 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.

7.2.7 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.

7.3 TEST APPROACH:

Testing can be done in two ways:

- Bottom up approach
- 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.

8.CONCLUSION

This paper presents the short review on the techniques of scanning barcode. Advantages and disadvantages of the techniques are focused. It has many good features such as authentication and it is very user friendly. The scanner is very efficient and accurate in scanning the objects. This paper delivers effective review on the barcode scanner.

9.BIBLIOGRAPHY

[vanRossum04] Guido van Rossum and Fred L. Drake, jr.. Copyright © 2004. Python Labs. https://www.python.org/doc/. Python Documentation.

Software Engineering-A Practitioners Approach,6th Edition,Tata McGraw Hill

Python crash course,Eric Matthes 2016 editioncourse,Eric Matthes 2016 edition

Automate the Boring Stuff with Python: Practical Programming for Total Beginners

A byte of Python from swaproop C.H. v1.2

10.REFERENCES

https://www.python.org

https://www.pyimagesearch.com/2018/05/21/an-opencv-barcode-and-qr-code-scanner-with-zbar/

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

https://www.researchgate.net/publication/264623038_Barcode_Recognition_System