

# **1. INTRODUCTION**

Optical mark recognition (also known as OMR) is the process of capturing human-marked data from document forms such as surveys and exams, etc. They are used to read questionnaires, multiple choice examination papers in the form of lines or shaded areas. It is a recognition technology used for collecting data from “fill-in-the-bubble” forms such as educational tests.

The function of this OMR scanner is to scan and capture the marks on the multiple choice scoring sheets, and save the data into a database which you can use to perform further analysis. It is full-fledged OMR checking software that can read and evaluate OMR sheets scanned at any angle and having any colour.

## **1.1 EXISTING SYSTEM**

OMR devices work with a dedicated scanner device that shines a beam of light onto the form paper. Some OMR devices use forms that are pre-printed onto "trans optic" paper and measure the amount of light which passes through the paper; thus a mark on either side of the paper will reduce the amount of light passing through the paper. These scanners work only for a specific color and thickness of the form hence cannot be printed on a general purpose printer. The OMR forms evaluated by dedicated scanners generally range in the quality of 90 - 110 gsm, which are much more expensive than the common plain papers (60 – 70 gsm). Another problem with dedicated machine is their cost and maintenance. Their usage is limited due to high cost, strict specifications and inflexibility in format of the form.

## **1.2 PROPOSED SYSTEM**

The proposed solution to this problem is very simple and cost-effective, which could easily replace the heavy and expensive present day dedicated OMR machines. Unlike scanners and computers, OMR machines are not commonplace hardware resource. The proposed system uses commonly available scanner and computer. The system offers flexibility to the users, which allows designing and printing of OMR

forms on simple ordinary sheets, without the aid of any special pre-processing and color constraints. The filled-in forms are scanned by a normal scanner, and scanned images are provided as input to the application. The application then scans the OMR form and computes the result. The results are stored in the database, which makes it easy to access and interpret the information, thus, making the system efficient and effective.

## 2. LITERATURE SURVEY

### 2.1 PROJECT LITERATURE

Literature 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 books or from websites. Before building the system the above considerations are taken into account for developing the proposed system.

### 2.2 INTRODUCTION TO PYTHON

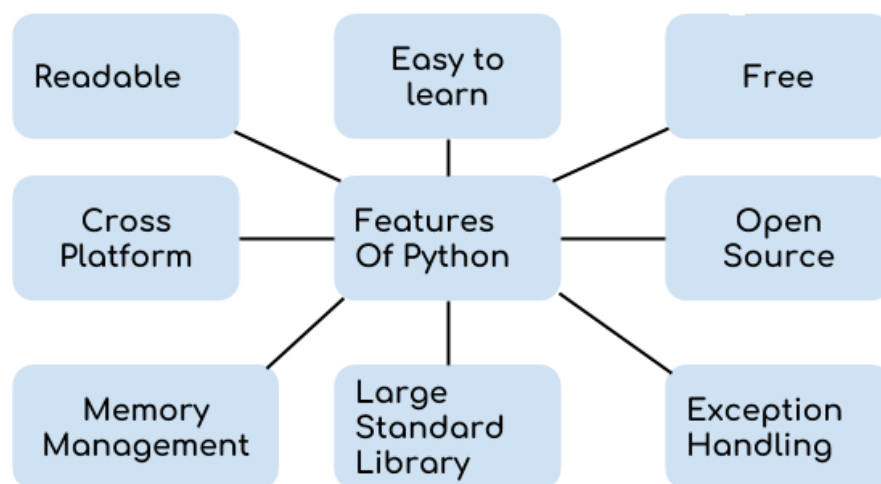
Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is developed by Guido van Rossum. Guido van Rossum started implementing Python in 1989.

#### Features Of Python

- **Readable:** Python is a very readable language.
- **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.
- **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, UNIX etc. This makes it a cross platform and portable language.
- **Open Source:** Python is an open source programming language.

- **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.
- **Free:** Python is free to download and use. This means you can download it for free and use it in your application. Python is an example of FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.
- **Supports exception handling:** An exception is an event that can occur during program execution and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.
- **Advanced features:** Supports generators and list comprehensions. We will cover these features later.
- **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically.



**Fig 2.1 Features of Python**

## 2.2.1 PYTHON TECHNOLOGY

Python is an interpreted, high-level, general-purpose programming language. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology.

- **Designed By :** Guido Van Rossum
- **Developer :** Python Software Foundation
- **First Appeared :** 1990
- **Os :** Linux, MacOS, Windows Vista (And Newer) And More
- **License :** Python Software Foundation License
- **Filename Extensions :** .Py, .Pyi, .Pyc, .Pyd, .Pyo
- **Website :** [Www.Python.Org](http://www.python.org)
- Python 3.8.3 Is The Latest Version Of Python Released On 13 May 2020.

## 2.3 LIBRARIES SPECIFIC TO THE PROJECT

### 2.3.1 Tkinter

**Tkinter** commonly comes bundled with Python, using Tk and is Python's standard GUI framework. It is famous for its simplicity and graphical user interface. It is open-source and available under the Python License. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

### 2.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. OpenCV is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BS (Berkeley Software Distribution) license. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

### 2.3.3 PyQt5

PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, Chromium based web browser, as well as traditional UI development.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called '*QtDesigner*' 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.

### **2.3.4 NumPy**

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Numpy is the core library for scientific computing in Python.

The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

### **2.3.5 SciPy**

SciPy is a free and open-source Python library used for scientific computing and technical computing.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers.

### **2.3.6 Imutils**

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python3.

### **Some common functions:**

#### **Translation**

Translation is the shifting of an image in either the x or y direction. To translate an image in OpenCV you would need to supply the (x, y)-shift, denoted as (tx, ty) to construct the translation matrix M:

#### **Rotation**

Rotating an image in OpenCV is accomplished by making a call to `cv2.getRotationMatrix2D` and `cv2.warpAffine`. Further care has to be taken to supply the (x, y)-coordinate of the point the image is to be rotated about. These calculation calls can quickly add up and make the code bulky and less readable. The `rotate` function in `imutils` helps resolve this problem.

#### **Resizing**

This `resize` function of `imutils` maintains the aspect ratio and provides the keyword arguments `width` and `height` so the image can be resized to the intended width/height while (1) maintaining aspect ratio and (2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.

#### **Skeletonization**

Skeletonization is the process of constructing the "topological skeleton" of an object in an image, where the object is presumed to be white on a black background. OpenCV does not provide a function to explicitly construct the skeleton, but does provide the morphological and binary functions to do so.

For convenience, the `skeletonize` function of `imutils` can be used to construct the topological skeleton of the image.



### 2.3.7 SQLite

SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features:

- Self-contained
- Server-less
- Zero-configuration
- Transactional

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems , among others. SQLite has bindings to many programming languages.

## **3. SYSTEM ANALYSIS AND REQUIREMENTS**

### **3.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and 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. Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

#### **3.1.1 Economical Feasibility**

The developed application can be freely accessed because all the technologies used are freely available (open source).

#### **3.1.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. The developed system has 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.

## **3.2 SOFTWARE AND HARDWARE REQUIREMENTS**

### **3.2.1 Hardware Requirements**

Processor : Core 2 Duo or Higher

Hard Disk : 10 GB

Ram : 1 GB

### **3.2.2 Software Requirements**

OS : Windows XP Professional/Vista/7/8/8.1/10 or Linux or iOS

Front End : Python 3 or above

Tool : Qt-Designer

Backend : Sqlite3

Tool : SqliteStudio

## **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 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

- UML is a pictorial language used to make software blueprints.
- UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

The underlying premise of UML is that no one diagram can capture the different elements of a system at different points often in the software life cycle of a system.

