iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.35306

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Project Sasha

Aparna Mete Sawant[1], Mehul Latkar[2], Om Deshpande[3], Rutuja Pawar[4], Sara Dixit[5], Ankita Ghosh[6]
[1,2,3,4,5,6]*Department of Information Technology and MCA, Vishwakarma Institute Of Technology, Bibwewadi, Pune 411037, India.*

Abstract: ***This paper presents an application that aids in the early detection and diagnosis of breast cancer in women, efficiently and accurately. Furthermore, the application eliminates the need for direct contact between patient and doctor by providing a virtual platform in the form of a GUI wherein the patient can upload scanned copies of test results as prescribed by an oncologist. The digitization of the registration process is done via face recognition using Haar Cascade. The application in this paper provides a platform for the doctors to- write a new prescription, view appointments, access reports, view the history of every patient; for patients to- book an appointment, view their prescriptions, access reports and review previous appointments; for pharmacists to view the prescription of a particular patient. The link between patients, doctors and pharmacists is highlighted in the proposed application. The latest object detection algorithm YOLOv3 is used for early detection of breast cancer after the image is annotated. After the training and testing, the model gives an accuracy between (75- 80)%.***

Keywords: face detection, breast cancer, API, object detection, dataset.

I. INTRODUCTION

Breast cancer is the second most common cancer in women. Recent studies have indicated that especially for cancer care, mHealth apps have proven beneficial in terms of creating awareness, promoting prevention of cancer and providing support to cancer survivors. A major reason to use such applications is that they could potentially save a lot of lives as they aid in early-stage detection. As a result of an early diagnosis, the patient can be administered treatment sooner. The proposed method secures the framework by providing a safe interface and data privacy. The method employs Haar Cascade for face detection, PyQt5 for the GUI and API, and CNN for the data training. Applications such as these provide a digital solution with image recognition for system login and receipt management. Furthermore, it provides a virtual approach to stay connected with doctors and patients without the requirement of physically being present in the hospital or pharmacy.

II. LITERATURE REVIEW

Antonio J. Colmenarez1 and Thomas S. Huang proposed a model called Face Detection and Recognition [1]. In this model, two of the most important aspects in the framework of face recognition by computer were addressed here: face and facial feature detection, and face recognition. In this approach, the overall face detection, facial feature localization, and face comparison are carried out in a single step. So this was a real-time face recognition system.

R. Padilla, C. F. F. Costa Filho and M. G. F. Costa proposed a paper on the Evaluation of Haar Cascade Classifiers Designed for Face Detection [2]. In this paper, they have mentioned how Haar Cascade is used in face detection & recognition. Also, how it has improved in the field of detection of faces in images, videos. This paper explains the features of the human face that are used by such vision-based automated systems for recognition & detection. The Worldwide breast cancer application by Corrine Ellsworth-Beaumont, focuses on feedback and an ono-to-one collaborative model and provides customized recommendations of screening based on the results, and its collaboration with other applications allows them to reach their target audience and grow exponentially. Provide a unique user-friendly, interactive interface helping them to gain popularity within a shorter duration of time. BELONG Beating Cancer Together, an application that provides free yet best care services directly from experts, medical professionals and also asks question-related about breast cancer. This also provides you with an option to keep your record and share that with your doctor or loved ones. Sign up for trials, access and advice from leading oncologists, researchers & nurses to resolve your query are some of the features of this application. Samar Zuhair Alshawwa, Rasha Assad Assiri proposed a model called mHealth Interventions for Cancer Care and Support: A Systematic Literature Review [3]. The key objective of the proposed application is to review the available literature on mHealth apps for different types of cancer patients and survivors. The vision is to synthesize the outcomes for cancer disease management, right from creating awareness to screening, prevention and treating it. The main outcomes assessed in this proposed method were lifestyle changes (eg, mood, social as well as emotional support, behavioural changes), clinical outcomes and process of care (e.g., counselling, follow-up, personal care, survivorship care). Other outcomes include cost-effectiveness and patient satisfaction. Structured personal appointment interventions showed a lot of improvement in physical activities as well as QoL in breast cancer patients.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

III. ALGORITHMS AND TOOLS USED

A. Haar Cascade for Face Detection

Face detection is used in our daily lives as well, from our smartphones, laptops, tablets. All of these use in-built software for face detection to authenticate the identity of the user. In 2001, Viola and Jones for the first time proposed a successful framework for object detection to detect the face in video footage [4].

Haar Cascade is an algorithm used for the identification of faces in images or real-time videos which uses edge or line detection features proposed by Viola and Jones. In this case, the algorithm is given a lot of positive images and a lot of negative images. The positive images consist of faces and the negative images do not consist of any faces which can be trained.