The UML diagrams are as follows:

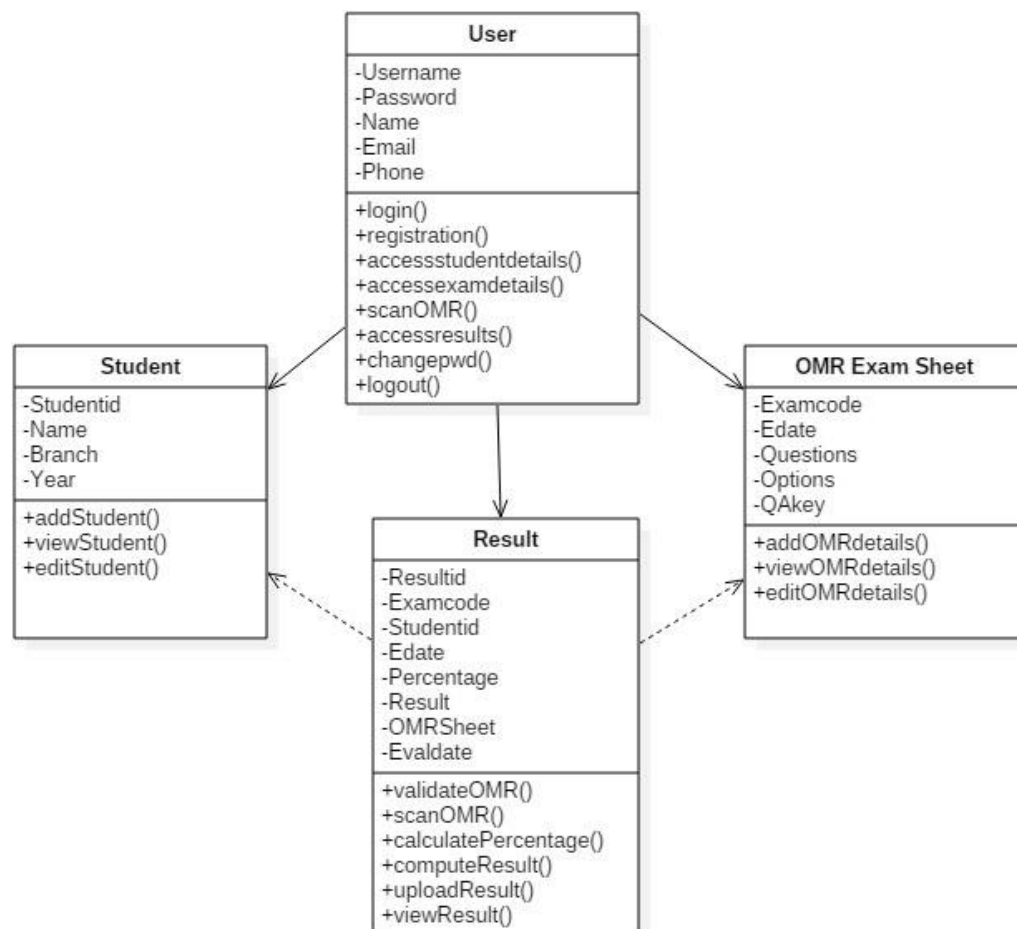
- Use case Diagram
- Activity Diagram.
- Class Diagram
- Sequence Diagram

### 4.1.1 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:

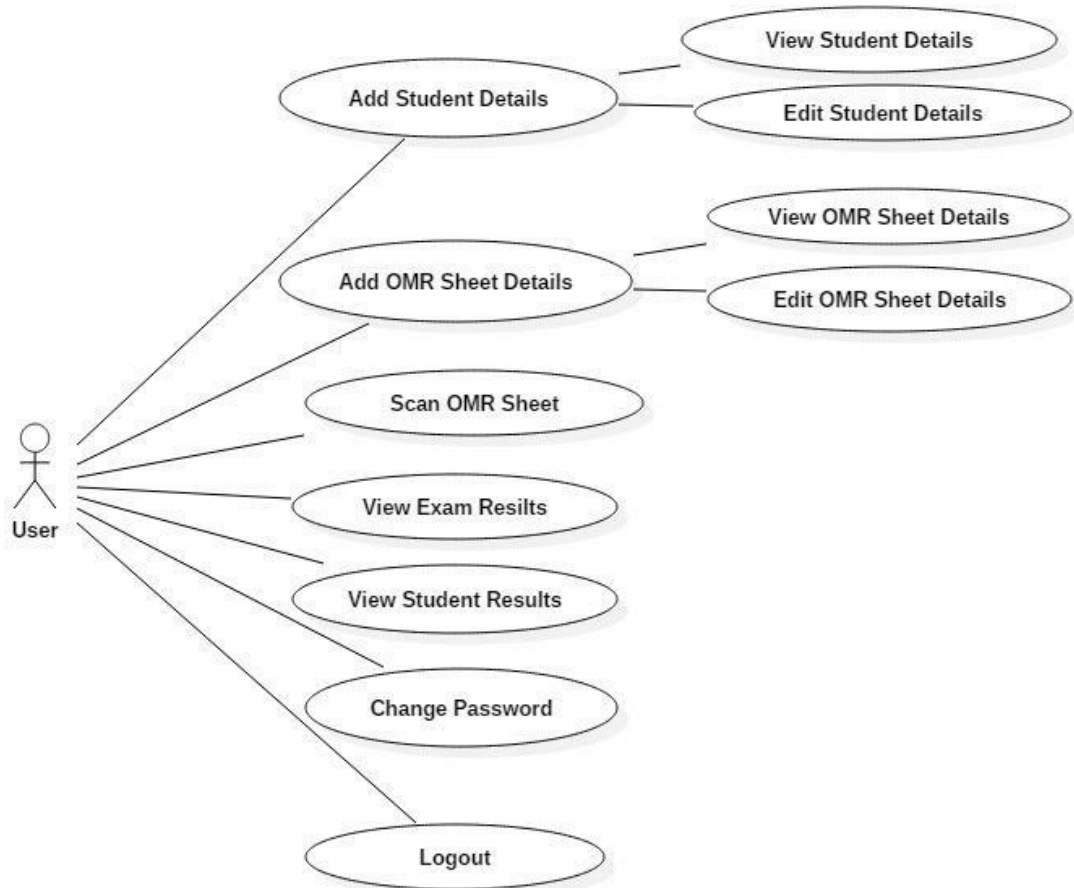
The upper part holds the name of the class. The middle part contains the attributes of the class. The bottom part gives the methods or operations the class can take or undertake.