This algorithm has four features known as “Haar Features”. They are very similar to the convolutional kernel.

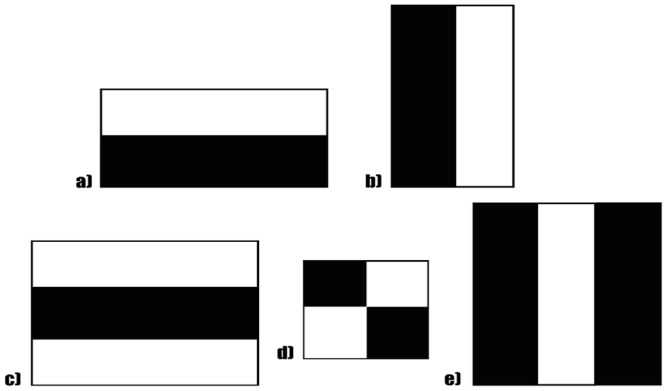


Fig. 1: Haar Features.

There are typically three haar features- Edge Features, Line Features, Four-Rectangle Features. Using these, all the sizes and locations of each kernel are used to calculate a lot of features. So, for all the features, we need to calculate the sum of pixels under white and black rectangles. Haar Cascade detection using OpenCV. OpenCV has many inbuilt pre-trained classifiers for face, eyes & smile. The XML files are stored in folders hence.

B. PyQt5 for GUI

PyQt is one of the most powerful and popular cross-platform GUI library python interfaces for Qt. It was developed by Riverbank Computing Ltd. Today with Python there are so many options to develop GUI applications and one of them is PyQt5. PyQt5 is a cross-platform GUI toolkit, bindings for Qt v5 which is a set of python. An interactive desktop application can be developed today easily because of the tools and simplicity provided by this library.

A GUI application consists of the front-end and back-end. “QtDesigner” , a tool to design the front-end by drag and drop method so that development can become faster and one can give more time on back-end stuff is provided today by PyQt5. A set of modules containing a large number of classes and functions I called PyQt API. The QtCore module contains non-GUI functionality for working the file and the directory also contains all the graphical controls. There are few modules for working with XML(QtXml),SVG(QtSvg) and SQL(QtSql) in addition as well.

C. API calls and Database Connectivity

The system uses SQLite database to store information from the patient and doctor API and Sqlite is the default configuration in Django. For API setup use - django-admin startapp api command, which creates a file tree. Once this is done we can set up our database using: python3 manage.py makemigrations and python3 manage.py migrate command. Creating the user and password. Django provides an in-build model, namely User and Group, to convert SQL records in browser compatible JSON. Now to build the API we used the Django Rest Framework module: pip3 install djangorestframework. The use of serializers help SQL to JSON, this is similar to Marshmallow which is used in Flask for serialization. This .py file is used for data representation. Viewsets are used to manage the CRUD operations in the SQL database to handle as well as accept these requests. They provide an allowance for the single endpoint to handle requests to provide list views of an object and object instance in the database. The urls.py helps to return a specific view for a particular route with the include and path the module django.urls, routers from the Django Rest framework and return views. registering view sets with the router class we could generate URL conf for the API. Finally, along with browsable API include the Login and logout views.setting.py is used to control and check the number of objects returned.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

D. YOLOv3 for Data Training

YOLOv3 is the latest variant of an object detection algorithm YOLO – You Only Look Once. An object detector is a combination of an object recognizer and an object locator. GPU version of Darknet(deep learning framework) along with OpenMP(to enable multiple processors) used for implementation of DNN. We downloaded the model using a script file containing pre-trained network weights and configuration. Reducing the default value of input width and height helps to get a faster result and to increase them gives accurate results. Read and process each frame which needs to be specifically of Blob format.

To train the YOLOv3 for new objects on our breast cancer dataset from Kaggle, we need a large sample set along with test and validation annotations box CSV file and then run and rerun them to get more images. We had split the data into 90-10. Train model with the single class to converge faster. To monitor progress save after 200 iterations for the first 2000 iteration and then save after 1000 iteration each, and recompile the framework.

For data annotation, we label file in <object-class-id> <center-x> <center-y> <width> <height> format object class id in range 0-1,next two by x,y the coordinates of center of bounding box,and followed by width,height of bounding box each of them divided by image width and height of whole image.to leverage learning speed we used a pre-trained model using a transfer learning process where the model contains weights trained on ImageNet. Then configure the model and process the fraction of batch size at one time using the subdivision variable onto your GPU, where we could process (batch/subdivision) number of images anytime but complete iteration is done only after processing defined number(here 64) images. remember to resize the images' width and height to maintain consistency and set the channel to 3 indicating RGB images as input. Reduce weight fluctuation between iteration using momentum parameter. Prevent overfitting using decay parameters. Keep learning rate high then low which is controlled by burn_in parameter/warm-up period. Data augmentation is done for data transformation. Set the number of iterations relatively higher. Finally, train and then test the model for future predictions.