**Fig 4.1 Class Diagram**

### 4.1.2 Use Case Diagram

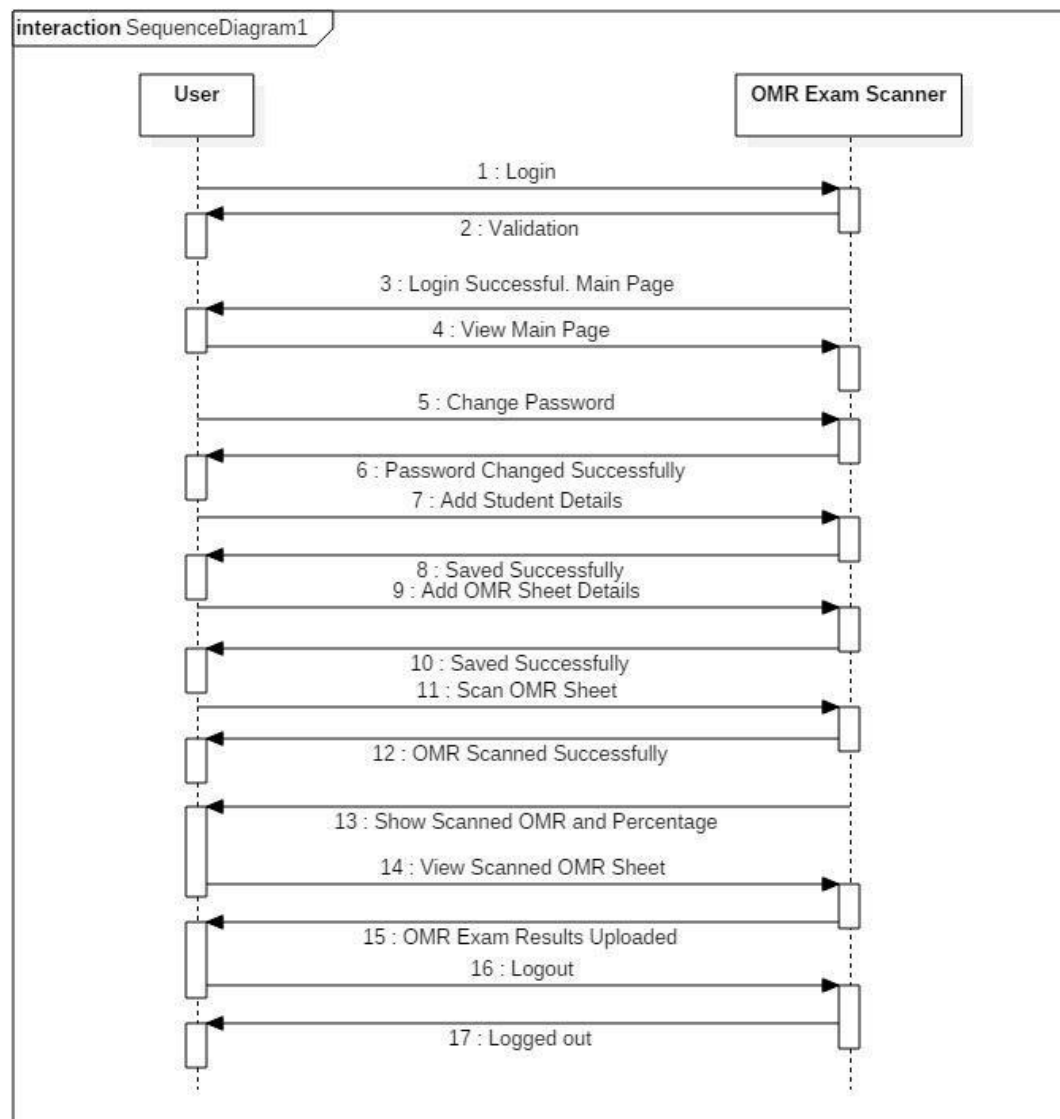
A Use Case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Fig 4.2 Use Case Diagram**

### 4.1.3 Sequence Diagram

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 View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

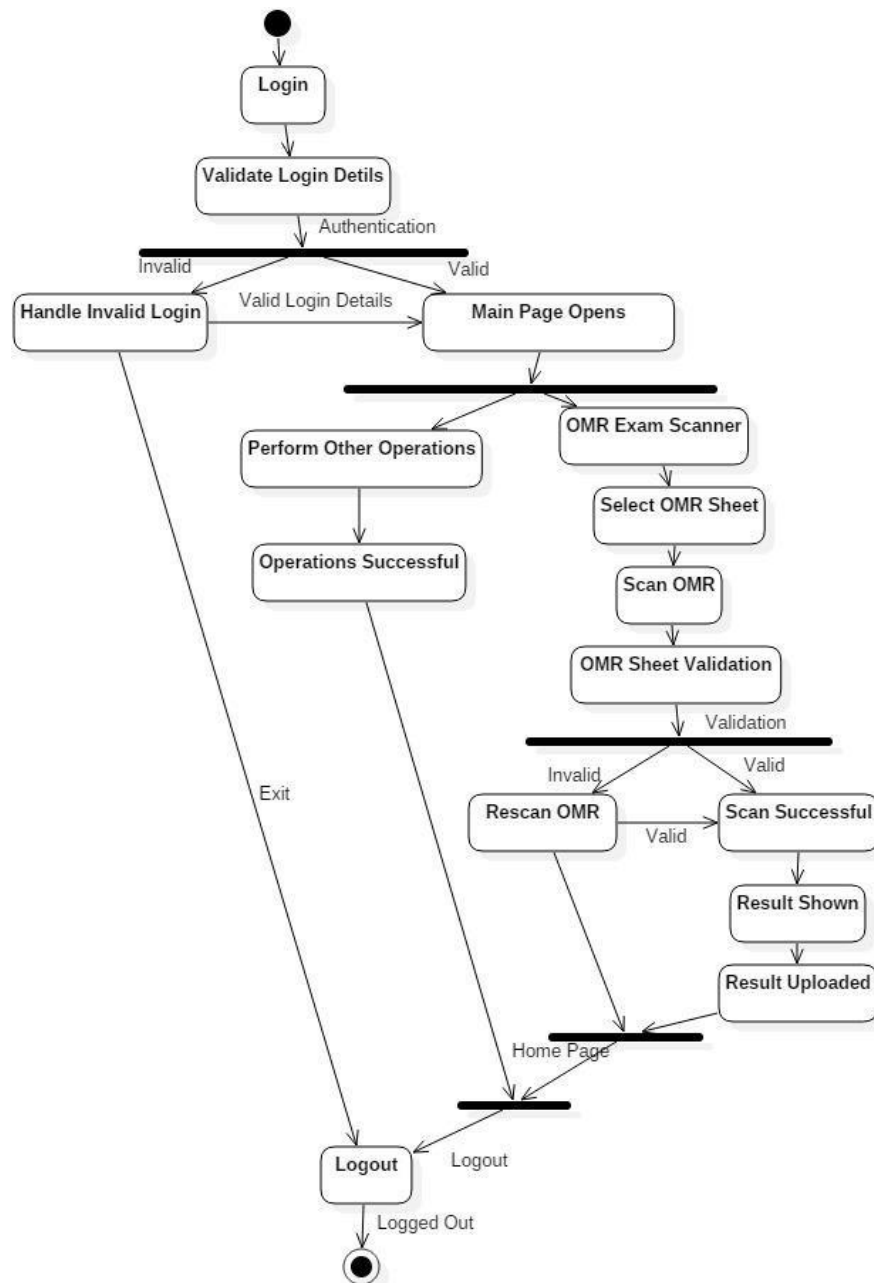


**Fig 4.3 Sequence Diagram**



#### 4.1.4 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 from 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.



**Fig 4.4 Activity Diagram**

## **4.2 MODULES DESCRIPTION**

This application consists of the following modules:

- Authentication Module
- Student Details Module
- OMR Sheet Details Module
- OMR Scanner Module
- Exam Results Module

### **Module I: Authentication Module**

The Authentication Module consists of Login, Registration and Change Password. The User has to Register successfully first by providing details such as Name, Email, Phone, Username and Password. An Existing user can directly Login and access the application. The User also has the option to change his password.

### **Module II: Student Details Module**

The Student Module has components such as Add Student Details, View, Edit or Delete. The Add Student Details takes in details such as Studentid, Name, Branch and Year. The user can edit/delete the student details using View Student Details option. All these details are added or updated into the Database.

### **Module III: OMR Sheet Details Module**

In this the user has to enter the Details of the OMR sheet he/she wants to scan. The Examcode, Exam date, Number of Questions, Number of Options and Question-Answer key in a specific format are taken from the user and are used to scan the OMR Sheet.

### **Module IV: OMR Scanner Module**

In this module the user gives the Pre-existing Examcode and Studentid to successfully scan an OMR sheet. The OMR Scanner scans the selected image and shows the Percentage scored as the output.

## **Module V: Exam Results**

This module automatically generates Percentage and Result and stores it into the database. Resultid is automatically generated using the Examcode and Studentid provided by the user. It also saves details such as Evaluated date and time and Image Path in order to know when the image was scanned and where the image is stored.

## 5. CODING TEMPLATES / CODE

### 5.1 CODING

#### 5.1.1 MAIN.py

```
#importing all the Dialogs
from LOGIN import *
from REGISTER import *

#Calling the functions (methods) when the buttons are clicked

    self.login.clicked.connect(self.openlogin)
    self.reg.clicked.connect(self.openregister)

#Importing Login Dialog and opening it when Login button is clicked
def openlogin(self):
    from LOGIN import Ui_Login
    Login = QtWidgets.QDialog()
    ui = Ui_Login()
    ui.setupUi(Login)
    ret=Login.exec()

#Importing Register Dialog and opening it when Register button is clicked
def openregister(self):
    from REGISTER import Ui_Dialog
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    ret=Dialog.exec()
```

### 5.1.2 LOGIN.py

```
From HOME import *

self.hide=Login.accept

self.login1.clicked.connect(self.login)

#To close the dialog when exit button is clicked

self.exit1.clicked.connect(Login.accept)

def login(self):

    import sqlite3

    Dialog=QtWidgets.QMessageBox()

    uname=self.uid.text()

    passwd=self.pwd.text()

    conn = sqlite3.connect('EXAMOMR.db')

    sql="Select * from REGISTER where Username='"+uname+"';"

    cur=conn.execute(sql)

    row=cur.fetchone()

    if row is not None:

        uid1=row[3]

        pwd1=row[4]

        if uid1==uname and passwd==pwd1:

            self.hide()

            from HOME import Ui_Dialog

            Dialog = QtWidgets.QDialog()

            ui = Ui_Dialog()

            ui.setupUi(Dialog)

            ret=Dialog.exec()

        elif uid1==uname and passwd!=pwd1:
```

```

        conn.rollback()

        Dialog.setWindowTitle("Invalid")

        Dialog.setIcon(QtWidgets.QMessageBox.Warning)

        Dialog.setText("Invalid Username or Password")

        ret=Dialog.exec()

        self.clear()

    else:

        Dialog.setWindowTitle("Invalid")

        Dialog.setIcon(QtWidgets.QMessageBox.Warning)

        Dialog.setText("Invalid Username or Password")

        ret=Dialog.exec()

        self.clear()

    def clear(self):

        self.uid.setText("")

        self.pwd.setText("")

```

### 5.1.3 HOME.py

```

from SDETAILS import *
from OMRDETAILS import *
from SCANOMR import *
from ERESULTS import *
from SRESULTS import *
from VIEWSTU import *
from VIEWOMR import *
from CHANGE import *

#Invoke functions for buttons when clicked

self.p1.clicked.connect(self.student)

self.p2.clicked.connect(self.eomr)

```

```

self.p3.clicked.connect(self.somr)
self.p4.clicked.connect(self.eresult)
self.p5.clicked.connect(self.sresult)
self.p6.clicked.connect(self.sview)
self.p7.clicked.connect(self.eview)
self.p8.clicked.connect(self.change)
self.p9.clicked.connect(Dialog.accept)

```

```

def student(self):
    from SDETAILS import Ui_Dialog
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    ret=Dialog.exec()

```

```

def eomr(self):
    from OMRDETAILS import Ui_Dialog
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    ret=Dialog.exec()

```

```

def somr(self):
    from SCANOMR import Ui_Dialog
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    ret=Dialog.exec()

```

```
def eresult(self):  
    from ERESULTS import Ui_Dialog  
    Dialog = QtWidgets.QDialog()  
    ui = Ui_Dialog()  
    ui.setupUi(Dialog)  
    ret=Dialog.exec()
```

```
def sresult(self):  
    from SRESULTS import Ui_Dialog  
    Dialog = QtWidgets.QDialog()  
    ui = Ui_Dialog()  
    ui.setupUi(Dialog)  
    ret=Dialog.exec()
```

```
def sview(self):  
    from VIEWSTU import Ui_Dialog  
    Dialog = QtWidgets.QDialog()  
    ui = Ui_Dialog()  
    ui.setupUi(Dialog)  
    ret=Dialog.exec()
```

```
def eview(self):  
    from VIEWOMR import Ui_Dialog  
    Dialog = QtWidgets.QDialog()  
    ui = Ui_Dialog()  
    ui.setupUi(Dialog)  
    ret=Dialog.exec()
```

```
def change(self):
```



```

from CHANGE import Ui_cp

cp = QtWidgets.QDialog()

ui = Ui_cp()

ui.setupUi(cp)

ret=cp.exec()

```

#### 5.1.4 OMRDETAILS.py

```

self.s2.clicked.connect(self.enter)

self.e2.clicked.connect(Dialog.accept)

self.e0.clicked.connect(self.help)

def enter(self):

    import datetime

    conn = sqlite3.connect('EXAMOMR.db')

    Dialog=QtWidgets.QMessageBox()

    ec=self.ec1.text()

    date=self.ed.text()

    nq=self.q.text()

    np=self.op.text()

    qak=self.qa.text()

    try:

        if (ec!="" and date!="" and nq!="" and np!="" and qak!=""):

            cur=conn.execute("INSERT INTO OMRDETAILS
(Examcode,Edate,Questions,Options,QAkey) VALUES
('"+str(ec)+"','"+date+"','"+nq+"','"+np+"','"+str(qak)+"');")

            conn.commit()

            Dialog.setWindowTitle("Saved")

            Dialog.setIcon(QtWidgets.QMessageBox.Information)

            Dialog.setText("Saved Successfully")

            ret=Dialog.exec()

```

```

        self.clear()
    else:
        conn.rollback()

        Dialog.setWindowTitle("Invalid")

        Dialog.setIcon(QtWidgets.QMessageBox.Warning)

        Dialog.setText("Add all the Details")

        ret=Dialog.exec()

        self.clear()

    except:

        Dialog.setWindowTitle("Invalid")

        Dialog.setIcon(QtWidgets.QMessageBox.Warning)

        Dialog.setText("Exam code already exists")

        ret=Dialog.exec()

        self.clear()

def clear(self):
    self.ec1.setText("")
    self.ed.setText("")
    self.q.setText("")
    self.op.setText("")
    self.qa.setText("")

def help(self):
    Dialog=QtWidgets.QMessageBox()

    Dialog.setWindowTitle("Help")

    Dialog.setIcon(QtWidgets.QMessageBox.Information)

    Dialog.setText("""Please Enter the Question Answer key starting from 0 format
Q:A,Q:A, ...

Example:- 0:1,1:2 specifies 1st answer as B(1) and 2nd answer as C(2)

```

Seperate Q and A using ':' and Q/A key pair using ','.

```
        """)
    ret=Dialog.exec()
```