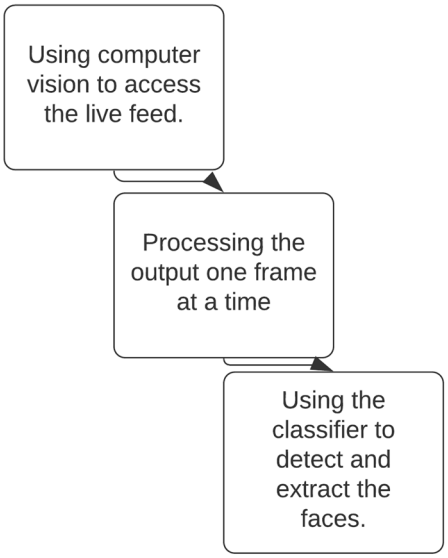
As we know the traditional computer vision methods use a sliding window for object detection at different scales and locations, but this comes with a disadvantage of expensive computational operation also the aspect ratio is assumed to be fixed. The selective searching method in R-CNN and Fast R-CNN is used for narrowing down the number of bound boxes for algorithm testing. On the other hand, YOLO follows a completely different algorithm, by forwarding the whole image only once through the network. In comparison to SSD- an algorithm for object detection where image forwarding is done once through a deep learning network, YOLOv3 is faster where SSD is relatively more accurate. Also, YOLO gives faster results in real-time on TitanX or 1080 Ti GPUs.

IV. WORKING & CODE USED

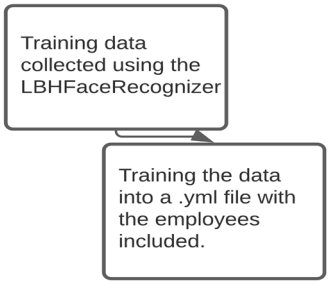
A. Face Detection for login and sign-up of user

In this project, face detection is used for signing in and logging in to the users. Here, the users are patients (or we can call them our major audience), doctors and pharmacists. Here we have used the Haar Cascade Algorithm in OpenCV.

1) Algorithm for Dataset creation using Haar-Cascade classifier.



2) Algorithm to add a user to the system



3) Code used for Face Detection

```
    for (x,y,w,h) in faces:

        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
        cv2.imwrite("dataset_patient/User." + str(face_id) + '.' + str(count) + ".jpg", img[y:y+h,x:x+w]

        cv2.imshow('image', img)

    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif count >= samples:
         break

cam.release()
cv2.destroyAllWindows()

import numpy as np
from PIL import Image
path = 'dataset_patient'

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

def getImagesAndLabels(path):
    import numpy as np
    from PIL import Image

    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []

    for imagePath in imagePaths:

        PIL_img = Image.open(imagePath).convert('L')
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
```

Fig. 2: Code snippet of face detection.

```
font = cv2.FONT_HERSHEY_SIMPLEX

frame_id = 0
with open('Patient.csv', newline='') as f:
    reader = csv.reader(f)
    name_list = list(reader)

a=[]
for i in range(0,len(name_list)):
    if i%2 == 0:
        a.append(name_list[i])
print (a)

names = ['None']
for i in range(0,len(a)):
    names.append(a[i][0])

#names = name_list[0:-1]

cam = cv2.VideoCapture(0)
cam.set(3, 640)
cam.set(4, 480)

minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:

    ret, img = cam.read()

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
    )

    for(x,y,w,h) in faces:
        frame_id=frame_id+1
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
```

Fig. 3:Code snippet for face detection.

B. APIs for the project

1) Code used for API

```
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'HACK_apis.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Fig. 4: Code snippet for API.

2) API created

Fig. 5: API created at localhost 8000.

3) Adding data

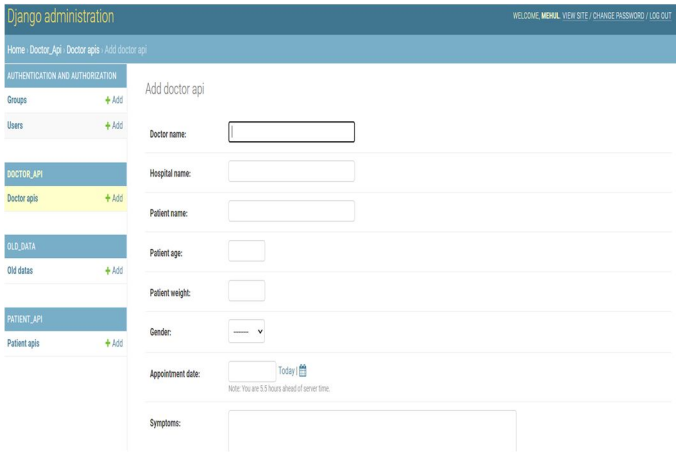


Fig. 6: Adding data in the API

C. GUI for the project

In this project, we have used PyQt5 to create a GUI for our software. There