### 5.1.5 VIEWSTU.py

```
conn = sqlite3.connect('EXAMOMR.db')

sql="Select * from STUDENT;"
result=conn.execute(sql)
self.t3.setRowCount(0)

for row_number,row_data in enumerate(result):
    self.t3.insertRow(row_number)

    for column_number,data in enumerate(row_data):
        self.t3.setItem(row_number,column_number,
QtWidgets.QTableWidgetItem(str(data)))

conn.close()

self.u1.clicked.connect(self.update)
self.d1.clicked.connect(self.delete)
self.ss.clicked.connect(self.get)
self.ee1.clicked.connect(Dialog.accept)

def get(self):
    sid4=self.vs.text()

    Dialog=QtWidgets.QMessageBox()
    conn=sqlite3.connect('EXAMOMR.db')
    sql="Select * from STUDENT where Studentid='"+sid4+"';"
    cur = conn.execute(sql)
    result = cur.fetchone()
    if result is not None:
        sid = result[0]
```

```

name = result[1]
branch = result[2]
year = result[3]

if sid==sid4:
    self.vs1.setText(sid)
    self.vs2.setText(name)
    b=self.vs3.findText(branch)
    self.vs3.setCurrentIndex(b)
    y=self.vs4.findText(year)
    self.vs4.setCurrentIndex(y)

else:
    Dialog.setWindowTitle("Invalid")
    Dialog.setIcon(QtWidgets.QMessageBox.Warning)
    Dialog.setText("Invalid StudentId")
    ret=Dialog.exec()
    self.clear()

def update(self):
    Dialog=QtWidgets.QMessageBox()
    d=self.vs1.text()
    if d=="":
        Dialog.setWindowTitle("Invalid")
        Dialog.setIcon(QtWidgets.QMessageBox.Warning)
        Dialog.setText("Please Enter Student Id")
        ret=Dialog.exec()
    else:
        conn=sqlite3.connect('EXAMOMR.db')

```

```

sid=self.vs1.text()

name=self.vs2.text()


branch=self.vs3.currentText()

year=self.vs4.currentText()

if year!="" and sid!="" and name!="" and branch!="":

    cur=conn.execute("UPDATE STUDENT SET
Name='"+name+"',Branch='"+branch+"',Year='"+year+"' where
Studentid='"+sid+"';")

    conn.commit()

    Dialog.setWindowTitle("Saved")

    Dialog.setIcon(QtWidgets.QMessageBox.Information)

    Dialog.setText("Updated Successfully")

    ret=Dialog.exec()

    self.clear()

    self.t3.clearContents()

    conn = sqlite3.connect('EXAMOMR.db')

    sql="Select * from STUDENT;"

    result=conn.execute(sql)

    self.t3.setRowCount(0)

    for row_number,row_data in enumerate(result):

        self.t3.insertRow(row_number)

        for column_number,data in enumerate(row_data):

            self.t3.setItem(row_number,column_number,
QtWidgets.QTableWidgetItem(str(data)))

        conn.close()

else:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Enter all Details")

```

```

        ret=Dialog.exec()

def delete(self):
    Dialog=QtWidgets.QMessageBox()
    conn=sqlite3.connect('EXAMOMR.db')
    Dialog=QtWidgets.QMessageBox()
    sid=self.vs1.text()
    if sid=="":
        Dialog.setWindowTitle("Invalid")
        Dialog.setIcon(QtWidgets.QMessageBox.Warning)
        Dialog.setText("Please Enter Student Id")
        ret=Dialog.exec()
    else:
        cur=conn.execute("DELETE from STUDENT where Studentid='"+sid+"'")
        conn.commit()
        Dialog.setIcon(QtWidgets.QMessageBox.Information)
        Dialog.setWindowTitle("Saved")
        Dialog.setText("Deleted Successfully")
        ret=Dialog.exec()
        self.t3.clearContents()
        self.clear()
        self.t3.clearContents()
        conn = sqlite3.connect('EXAMOMR.db')
        sql="Select * from STUDENT;"
        result=conn.execute(sql)
        self.t3.setRowCount(0)
        for row_number,row_data in enumerate(result):
            self.t3.insertRow(row_number)
            for column_number,data in enumerate(row_data):

```

```

        self.t3.setItem(row_number,column_number,
QtWidgets.QTableWidgetItem(str(data)))

        conn.close()

```

```

def clear(self):

    self.vs.setText("")

    self.vs1.setText("")

    self.vs2.setText("")

    self.vs3.setCurrentIndex(0)

    self.vs4.setCurrentIndex(0)

```

### 5.1.6 ERESULTS.py

```

        self.s5.clicked.connect(self.get)

        self.e5.clicked.connect(Dialog.accept)

def get(self):

    Dialog=QtWidgets.QMessageBox()

    ecode=self.ec3.text()

    conn=sqlite3.connect('EXAMOMR.db')

    sql="Select STUDENT.*,RESULT.Percentage,RESULT.Result from
STUDENT,RESULT WHERE RESULT.Studentid=STUDENT.Studentid and
RESULT.Examcode='"+ecode+"';"

    sql1="Select * from RESULT where Examcode='"+ecode+"';"

    cur=conn.execute(sql1)

    row=cur.fetchone()

    if row is not None:

        ec=row[0]

        if ecode==ec:

            result=conn.execute(sql)

            if result is not None:

                self.t1.setRowCount(0)

```

```

        for row_number,row_data in enumerate(result):
            self.t1.insertRow(row_number)

            for column_number,data in enumerate(row_data):
                self.t1.setItem(row_number,column_number,
QtWidgets.QTableWidgetItem(str(data)))

            conn.close()

    else:
        Dialog.setWindowTitle("Invalid")
        Dialog.setIcon(QtWidgets.QMessageBox.Warning)
        Dialog.setText("Invalid Examcode")
        ret=Dialog.exec()
        self.clear()

    def clear(self):
        self.ec3.setText("")
        self.t1.clearContents()
        self.t2.clearContents()

```

### **5.1.7 SCANOMR.py**

```

# import the necessary packages
from imutils.perspective import four_point_transform
from imutils import contours
import numpy as np
import argparse
import imutils
import cv2
import tkinter as tk
from tkinter import filedialog

self.s34.clicked.connect(self.proceed)

```



```

self.e34.clicked.connect(Dialog.accept)

def proceed(self):

    import sqlite3

    from datetime import datetime

    Dialog=QtWidgets.QMessageBox()

    combo=""

    ri=""

    ec=self.ec2.text()

    sid=self.sid1.text()

    combo=ec+sid

    conn = sqlite3.connect('EXAMOMR.db')

    sql="Select OMRDETAILS.*,STUDENT.Studentid from
OMRDETAILS,STUDENT where OMRDETAILS.Examcode='"+ec+"' and
STUDENT.Studentid='"+sid+"";"

    cur=conn.execute(sql)

    row=cur.fetchone()

    sql1="Select * from RESULT where Resultid='"+combo+"";"

    cur1=conn.execute(sql1)

    result=cur1.fetchone()

    if result is not None:

        ri=result[7]

    if row is not None:

        examcode=row[0]

        edate=row[1]

        q1=row[2]

        nq=int(q1)

        o1=row[3]

        op=int(o1)

        qakey=row[4]

        studid=row[5]

```

if examcode==ec and studid==sid:

try:

```
ak=dict(item.split(":") for item in qakey.split(","))
ANSWER_KEY= {int(k):int(v) for k,v in ak.items()}
now = datetime.now()
dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
print(dt_string)
root = tk.Tk()
root.withdraw()
file_path = filedialog.askopenfilename()
print(file_path)
# load the image, convert it to grayscale, blur it
# slightly, then find edges
image = cv2.imread(r""+file_path+"")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
edged = cv2.Canny(blurred, 75, 200)
# find contours in the edge map, then initialize
# the contour that corresponds to the document
cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
docCnt = None
# ensure that at least one contour was found
if len(cnts) > 0:
# sort the contours according to their size in descending order
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
# loop over the sorted contours
    for c in cnts:
```

```

# approximate the contour
peri = cv2.arcLength(c, True)
approx = cv2.approxPolyDP(c, 0.02 * peri, True)
# if our approximated contour has four points,
# then we can assume we have found the paper
if len(approx) == 4:
    docCnt = approx
    break

# apply a four point perspective transform to both the
# original image and grayscale image to obtain a top-down
# birds eye view of the paper
paper = four_point_transform(image, docCnt.reshape(4, 2))
warped = four_point_transform(gray, docCnt.reshape(4, 2))
epaper = four_point_transform(image, docCnt.reshape(4, 2))
# apply Otsu's thresholding method to binarize the warped
# piece of paper
thresh = cv2.threshold(warped, 0, 255, cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]
# find contours in the thresholded image, then initialize
# the list of contours that correspond to questions
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
questionCnts = []
# loop over the contours
for c in cnts:
    # compute the bounding box of the contour, then use the
    # bounding box to derive the aspect ratio
    (x, y, w, h) = cv2.boundingRect(c)

```

```

        ar = w / float(h)

        # in order to label the contour as a question, region
        # should be sufficiently wide, sufficiently tall, and
        # have an aspect ratio approximately equal to 1

        if w >= 20 and h >= 20 and ar >= 0.9 and ar <= 1.1:
            questionCnts.append(c)

    # sort the question contours top-to-bottom, then initialize
    # the total number of correct answers
    questionCnts = contours.sort_contours(questionCnts,method="top-to-
bottom")[0]

    correct = 0

    # each question has 5 possible answers, to loop over the
    # question in batches of 5
    for (q, i) in enumerate(np.arange(0, len(questionCnts), op)):
        # sort the contours for the current question from
        # left to right, then initialize the index of the
        # bubbled answer

        cnts = contours.sort_contours(questionCnts[i:i + op])[0]

        bubbled = None

        # loop over the sorted contours
        for (j, c) in enumerate(cnts):
            # construct a mask that reveals only the current
            # "bubble" for the question

            mask = np.zeros(thresh.shape, dtype="uint8")

            cv2.drawContours(mask, [c], -1, 255, -1)

            # apply the mask to the thresholded image, then
            # count the number of non-zero pixels in the
            # bubble area

```

```

        mask = cv2.bitwise_and(thresh, thresh, mask=mask)

        total = cv2.countNonZero(mask)

        # if the current total has a larger number of total

        # non-zero pixels, then we are examining the currently

        # bubbled-in answer

        if bubbled is None or total > bubbled[0]:

            bubbled = (total, j)

# initialize the contour color and the index of the *correct* answer

color = (0, 0, 255)

k = ANSWER_KEY[q]

# check to see if the bubbled answer is correct

if k == bubbled[1]:

    color = (0, 255, 0)

    correct += 1

# draw the outline of the correct answer on the test

cv2.drawContours(paper, [cnts[k]], -1, color, 3)

# grab the test taker

score = (correct / nq) * 100

p=score

perc=round(p,2)

print(perc)

res=""

if score>=36:

    res="Pass"

else:

    res="Fail"

print("[INFO] score: {:.2f}%".format(score))

cv2.putText(paper, "{:.2f}%".format(score), (10, 30),

cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

```

```

scale_percent = 85 # percent of original size

w = int(paper.shape[1] * scale_percent / 100)
h = int(paper.shape[0] * scale_percent / 100)

dim = (w, h)

# resize image

exam = cv2.resize(paper, dim, interpolation = cv2.INTER_AREA)

width = int(epaper.shape[1] * scale_percent / 100)
height = int(epaper.shape[0] * scale_percent / 100)

dim = (width, height)

# resize image

original = cv2.resize(epaper, dim, interpolation = cv2.INTER_AREA)

cv2.imshow("Exam", original)

cv2.imshow("Result", exam)

cv2.waitKey(0)

self.sid1.setText("")

if ri==combo:

    cur2=conn.execute("UPDATE RESULT SET
Percentage='"+str(perc)+"',Result='"+str(res)+"',Omrsheet='"+str(file_path)+"',Evadate='"+str(dt_string)+"' where Resultid='"+combo+"';")

    cur2=conn.execute(sql)

    conn.commit()

    Dialog.setWindowTitle("Updated")

    Dialog.setIcon(QtWidgets.QMessageBox.Information)

    Dialog.setText("OMR Sheet Updated Successfully")

    ret=Dialog.exec()

else:

    cur3=conn.execute("INSERT INTO RESULT
(Examcode,Studentid,Edate,Percentage,Result,Omrsheet,Evadata,Resultid) VALUES
('"+str(examcode)+"','"+str(sid)+"','"+str(edate)+"','"+str(perc)+"','"+str(res)+"','"+str(file_path)+"','"+str(dt_string)+"','"+str(combo)+"');")

    cur3=conn.execute(sql)

```

```

        conn.commit()

        Dialog.setWindowTitle("Inserted")

        Dialog.setIcon(QtWidgets.QMessageBox.Information)

        Dialog.setText("OMR Sheet Scanner Successfully")

        ret=Dialog.exec()

except AttributeError:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Invalid Image")

    ret=Dialog.exec()

    self.clear()

except KeyError:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Rescan OMR sheet")

    ret=Dialog.exec()

    self.clear()

except ValueError:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Rescan OMR sheet")

    ret=Dialog.exec()

    self.clear()

except IndexError:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Rescan OMR sheet")

    ret=Dialog.exec()

    self.clear()

```

```

except sqlite3.Error:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Database Error")

    ret=Dialog.exec()

    self.clear()

else:

    Dialog.setWindowTitle("Invalid")

    Dialog.setIcon(QtWidgets.QMessageBox.Warning)

    Dialog.setText("Invalid Examcode or Studentid")

    ret=Dialog.exec()

    self.clear()

def clear(self):

    self.ec2.setText("")

    self.sid1.setText("")

```



## **6. SYSTEM TESTING**

### **6.1 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.

### **6.2 TYPES OF TESTS**

#### **6.2.1 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.

### **6.2.2 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 satisfactory, as shown by successful 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.

### **6.2.3 Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- 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.

#### **6.2.4 System Testing**

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.

#### **6.2.5 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.

#### **6.2.6 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 and you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **6.2.7 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.

## **6.3 TEST APPROACH**

Testing can be done in two ways

### **6.3.1 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 within 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.

### **6.3.2 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.

## 7. RESULTS AND VALIDATION

### 7.1 OUTPUT SCREENS

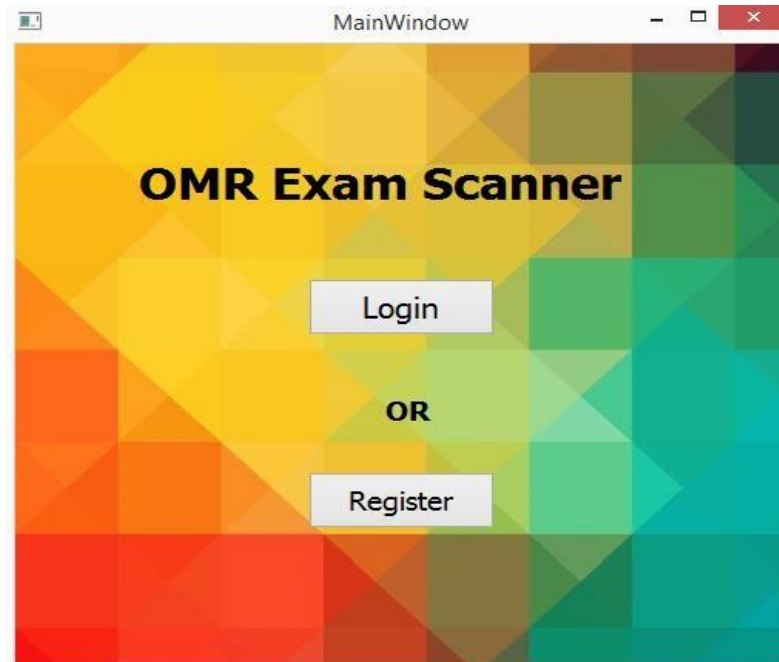


Fig 7.1 Main Screen



Fig 7.2 Login Screen



A screenshot of a 'Register' dialog box. The title bar says 'Dialog' with a question mark and a close button. The background has a colorful geometric pattern. At the top, the word 'Register' is in a pink box. Below it are five input fields labeled 'Name', 'Email Id', 'Phone Number', 'Username', and 'Password'. At the bottom are two buttons: 'Register' and 'Exit'.