```
def retranslateUi(self, main_window):
    _translate = QtCore.QCoreApplication.translate
    main_window.setWindowTitle(_translate("main_window", "Dialog"))
    self.pushButton_2.setText(_translate("main_window", "Doctor Sign-up"))
    self.pushButton.setText(_translate("main_window", "Doctor Login"))
    self.pushButton_4.setText(_translate("main_window", "Patient Sign-up"))
    self.pushButton_3.setText(_translate("main_window", "Patient Login"))
    self.pushButton_5.setText(_translate("main_window", "Pharmacist Sign-up"))
    self.pushButton_6.setText(_translate("main_window", "Pharmacist Login"))
    self.pushButton_2.clicked.connect(self.on_click1)
    self.pushButton.clicked.connect(self.on_click2)
    self.pushButton_3.clicked.connect(self.on_click3)
    self.pushButton_4.clicked.connect(self.on_click4)
    self.pushButton_5.clicked.connect(self.on_click6)
    self.pushButton_6.clicked.connect(self.on_click5)

#doctor
def on_click1(self):
    import os
    os.system("start cmd /c python doctor_sign.py")

def on_click2(self):
    import os
    os.system("start cmd /c python doctor_login.py")

#patient
def on_click3(self):
    import os
    os.system("start cmd /c python patient_login.py")

def on_click4(self):
    import os
    os.system("start cmd /c python patient_sign.py")

#pharma
def on_click5(self):
    import os
    os.system("start cmd /c python pharma_login.py")
```

Fig. 7: Code snippet for GUI used in our project.

D. Data Training using YOLOv3

Till now we have trained a dataset of 640 images for this project using YOLOv3. We initially annotated the dataset for 640 images. Data Annotation is the process of adding metadata to a dataset. To train the dataset of breast cancer, we need to use clean, annotated data. We have annotated the dataset using labelImg.

1) Data Annotation

```
self.dock = QDockWidget(get_str('boxLabelText'), self)
self.dock.setObjectName(get_str('labels'))
self.dock.setWidget(label_list_container)

self.file_list_widget = QListWidget()
self.file_list_widget.itemDoubleClicked.connect(self.file_item_double_clicked)
file_list_layout = QVBoxLayout()
file_list_layout.setContentsMargins(0, 0, 0, 0)
file_list_layout.addWidget(self.file_list_widget)
file_list_container = QWidget()
file_list_container.setLayout(file_list_layout)
self.file_dock = QDockWidget(get_str('fileList'), self)
self.file_dock.setObjectName(get_str('files'))
self.file_dock.setWidget(file_list_container)

self.zoom_widget = ZoomWidget()
self.color_dialog = ColorDialog(parent=self)

self.canvas = Canvas(parent=self)
self.canvas.zoomRequest.connect(self.zoom_request)
self.canvas.set_drawing_shape_to_square(settings.get(SETTING_DRAW_SQUARE, False))

scroll = QScrollArea()
scroll.setWidget(self.canvas)
scroll.setWidgetResizable(True)
self.scroll_bars = {
    Qt.Vertical: scroll.verticalScrollBar(),
    Qt.Horizontal: scroll.horizontalScrollBar()
}
```

Fig. 8: Data Annotation

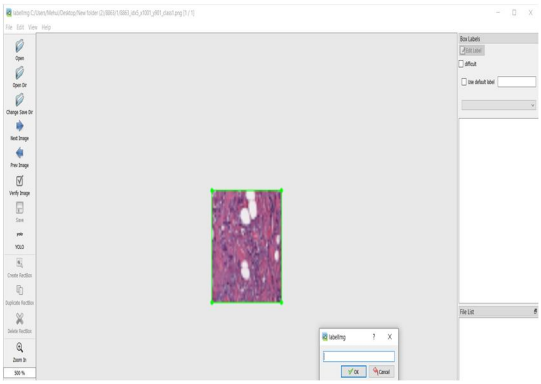


Fig. 9: Data Annotation

2) Annotated files created

Fig. 10: Annotated Files

3) Data Training