Register

Name

Email Id


Phone Number

Username

Password

Register Exit

**Fig 7.3 Register Screen**



A screenshot of a 'Home' dialog box. The title bar says 'Dialog' with a question mark and a close button. The background has a colorful geometric pattern. At the top, the word 'Welcome' is in the center. Below it are six buttons arranged in three rows: 'Enter Student Details' and 'View Student Details' in the first row; 'Enter OMR Sheet Details' and 'View OMR Sheet Details' in the second row; 'Fetch Exam Results' and 'Fetch Student Results' in the third row. Below these is a single button 'Scan OMR Sheets'. At the bottom are two buttons: 'Change Password' and 'Logout'.

Welcome

Enter Student Details View Student Details

Enter OMR Sheet Details View OMR Sheet Details

Fetch Exam Results Fetch Student Results

Scan OMR Sheets

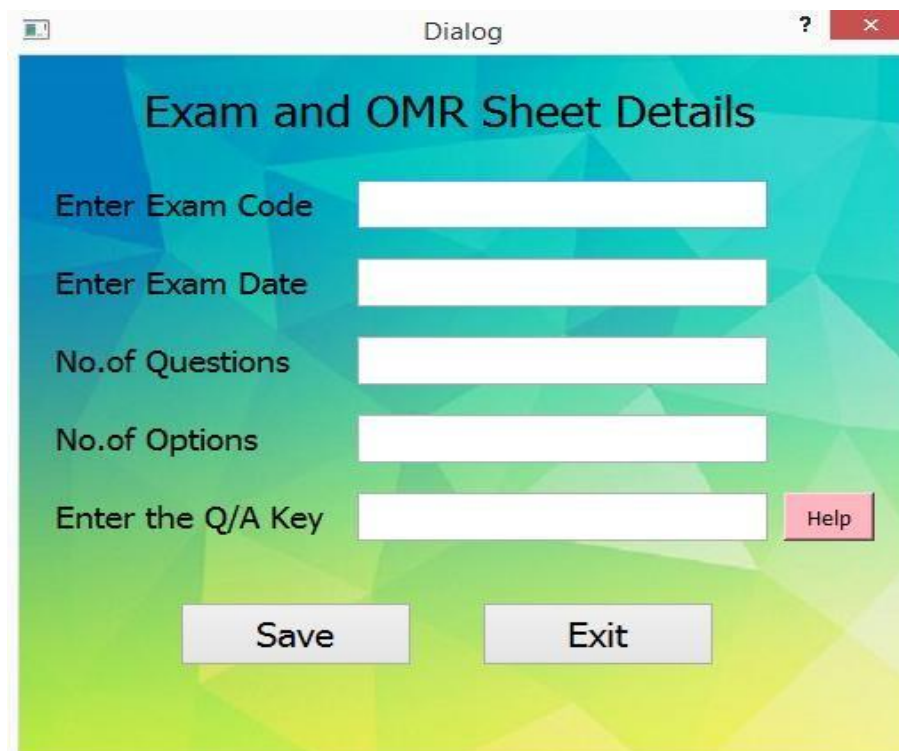
Change Password Logout

**Fig 7.4 Home**



A screenshot of a Windows-style dialog box titled "Enter Student Details". The dialog has a teal and green geometric background. It contains four input fields: "Student Id" (text), "Name" (text), "Branch" (dropdown menu showing "CSE"), and "Year" (dropdown menu showing "I"). At the bottom, there are two buttons: "Save" and "Exit". The window title bar shows "Dialog" and standard Windows window controls.

**Fig 7.5 Student Details**



A screenshot of a Windows-style dialog box titled "Exam and OMR Sheet Details". The dialog has a teal and green geometric background. It contains five input fields: "Enter Exam Code" (text), "Enter Exam Date" (text), "No.of Questions" (text), "No.of Options" (text), and "Enter the Q/A Key" (text). To the right of the "Enter the Q/A Key" field is a small pink "Help" button. At the bottom, there are two buttons: "Save" and "Exit". The window title bar shows "Dialog" and standard Windows window controls.

**Fig 7.6 Exam Details**



**View Student Details**

Enter Student Id

	Student Id	Name	Branch	Year
1	19CS01	Naveen	CSE	I
2	19CS02	Shwetha	CSE	I
3	19CS03	Harika	CSE	I
4	19ME01	Ravi	ME	I
5	19ME02	Anil	ME	I
6	19ME03	Akhilesh	ME	I
7	18CS01	Hari	CSE	II

**Edit Student Details**

Student Id

Name

Branch

Year

**Fig 7.7 View Student Details**

**Exam Results**

Enter Exam Code

	Student Id	Name	Branch	Year	Percentage	Results
1	17CS01	Sai Kumar	CSE	III	80	Pass
2	17CS02	Bhavika	CSE	III	50	Pass
3	17CS03	Ramya	CSE	III	30	Fail
4	17CS04	Kavya	CSE	III	90	Pass

**Fig 7.8 Exam Results**

Dialog

### View OMR Sheet Details

Enter Exam Code

	Exam Code	Exam Date	No.of Questions	No.of Options	Q/A Key
1	1	2020-05-20	5	5	0:0,1:1,2:2,3:3,4:4
2	19PPS	2019-05-14	5	5	0:4,1:1,2:2,3:0,4:4
3	19PHY	2019-05-16	15	5	0:1,1:0,2:3,3:1,4:4
4	19MAT	2019-05-18	10	5	0:0,1:1,2:2,3:1,4:4
5	19ESC	2019-05-19	6	4	0:0,1:3,2:1,3:3,4:4
6	18DBM	2018-04-28	10	5	0:1,1:2,2:3,3:4,4:4

### Edit OMR Sheet Details

Exam Code

Exam Date

No.of Questions

No.of Options

Enter the Q/A Key

**Fig 7.9 View OMR Sheet Details**

Dialog


### Student Results

Enter Student Id

Student Name  Branch

	Exam Code	Exam Date	Year	Percentage	Results
1	19MAT	2019-05-18	I	50	Pass
2	19PHY	2019-05-16	I	73.33	Pass
3	19PPS	2019-05-14	I	60	Pass

**Fig 7.10 Student Results**



A dialog box titled "Scan Exam OMR Sheets" with a red-to-blue gradient background. It contains two input fields: "Enter Exam Code" and "Enter Student Id". At the bottom, there are two buttons: "Scan OMR Sheet" and "Exit".

Dialog

### Scan Exam OMR Sheets

Enter Exam Code

Enter Student Id

**Fig 7.11 Scan OMR Sheet**



A dialog box titled "Change Password" with a yellow-to-pink gradient background. It contains three input fields: "Username", "Old Password", and "New Password". At the bottom, there are two buttons: "Change Password" and "Cancel".