```
elif add[0]=="4":
    final.append([570,"5"])
    frame=cv2.imread("4_yolo_out_py.jpg")
    cv2.putText(frame, final[-1][1], (570, 665-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    cv2.imwrite("output/OP_4.jpg",frame)
    cv2.imshow("Output Frame",frame)
    cv2.waitKey()
    cv2.destroyAllWindows()
elif add[0]=="60":
    final.append([336,"3"])
    frame=cv2.imread("60_yolo_out_py.jpg")
    cv2.putText(frame, final[-1][1], (336, 526-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    cv2.imwrite("output/OP_60.jpg",frame)
    cv2.imshow("Output Frame",frame)
    cv2.waitKey()
    cv2.destroyAllWindows()
elif add[0]=="25":
    final.append([585,"4"])
    frame=cv2.imread("25_yolo_out_py.jpg")
    cv2.putText(frame, final[-1][1], (585, 681-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    cv2.imwrite("output/OP_25.jpg",frame)
    cv2.imshow("Output Frame",frame)
    cv2.waitKey()
    cv2.destroyAllWindows()
elif add[0]=="98":
    final.append([536,"9"])
    frame=cv2.imread("98_yolo_out_py.jpg")
    cv2.putText(frame, final[-1][1], (536, 557-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    final.append([765,"5"])
    cv2.putText(frame, final[-1][1], (765, 560-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    cv2.imwrite("output/OP_98.jpg",frame)
    cv2.imshow("Output Frame",frame)