Dialog

### Change Password

Username

Old Password

New Password

**Fig 7.12 Change Password**

Exam Code : 19PHY  
Student Id : 19CS02

1.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input checked="" type="radio"/> E
2.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input checked="" type="radio"/> D	<input type="radio"/> E
3.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input checked="" type="radio"/> D	<input type="radio"/> E
4.	<input type="radio"/> A	<input checked="" type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
5.	<input type="radio"/> A	<input type="radio"/> B	<input checked="" type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
6.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input checked="" type="radio"/> E
7.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input checked="" type="radio"/> E
8.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input checked="" type="radio"/> D	<input type="radio"/> E
9.	<input checked="" type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
10.	<input type="radio"/> A	<input type="radio"/> B	<input checked="" type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
11.	<input type="radio"/> A	<input type="radio"/> B	<input checked="" type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
12.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input checked="" type="radio"/> E
13.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input checked="" type="radio"/> D	<input type="radio"/> E
14.	<input checked="" type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E
15.	<input type="radio"/> A	<input checked="" type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D	<input type="radio"/> E

Fig 7.13 Original OMR sheet



Exam

Exam Code : 19PHY  
Student Id : 19CS02

1.

A

B

C

D

2.

A

B

C

E

3.

A

B

C

E

4.

A

C

D

E

5.

A

B

D

E

6.

A

B

C

D

7.

A

B

C

D

8.

A

B

C

E

9.

B

C

D

E

10.

A

B

D

E

11.

A

B

D

E

12.

A

B

C

D

13.

A

B

C

E

14.

B

C

D

E

15.

A

C

D

E

Result

73.33% Exam Code : 19PHY  
Student Id : 19CS02

1.

A

B

C

D

2.

A

B

C

E

3.

A

B

C

E

4.

A

C

D

E

5.

A

B

D

E

6.

A

B

C

D

7.

A

B

C

D

8.

A

B

C

E

9.

B

C

D

E

10.

A

B

D

E

11.

A

B

D

E

12.

A

B

C

D

13.

A

B

C

E

14.

B

C

D

E

15.

A

C

D

E

Fig 7.14 Result after Scanning OMR

SQLiteStudio (3.2.1) - [STUDENT (EXAMOMR)]

Database Structure View Tools Help

Databases Filter by name

EXAMOMR (SQLite 3)

Tables (4)

- OMRDETAILS
- REGISTER
- RESULT
- STUDENT

Views

Grid view Form view

	Studentid	Name	Branch	Year
1	19CS01	Naveen	CSE	I
2	19CS02	Shwetha	CSE	I
3	19CS03	Harika	CSE	I
4	19ME01	Ravi	ME	I
5	19ME02	Anil	ME	I
6	19ME03	Akhilesh	ME	I
7	18CS01	Hari	CSE	II
8	18CS02	Bala	CSE	II
9	18CS03	Ram	CSE	II
10	18IT01	Nihitha	IT	II
11	18IT02	Santosh	IT	II
12	18EC01	Venu	ECE	II
13	18EC02	Priyanka	ECE	II
14	17CS01	Sai Kumar	CSE	III
15	17CS02	Bhavika	CSE	III
16	17CS04	Kavya	CSE	III
17	17CS03	Ramya	CSE	III

Fig 7.15 STUDENT Table

SQLiteStudio (3.2.1) - [OMRDETAILS (EXAMOMR)]

Database Structure View Tools Help

Databases Filter by name

EXAMOMR (SQLite 3)

Tables (4)

- OMRDETAILS
- REGISTER
- RESULT
- STUDENT

Views

Grid view Form view

	Examcode	Edate	Questions	Options	QAkey
1	16PPS	2016-12-05	5	5	0:0,1:1,2:2,3:3,4:4
2	19PPS	2019-12-14	5	5	0:4,1:1,2:2,3:0,4:3
3	19PHY	2019-12-16	15	5	0:1,1:0,2:3,3:1,4:2,5:4,6:4,7:3,8:0,9:1,10:2,11:4,12:3,13:0,14:2
4	19MAT	2019-12-18	10	5	0:0,1:1,2:2,3:1,4:4,5:3,6:2,7:0,8:3,9:1
5	19ESC	2019-11-30	6	4	0:0,1:3,2:1,3:3,4:0,5:2
6	18DBM	2018-11-28	10	5	0:1,1:2,2:3,3:4,4:0,5:4,6:2,7:1,8:3,9:0

Fig 7.16 OMRDETAILS Table

SQLiteStudio (3.2.1) - [RESULT (EXAMOMR)]

Database Structure View Tools Help

Databases EXAMOMR (SQLite 3)

Filter by name

Tables (4)

- OMRDETAILS
- REGISTER
- RESULT**
- STUDENT

Views

Grid view Form view

	Examcode	Studentid	Edate	Percenta	Result	Omrshs	Evadate	Resultid
1	18DBM	17CS01	2018-04-28	80	Pass	C:/Us...	2020-06...	18DBM17CS01
2	18DBM	17CS02	2018-04-28	50	Pass	C:/Us...	2020-06...	18DBM17CS02
3	18DBM	17CS03	2018-04-28	30	Fail	C:/Us...	2020-06...	18DBM17CS03
4	18DBM	17CS04	2018-04-28	90	Pass	C:/Us...	2020-06...	18DBM17CS04
5	19ESC	18CS01	2019-05-19	100	Pass	C:/Us...	2020-06...	19ESC18CS01
6	19ESC	18CS02	2019-05-19	83.33	Pass	C:/Us...	2020-06...	19ESC18CS02
7	19ESC	18CS03	2019-05-19	83.33	Pass	C:/Us...	2020-06...	19ESC18CS03
8	19ESC	18EC01	2019-05-19	50	Pass	C:/Us...	2020-06...	19ESC18EC01
9	19ESC	18EC02	2019-05-19	83.33	Pass	C:/Us...	2020-06...	19ESC18EC02
10	19ESC	18IT02	2019-05-19	66.67	Pass	C:/Us...	2020-06...	19ESC18IT02
11	19ESC	18IT01	2019-05-19	50	Pass	C:/Us...	2020-06...	19ESC18IT01
12	19MAT	19CS01	2019-05-18	50	Pass	C:/Us...	2020-06...	19MAT19CS01
13	19MAT	19ME02	2019-05-18	100	Pass	C:/Us...	2020-06...	19MAT19ME02
14	19MAT	19ME03	2019-05-18	50	Pass	C:/Us...	2020-06...	19MAT19ME03
15	19MAT	19CS03	2019-05-18	80	Pass	C:/Us...	2020-06...	19MAT19CS03
16	19MAT	19CS02	2019-05-18	60	Pass	C:/Us...	2020-06...	19MAT19CS02
17	19PHY	19CS01	2019-05-16	73.33	Pass	C:/Us...	2020-06...	19PHY19CS01
18	19PHY	19CS02	2019-05-16	73.33	Pass	C:/Us...	2020-06...	19PHY19CS02

Fig 7.17 RESULT Table

SQLiteStudio (3.2.1) - [REGISTER (EXAMOMR)]

Database Structure View Tools Help

Databases EXAMOMR (SQLite 3)

Filter by name

Tables (4)

- OMRDETAILS
- REGISTER**
- RESULT
- STUDENT

Views

Grid view Form view

	Name	Email	Phone	Username	Password
1	Krishna	mhrk123@gmail.com	7898683848	Rama	rama123
2	Vardhan	bala456@yahoo.com	9089786756	Bala	bala456
3	Sriram	ram789@gmail.com	8978654354	Sriram	sriram

Fig 7.18 REGISTER Table