    cv2.waitKey()
    cv2.destroyAllWindows()
elif add[0]=="100":
    final.append([433,"5"])
    frame=cv2.imread("output/OP_100.jpg")
    cv2.putText(frame, final[-1][1], (433, 661-25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
    cv2.imwrite("output/OP_100.jpg",frame)
    cv2.imshow("Output Frame",frame)
    cv2.waitKey()
```

Fig. 11: Code snippet for data training

E. Flowchart

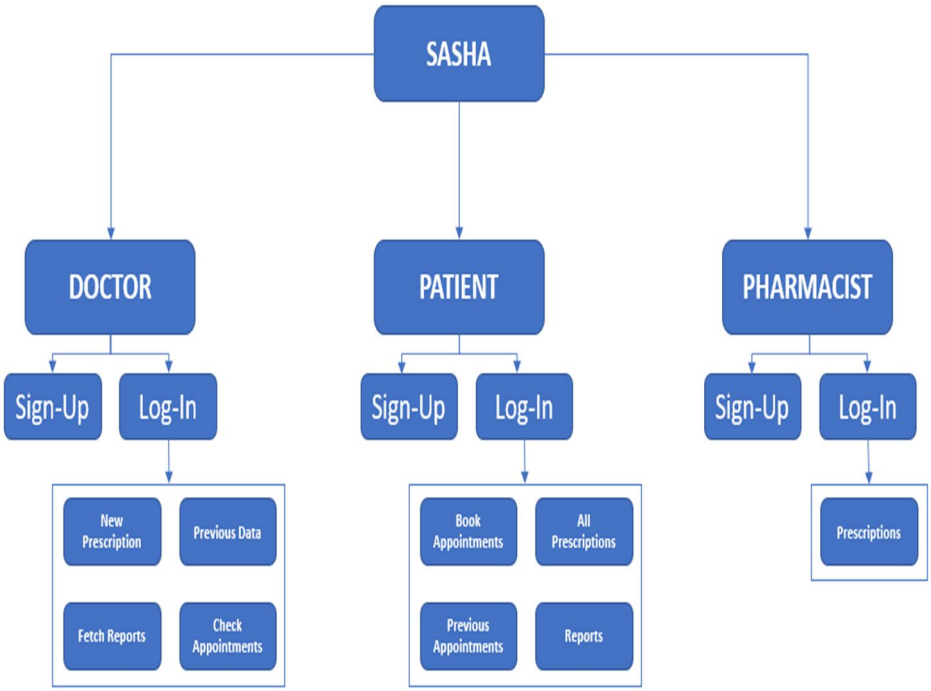


Fig. 12: Flowchart of the project

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

V. RESULTS AND OUTPUT

The device was optimized after sufficient testing and training of the Haar Cascade classifier created to detect the face region from the individual frame. The classifier was trained on the dataset collected to add the user information to the system. A Sign-In/Log-In system was implemented into the system. The system was housed in a GUI to increase the ease of use for the user.
Results obtained:

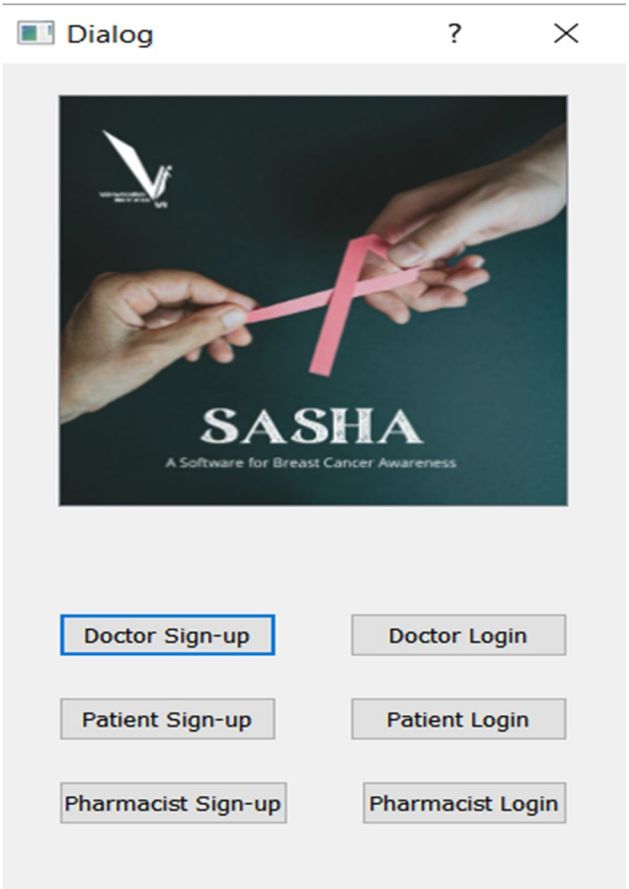


Fig. 13: GUI for Sign-up and Log-in Page (Main Page).

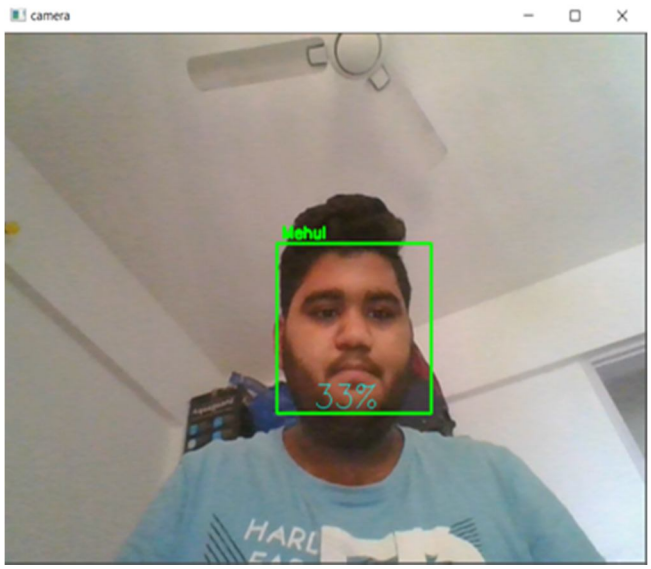


Fig. 14: Logging into the system

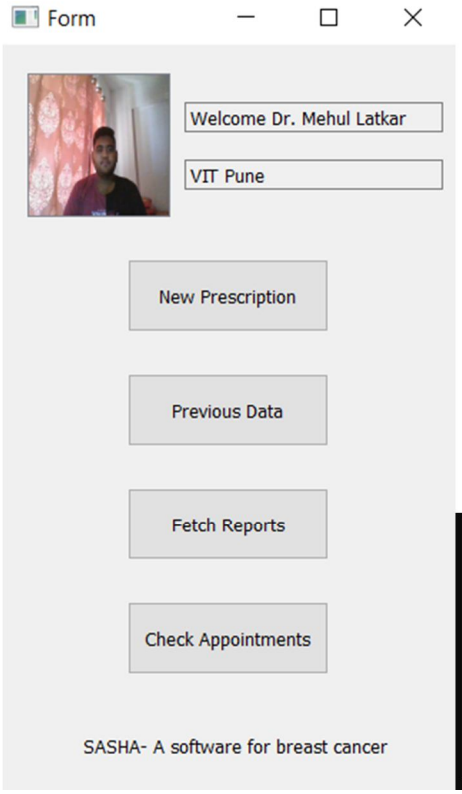


Fig. 15: After-Login page for Doctor.

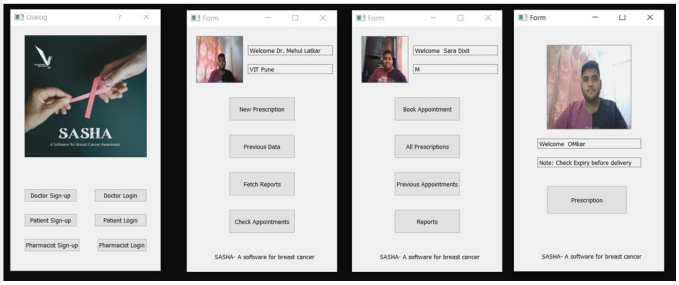


Fig. 16: After-Login page for Doctors, Patients & Pharmacists.

```
--------------------------------------------------
--------------------------------------------------
Patient Name: Aniruddha
Gender: M
Address: qws
Patient Contact: 95621485
Appointment Date: 2021-06-11
Appointment Time: 10:00 - 11:00
Doctor Name: Mehul
Hospital Name: VIT Pune
Description of illness: Cough
--------------------------------------------------
Patient Name: weds
Gender: M
Address: werds
Patient Contact: 123456
Appointment Date: 2021-06-11
Appointment Time: 9:00 - 10:00
Doctor Name: wedsa
Hospital Name: w
Description of illness: wq
--------------------------------------------------
Patient Name: ASD
Gender: F
Address: QWS
Patient Contact: 234
Appointment Date: 2021-06-09
Appointment Time: 11:00 - 12:00
Doctor Name: qasdewsdesd
Hospital Name: qwertyuiop
Description of illness: dkjbvbabviaudbvi
--------------------------------------------------
Patient Name: wdcv
Gender: M
Address: wsx
Patient Contact: 1234
Appointment Date: 2021-06-09
Appointment Time: 9:00 - 10:00
Doctor Name: wsdc
Hospital Name: qwsa
Description of illness: qwerfgvcxz
--------------------------------------------------
```

Fig. 17: Output after fetching appointments.

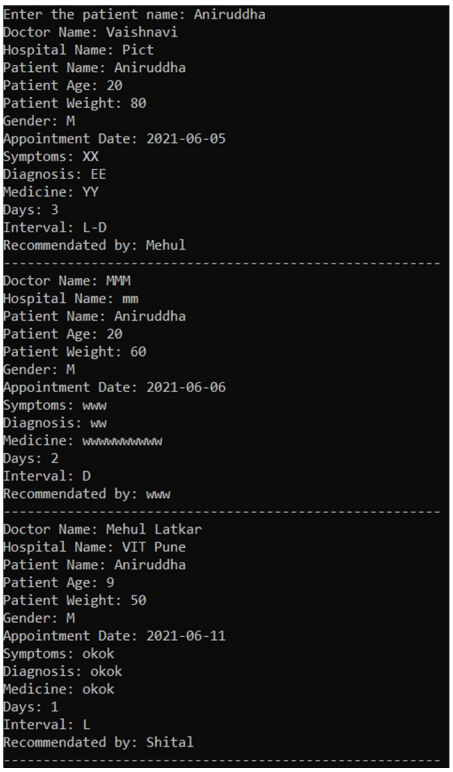


Fig. 18: Fetching past records of patients.

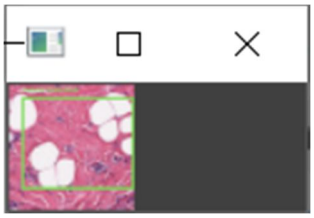


Fig. 19: Results after fetching reports. Fig.

Fig. 20: Results after fetching reports.

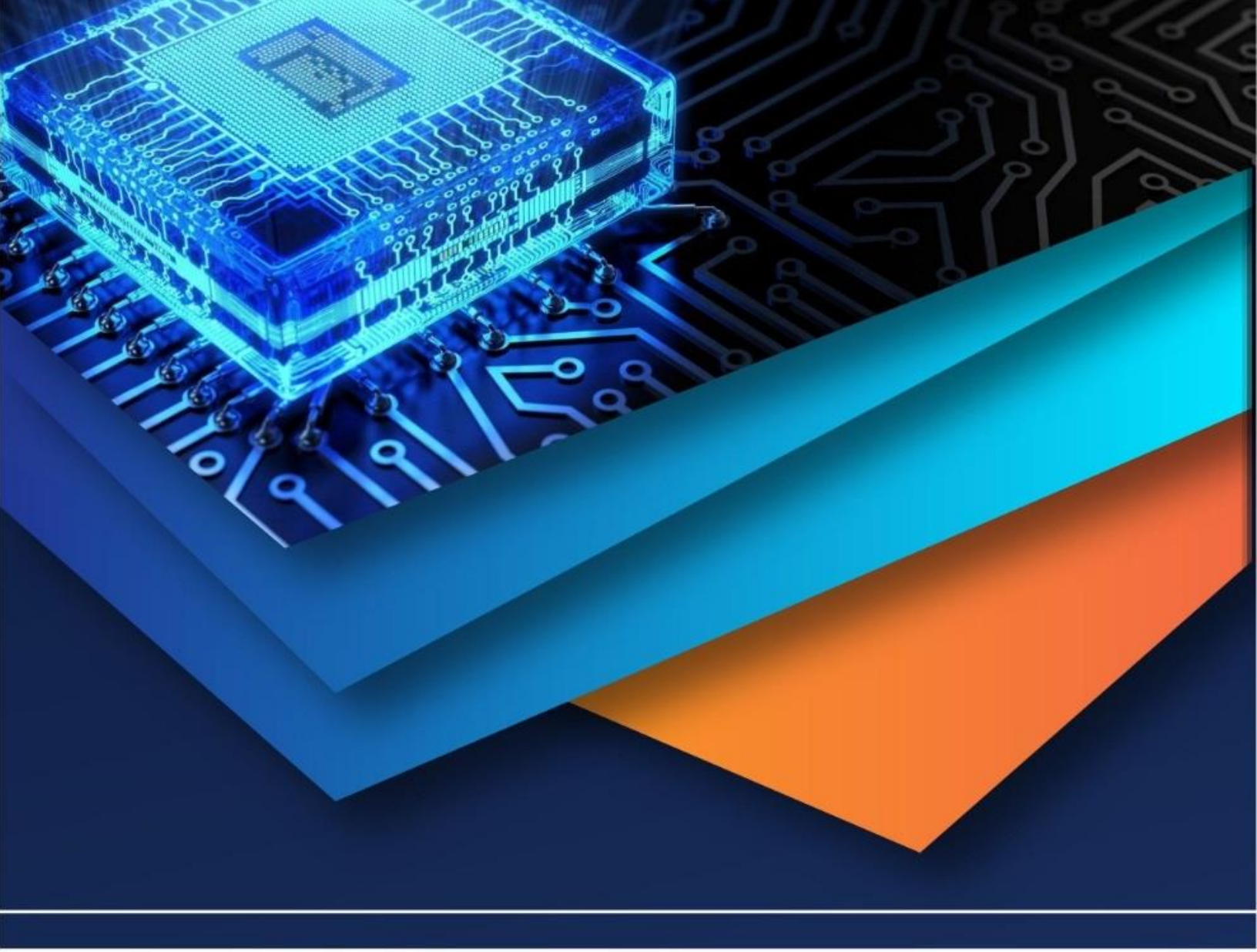
VI. CONCLUSION

The proposed method is efficient in aiding medical personnel with the early detection of breast cancer. The application provides a digital solution with image recognition for system login and receipt management. Furthermore, it provides a virtual approach to stay connected with doctors and patients without the requirement of physically being present in the hospital or pharmacy. It is a secure system wherein the details of the patient are confidential and only accessible by the doctor and pharmacist. The proposed method can be modified and used for detecting other conditions such as lung cancer, brain tumour, kidney stones, etc. The future scope includes plans to build a mobile application for further user convenience.

REFERENCES

[1] Colmenarez A.J., Huang T.S. (1998) Face Detection and Recognition. In: Wechsler H., Phillips P.J., Bruce V., Soulié F.F., Huang T.S. (eds) Face Recognition. NATO ASI Series (Series F: Computer and Systems Sciences), vol 163. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-72201-1_9
[2] R. Padilla, C. F. F. Costa Filho and M. G. F. Cost, “Evaluation of Haar Cascade Classifiers Designed for Face Detection”, World Academy of Science, Engineering and Technology 64, 2012.
[3] Samar Zuhair Alshawwa, Rasha Assad Assiri, “mHealth Interventions for Cancer Care and Support A Systematic Literature Review”, Sys Rev Pharm 2020;11(9):725-744, Vol 11, Issue 9, Sep-Oct 2020.
[4] B. Moghaddam and A. Pentland, “Maximum Likelihood Detection of Faces and Hands”, Int. Workshop on Automatic Face- and Gesture-Recognition, Zurich, 1995.
[5] F. Soulie, E. Viennet, and B. Lamy, “Multi-Modular Neural Network Architectures: Pattern Recognition Applications in Optical Character Recognition and Human Face Recognition”, Intl. Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No 4, 1993. Google Scholar

[6] U.S. Cancer Statistics Working Group. United States Cancer Statistics: 1999–2008 Incidence and Mortality Web-based Report. Atlanta (GA): Department of Health and Human Services, Centers for Disease Control Google Scholar
[7] Silva, J., Lezama, O.B.P., Varela, N., Borrero, L.A.: Integration of data mining classification techniques and ensemble learning for predicting the type of breast cancer recurrence. In: Miani, R., Camargos, L., Zarpelão, B., Rosas, E., Pasquini, R. (eds.) GPC 2019. LNCS, vol. 11484, pp. 18–30. Springer, Cham (2019).
[8] Ojha U., Goel, S.: A study on prediction of breast cancer recurrence using data mining techniques. In: 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, pp. 527–530, 2017.
[9] Pritom, A.I., Munshi, M.A.R., Sabab, S.A., Shihab, S.: Predicting breast cancer recurrence using effective classification and feature selection technique. In: 19th International Conference on Computer and Information Technology (ICCIT), pp. 310–314. IEEE (2016)
[10] Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming (paperback) M Summerfield - 2007 - books.google.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY