



UGC Autonomous College | Accredited by NAAC | Accredited by NBA

Project Report

On

“EXAMINATION MANAGEMENT SYSTEM”

Submitted in the Partial fulfilment of the requirement for the Award of Degree of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

(2020-2024)

Submitted to:

Er. Ajay Sharma (Associate Professor)

Er. Neha (Assistant Prof.)

Er. Anamika Jain (Assistant Prof.)

Submitted by:

Kartik Arora (2132015)

Rohan Sharma (2000184)

Rohin Biyal (2000185)

Rudra Pratap Samal (2000189)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Amritsar Group of Colleges, Amritsar

ACKNOWLEDGEMENT

It is our proud privilege to release the feelings of our gratitude to several persons who helped us directly or indirectly to conduct this Analysis Project Work. We express our heart full thanks and owe a deep sense of gratitude to our teacher and my faculty guide ***Er. Ajay Sharma (Associate Professor, Department of CSE, Amritsar Group of Colleges, Amritsar)*** and ***Er. Neha Chadha (Assistant Professor, Department of CSE, Amritsar Group of Colleges, Amritsar)***, for his guidance and inspiration in completing this project.

We are extremely thankful to ***Dr. Sandeep Kad (Head of Department, Department of Computer Science & Engineering, Amritsar Group of Colleges)*** and all faculty members of CSE Department at Amritsar Group of Colleges, Amritsar for their co-ordination and co-operation and for their kind guidance and encouragement.

We also thank all our friends who have more or less contributed to the preparation of this project Report, we will be always indebted to them. This project completion has indeed helped us explore more knowledge avenues related to RDBMS/Python and we are sure it will help us in future too.

DECLARATION

We, hereby, declare that the project work entitled *EXAMINATION MANAGEMENT SYSTEM* submitted to **Department of Computer Science & Engineering, AGC – Amritsar Group of Colleges**, is a record of an original work done by us under the guidance of **Er. Ajay Sharma, Er. Neha Chadha, Er. Anamika Jain** and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Kartik Arora
Univ Roll No: 2132015

Rohan Sharma
Univ Roll No: 2000184

Rohin Biyal
Univ Roll No: 2000185

Rudra Pratap Samal
Univ Roll No: 2000189

TABLE OF CONTENTS

Sr. No.	Content	Page No.
1	Introduction to RDBMS	1
2	Introduction to Python	3
3	Introduction of the Project	4
4	Objectives	5
5	Tools Used	6
6	Modules Used	8
7	System Analysis and Data Tables	10
8	SourceCode	12
9	Screenshots	43
10	References	50
11	Future Scope	51

INTRODUCTION TO RDBMS

Relational Data Model has an advantage that it is simple to implement and easy to understand as it uses table format. In this approach, a relation is only constructed by setting the association among the attributes of an entity as well the relationship among different entities. One of the main reasons for introducing this model was to increase the productivity of the application programmers by eliminating the need to change application program, when a change is made to the database. Data structure used in the data model is represented by both entities and relationship between them. Information is represented in the relational model in shape of tables. There are columns in a table which represent the attributes of an entity about which the table is constructed. The rows of a table are referred to as tuples.

1.1. STRUCTURED QUERY LANGUAGE

SQL tables, which consists of rows and columns, are used for storing data. The columns refer to the attributes. Each column in a table has a column name and a data type. A value's data type associates a fixed set of properties with a value. The data types available in Oracle fall under the following categories.

Category	Available data types
Character	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR2, LONG, RAW, LONGRAW
Number	Number
Date/Time	Date
LOB's	BFILE, BLOB CLOB, NCLOB

WHAT IS A TABLE ?

A table is a database object which is used to store data in relational databases. Each table consists of rows and columns. A column in the database table represents the table's attributes and a row represents a single set of column values in a database table. Each column of the table has a column name and a data type associated with it. The data types which can be used include VARCHAR2, NUMBER, DATE etc. A row of a table is also known as record. Within a table foreign keys are used to represent relationships.

CREATING A TABLE

In order to store and manage data it is necessary to create tables. In Oracle, tables are created using CREATE TABLE command which is one of the important DDL statement. The CREATE TABLE command specifies the name of the table, name of columns in the table as well as the data types and if required constraints associated with each column of the table.

INTEGRITY CONSTRAINTS IN CREATE TABLE

Oracle uses integrity constraints to prevent invalid data entry into the base tables of the database. One can define integrity constraints to enforce the business rules that one wants to associate with the information in the database. If the integrity constraints are violated due to the execution of any of DML statements (Insert, Update, Select) then Oracle rolls back the statement and returns an error. The different kinds of constraints are:

- Primary Key Constraint
- Unique Key Integrity Constraint
- NOT NULL Integrity Constraint
- Foreign Key Constraint
- Check Integrity Constraint

INTRODUCTION TO PYTHON

Python is a general purpose programming language that is often applied in scripting roles.

It is also called as Interpreted language.

The Origin of Python:

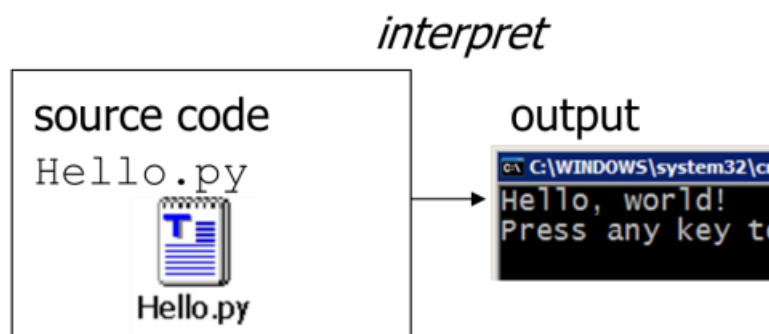
It is invented in Netherlands, in early 90's and its implementation was started in December 1989.

Guido Van Rossum is a fan of 'Monty Python's Flying Circus', famous TV show in Netherlands. So he named the language after Monty Python.

Features Of Python:

- It's Free: Downloading and installing Python is free and easy. Source code is easily accessible.
- Its Portable: Python runs virtually on every major platform used today. Programs runs exactly in the same manner irrespective of platform.
- It's Powerful:: Dynamic typing , Built-in types and tools , library utilities , third-party utilities(e.g. Numpy , Scipy)
- It's Mixable: Integration of python with other languages is widely used .
- It's Relatively Easy to Use: Python Programs are compiled automatically to an intermediate form called bytecode , which the interpreter then reads.
- It's Relatively Easy to Learn
- Its Object – Oriented and Functional

Python is instead directly interpreted into machine instructions.



INTRODUCTION OF THE PROJECT

This project aims at building an **Examination Management System**. It is divided into different modules to make it more user-friendly. The front –end is designed using Python3 – PyQt5 Tool.

This project has been developed to override the problems of prevailing in the practicing manual system. This software is supported to eliminate and reduce the hardships faced by the existing system. Moreover this is designed for the particular need of the institute to carry out operations in a smooth and effective manner.

The purpose of the Examination Management System is to automate the existing manual system by the help of the computerized equipment and full-fledged computer software, fulfilling their requirements so that the valuable data/information can be stored for a long period with easy accessing and manipulation of the same.

The main objective of the Project on Examination Management System is to manage the details of Exams, Students, Courses, Semesters and the Subjects. It can manage all the information about students' academic details along with their required information.

The project is built with administrative authorization where multiple users with different roles can access and manipulate the data/information according to their roles.

Functionalities provided by Examination Management System:

- Manages the users such as DEOs, Evaluators and Admins, only by authorized access.
- Manages the data/information of the students (Roll no., Name, Course, Semester and their results)
- Adding, Updating and Deletions are possible in different modules according to the necessities.
- To increase the efficiency of managing the exams and results in an institution.

OBJECTIVES OF THE PROJECT

- Authorized LogIn can create, update or delete the users presently working with the system.
- Authorized LogIn also has full access of all the modules which again gives it full authorization to add, update or delete in students, subjects, exams and result related details.
- This system is built for only for the users who will manage the pre-examination data and post-examination data.
- Pre-examination refers to selection of subjects and generation of date-sheets of the examination for each course and semester currently present.
- For pre-examination, if any changes in subjects or date-sheets need data to be updated which is already stored, then only Admins with any designation of the institution will have right to do so.
- For post-examination, if anyhow result which is stored or marks which are generated are not valid or need to be changed, authorized LogIn like DEOs with any designation of the Institute can update the data.
- All users are provided a feature to update their profile to some extent.
- Whereas Admin with any designation, of the Institute can update the details of the other users which are not accessible by those.
- Each module provides a well design to show, create, update and delete the contents related to the relations required in this system, whereas some modules don't have few feature like creating and deleting content such as creating subjects or deleting students' details.
- Date-sheet Generator is an excellent feature of this system where any user with Admin role can generate date-sheets of each semester with one click.
- The Main Window may supposed to have one Mask Layer Button which will mask the whole window for the user, if it is needed and then it can only be unmasked if provided right credentials (password) for that User.

TOOLS USED

Front-end: Python PyQt5

Back-end: Oracle Database

3.1 Introduction to PyQt5

PyQt is a Python library for creating GUI applications using the Qt toolkit. Created by Riverbank Computing, PyQt is free software (GPL licensed) and has been in development since 1999. PyQt5 was released in 2016 and last updated in October 2021. PyQt5 is the Qt5-based edition of the Python GUI library PyQt from Riverbank Computing. PyQt which consists of more than six hundred classes covering a range of features such as

- Graphical User Interfaces
- SQL Databases
- Web toolkits
- XML processing
- Networking

These features can be combined to create advanced UIs as well as standalone applications. A lot of major companies across all industries use Qt. Some examples are LG, Mercedes, AMD, Panasonic, Harman, etc.



3.2.Introduction to Oracle Database

Oracle database (Oracle DB) is a relational database management system (RDBMS) from the Oracle Corporation. Originally developed in 1977 by Lawrence Ellison and other developers, Oracle DB is one of the most trusted and widely-used relational database engines.

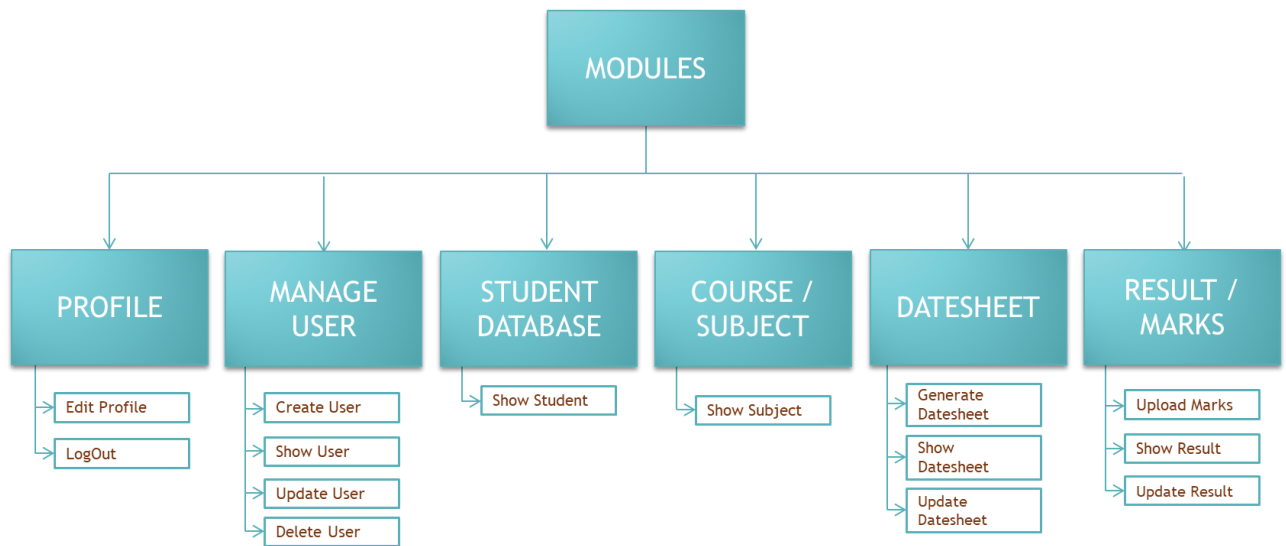
The system is built around a relational database framework in which data objects may be directly accessed by users (or an application front end) through structured query language (SQL). Oracle is fully scalable relational database architecture and is often used by global enterprises, which manage and process data across wide and local area networks. The Oracle database has its own network component to allow communications across networks.

A key feature of Oracle is that its architecture is split between the logical and the physical. This structure means that for large-scale distributed computing, also known as grid computing, the data location is irrelevant and transparent to the user, allowing for a more modular physical structure that can be added to and altered without affecting the activity of the database, its data or users. The sharing of resources in this way allows for very flexible data networks whose capacity can be adjusted up or down to suit demand, without degradation of service. It also allows for a robust system to be devised as there is no single point at which a failure can bring down the database, as the networked schema of the storage resources means that any failure would be local only.



MODULES USED

Following are the modules developed in the project:



- **Profile Module**

All users are provided a feature to update their profile to some extent.

- **Manage User**

Authorized LogIn can create, update or delete the users presently working with the system. Whereas Admin with any designation, of the Institute can update the details of the other users which are not accessible by those

- **Student Database**

Authorized LogIn also has full access of all the modules which again gives it full authorization to add, update or delete in students, subjects, exams and result related details. Each module provides a well design to show, create, update and delete the contents related to the relations required in this system, whereas some modules don't have few feature like creating and deleting content such as creating subjects or deleting students' details. Pre-examination refers to selection of subjects and generation of date-sheets of the examination for each course and semester currently present. Post-examination refers to generation of marks and results of each student, maybe based upon MCQs which can use OMR to generate result, or descriptive examination which needs to upload result manually by DEO.

- **Course/Subject**

For pre-examination, if any changes in subjects or date-sheets need data to be updated which is already stored, then only Admins with any designation of the institution will have right to do so.

- **Datesheet**

For pre-examination, if any changes in subjects or date-sheets need data to be updated which is already stored, then only Admins with any designation of the institution will have right to do so.

- **Result/ Marks**

Post-examination refers to generation of marks and results of each student, maybe based upon MCQs which can use OMR to generate result, or descriptive examination which needs to upload result manually by DEO. For post-examination, if anyhow result which is stored or marks which are generated are not valid or need to be changed, authorized LogIn like DEOs with any designation of the Institute can update the data.

SYSTEM ANALYSIS AND DATA TABLES

- **USER**

COLUMN NAME	TYPE	CONSTRAINTS
USERNAME	VARCHAR2(10)	PRIMARY KEY
PASSWORD	VARCHAR2(12)	NOT NULL
NAME	VARCHAR2(20)	NOT NULL
DESIGNATION	VARCHAR2(15)	NOT NULL
ROLE	VARCHAR2(15)	NOT NULL
CONTACT	NUMBER	UNIQUE

- **STUDENT**

COLUMN NAME	TYPE	CONSTRAINTS
ROLL NO	NUMBER(6)	PRIMARY KEY
SNAME	VARCHAR2(30)	NOT NULL
COURSE	VARCHAR2(15)	NOT NULL
SEM	NUMBER	CHECK (SEM BETWEEN 1 AND 8)
BATCH	NUMBER(6)	CHECK (BATCH>2000)

- **SUBJECT**

Each semester of each course will have one table for the subjects

COLUMN NAME	TYPE	CONSTRAINTS
SUB	VARCHAR2(20)	NOT NULL
SUBCODE	VARCHAR2(10)	PRIMARY KEY

• DATESHEET

COLUMN NAME	TYPE	CONSTRAINTS
DATE	DATE	NOT NULL
SUB	VARCHAR2(20)	NOT NULL
SUBCODE	VARCHAR2(10)	REFERENCES SUBJECT(SUBCODE)

• MARKS/RESULT

Each semester of each course will have one table for the result of the students

COLUMN NAME	TYPE	CONSTRAINTS
ROLLNO	NUMBER(6)	REFERENCES STUDENT(ROLLNO)
SUBCODE1	NUMBER	CHECK (SUBCODE1<=100)
SUBCODE2	NUMBER	CHECK (SUBCODE2<=100)
SUBCODE3	NUMBER	CHECK (SUBCODE3<=100)
SUBCODE4	NUMBER	CHECK (SUBCODE4<=100)
SUBCODE5	NUMBER	CHECK (SUBCODE5<=100)
TOTAL	NUMBER	CHECK (TOTAL<=100)

- Containing details about all the users present who can retrieve or manipulate data from other tables
- And this table is only accessible by user - SUPERVISOR

USER'S TABLE



- Containing details about all the students enrolled in the institution

STUDENT TABLE



- Containing the details about all the subjects in each semester of every course

COURSE/SUBJECT TABLE



- Containing details about the exam's datesheet which has been generated by any Admin

DATESHEET TABLE



- Containing details about the marks of the students which would be used to generate result according to the user's need

RESULT/MARKS



SOURCECODE

```
import sys
import cx_Oracle
from PyQt5 import QtCore,QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from qt_material import apply_stylesheet
from random import *
from datetime import *

class mainWindow(QStackedWidget):

    def __init__(self):
        QWidget.__init__(self)

        self.setWindowTitle("Examination Management System")
        self.setFixedSize(1000,600)
        self.setWindowIcon(QIcon('2094145_1.jpg'))

        self.ad=QWidget(self)
        self.addWidget(self.ad)

        self.loginWidget=QWidget(self.ad)

        self.mw=QWidget(self)
        self.addWidget(self.mw)

        self.uiMain=QStackedWidget(self.mw)
        self.uiMain.setGeometry(50,0,700,500)

        self.bla=QWidget(self.uiMain)
        self.uiMain.addWidget(self.bla)

        self.pfst=QWidget(self.uiMain)
        self.uiMain.addWidget(self.pfst)

        self.crUser=QWidget(self.uiMain)
        self.uiMain.addWidget(self.crUser)

        self.shUser=QWidget(self.uiMain)
        self.uiMain.addWidget(self.shUser)

        self.upUser=QWidget(self.uiMain)
        self.uiMain.addWidget(self.upUser)

        self.deUser=QWidget(self.uiMain)
        self.uiMain.addWidget(self.deUser)

        self.crStudent=QWidget(self.uiMain)
        self.uiMain.addWidget(self.crStudent)

        self.shStudent=QWidget(self.uiMain)
        self.uiMain.addWidget(self.shStudent)

        self.upStudent=QWidget(self.uiMain)
```



```

self.uiMain.addWidget(self.upStudent)

self.deStudent=QWidget(self.uiMain)
self.uiMain.addWidget(self.deStudent)

self.shSub=QWidget(self.uiMain)
self.uiMain.addWidget(self.shSub)

self.gDate=QWidget(self.uiMain)
self.uiMain.addWidget(self.gDate)

self.sDate=QWidget(self.uiMain)
self.uiMain.addWidget(self.sDate)

self.uplResult=QWidget(self.uiMain)
self.uiMain.addWidget(self.uplResult)

self.shResult=QWidget(self.uiMain)
self.uiMain.addWidget(self.shResult)

self.updResult=QWidget(self.uiMain)
self.uiMain.addWidget(self.updResult)

self.logIn()

def logIn(self):

    self.setCurrentWidget(self.ad)

    self.adHBox=QHBoxLayout()
    self.ad.setLayout(self.adHBox)

    photo=QWidget(self.loginWidget)
    photo.setFixedSize(400,210)
    photo.setStyleSheet('background-color:none;')

    elements=QWidget(self.loginWidget)
    elements.setFixedSize(380,150)
    elements.setStyleSheet('background-color:none;')

    self.message=QWidget(self.loginWidget)
    self.message.setFixedSize(380,100)
    self.message.setStyleSheet('background-color:none;')

    blank=QWidget(self.ad)
    blank.setFixedSize(600,600)
    blank2=QWidget(self.ad)
    blank2.setFixedSize(400,200)
    blank2.setStyleSheet('background-color:none;')

    self.adHBox.addWidget(self.loginWidget)
    self.adHBox.addWidget(blank)

    self.loginWidget.setFixedSize(400,800)
    self.loginWidget.setStyleSheet('background: rgba(252, 255, 255, 0.85);'

```

```

        'border-radius: 50px;')

self.pic=QPixmap('pic180.png')
self.profile=QLabel(photo)
self.profile.setPixmap(self.pic)
self.profile.resize(200,200)
self.profile.move(100,20)

self.userLabel = QLabel(elements)
self.userLabel.setText('UserName')
self.userLabel.setStyleSheet('color: #31363B;'
                              'font-weight: bold')
#self.userLabel.move(350,220)

self.passLabel = QLabel(elements)
self.passLabel.setText('Password')
self.passLabel.setStyleSheet('color: #31363B;'
                              'font-weight: bold')
#self.passLabel.move(350,260)

self.userLine = QLineEdit(elements)
self.userLine.setStyleSheet('background-color: #31363B;'
                             'font-weight: bold;'
                             'border-radius:5;')
#self.userLine.move(450, 215)
#self.userLine.resize(200, 25)

self.passLine = QLineEdit(elements)
#self.passLine.move(450, 255)
self.passLine.setEchoMode(QLineEdit.Password)
#self.passLine.resize(200, 25)
self.passLine.setStyleSheet('background-color: #31363B;'
                             'font-weight: bold;'
                             'border-radius:5px')
self.passLine.returnPressed.connect(self.getCred)

self.loginButton=QPushButton(elements)
#self.loginButton.move(450,300)
self.loginButton.setText("LogIn")
#self.loginButton.adjustSize()
self.loginButton.setStyleSheet('background-color: #31363B;'
                                'border-radius:7px')
self.loginButton.clicked.connect(self.getCred)

self.resetButton=QPushButton(elements)
#self.resetButton.move(530,300)
self.resetButton.setText("Reset")
self.resetButton.setStyleSheet('background-color: #31363B;'
                                'border-radius:7px')
#self.resetButton.adjustSize()
self.resetButton.clicked.connect(self.clearPass)

self.loginWidgetVBox=QVBoxLayout()
self.loginWidget.setLayout(self.loginWidgetVBox)

```

```

elementsGrid=QGridLayout()
elements.setLayout(elementsGrid)

self.messageHBox=QHBoxLayout()
self.message.setLayout(self.messageHBox)

self.loginWidgetVBox.addWidget(photo)
self.loginWidgetVBox.addWidget(elements)
self.loginWidgetVBox.addWidget(self.message)
self.loginWidgetVBox.addWidget(blank2)

elementsGrid.addWidget(self.userLabel,0,0)
elementsGrid.addWidget(self.userLine,0,1,1,2)
elementsGrid.addWidget(self.passLabel,1,0)
elementsGrid.addWidget(self.passLine,1,1,1,2)
elementsGrid.addWidget(self.loginButton,2,1)
elementsGrid.addWidget(self.resetButton,2,2)

return

def getCred(self):
    self.user=self.userLine.text()
    self.pas=self.passLine.text()
    self.checkCred()

def checkCred(self):
    try:
        self.con=cx_Oracle.connect(self.user,self.pas)
    except cx_Oracle.DatabaseError:
        self.wrongPass()
    else:
        self.clearPass()
        self.cur=self.con.cursor()
        self.ui()

def wrongPass(self):

    self.resetButton.setEnabled(False)
    self.loginButton.setEnabled(False)
    self.userLine.setEnabled(False)
    self.passLine.setEnabled(False)

    self.msg=QLabel()
    self.msg.setText('Wrong UserName/Password !')
    self.msg.setStyleSheet('color: #31363B')
    #self.msg.resize(320,20)
    #self.msg.move(20,500)

    self.retry=QPushButton()
    self.retry.setText('Retry')
    self.retry.setStyleSheet('background-color: #31363B;'
                             'border-radius:7px')
    #self.retry.move(460,440)
    #self.retry.resize(80,40)

```

```

        self.retry.clicked.connect(self.retryPass)

        self.messageHBox.addWidget(self.msg)
        self.messageHBox.addWidget(self.retry)

    return

def clearPass(self):

    self.userLine.clear()
    self.passLine.clear()

def retryPass(self):

    self.resetButton.setEnabled(True)
    self.loginButton.setEnabled(True)
    self.userLine.setEnabled(True)
    self.passLine.setEnabled(True)

    self.userLine.clear()
    self.passLine.clear()

    self.msg.hide()
    self.retry.hide()

def ui(self):

    self.setCurrentWidget(self.mw)
    self.uiMain.setCurrentWidget(self.bla)

    self.mainPic=QPixmap('pic50.png')
    self.mainProfile=QLabel(self.mw)
    self.mainProfile.setPixmap(self.mainPic)
    self.mainProfile.resize(50,50)
    self.mainProfile.setAlignment(Qt.AlignRight)
    self.mainProfile.setStyleSheet("padding-right: 5;"
                                    "background-color:none;")

    self.profileCur=self.con.cursor()
    query='select name,designation,contact from projectuser where username=:username'
    self.profileCur.execute(query,(self.user,))
    profileData=self.profileCur.fetchall()

    self.profileMenu=QComboBox(self.mw)
    menu=[profileData[0][0],'Profile','LogOut']
    self.profileMenu.addItem(menu)
    self.profileMenu.adjustSize()
    self.profileMenu.setCurrentIndex(0)
    self.profileMenu.setStyleSheet("background-color: #232629;"
                                    "border-radius:5;")
    self.profileMenu.activated.connect(self.userSett)

    self.userTable=QComboBox(self.mw)
    users=['Manage Users','Create User','Show Users','Update Users','Delete User']

```

```

self.userTable.addItem(users)
self.userTable.setStyleSheet('background-color: #232629;'
                              'border-radius:5')
self.userTable.activated.connect(self.userMenu)

self.stuTable=QComboBox(self.mw)
students=['Student Database','Show Student Details']
self.stuTable.addItem(students)
self.stuTable.setStyleSheet('background-color: #232629;'
                              'border-radius:5')
self.stuTable.activated.connect(self.stuMenu)

self.subTable=QComboBox(self.mw)
subjects=['Course/Subject','Show Course/Subjects']
self.subTable.addItem(subjects)
self.subTable.setStyleSheet('background-color: #232629;'
                              'border-radius:5')
self.subTable.activated.connect(self.subMenu)

self.datesheet=QComboBox(self.mw)
datesheet=['Datesheet','Generate Datesheet','Show Datesheet','Update Datesheet']
self.datesheet.addItem(datesheet)
self.datesheet.setStyleSheet('background-color: #232629;'
                              'border-radius:5')
self.datesheet.activated.connect(self.dateMenu)

self.resultTable=QComboBox(self.mw)
result=['Result/Marks','Upload Marks','Show Result','Update Result']
self.resultTable.setStyleSheet('background-color: #232629;'
                              'border-radius:5')
self.resultTable.addItem(result)
self.resultTable.activated.connect(self.resultMenu)

self.grid=QGridLayout()
self.mw.setLayout(self.grid)

self.headOut=QWidget(self.mw)
self.headOut.setFixedSize(1100,120)
self.headOut.setStyleSheet('background: rgba(255, 255, 255, 0.85);'
                              'border-radius:50px')

self.head=QWidget(self.headOut)
self.head.setGeometry(40,0,950,120)
self.head.setStyleSheet('background-color:none;')

self.headGrid=QGridLayout()
self.head.setLayout(self.headGrid)

self.grid.addWidget(self.headOut,0,0)
self.grid.addWidget(self.uiMain,1,0)

self.headGrid.addWidget(self.mainProfile,0,3)
self.headGrid.addWidget(self.profileMenu,0,4)
self.headGrid.addWidget(self.userTable,2,0)
self.headGrid.addWidget(self.stuTable,2,1)

```

```

self.headGrid.addWidget(self.subTable,2,2)
self.headGrid.addWidget(self.datesheet,2,3)
self.headGrid.addWidget(self.resultTable,2,4)

#self.mw.show()

def profileSettings(self):
    self.uiMain.setCurrentWidget(self.pfst)
    #self.uiMain.setStyleSheet('padding-top:0')
    self.pfst.setFixedSize(800,300)

    self.profileCur=self.con.cursor()
    query='select name,designation,contact from projectuser where username=:username'
    self.profileCur.execute(query,(self.user,))
    profileData=self.profileCur.fetchall()

    psPic=QPixmap('pic150.png')
    userPic=QLabel(self.pfst)
    userPic.setPixmap(psPic)
    userPic.setStyleSheet('padding-left:30')

    ""uName=QLabel(self.pfst)
    uName.setText(profileData[0][0])
    uName.setStyleSheet('padding-top:1')

    uDesg=QLabel(self.pfst)
    uDesg.setText(profileData[0][1])
    uDesg.setStyleSheet('padding-top:30')

    uContact=QLabel(self.pfst)
    uContact.setText(str(profileData[0][2]))
    uContact.setStyleSheet('padding-top:60')""

    name=QLabel(self.pfst)
    name.setText('Name')
    name.setStyleSheet('padding-left:150')

    self.nameLine=QLineEdit(self.pfst)
    self.nameLine.setText(profileData[0][0])
    self.nameLine.setEnabled(False)

    desg=QLabel(self.pfst)
    desg.setText('Designation')
    desg.setStyleSheet('padding-left:150')

    self.desgLine=QLineEdit(self.pfst)
    self.desgLine.setText(profileData[0][1])
    self.desgLine.setEnabled(False)

    contact=QLabel(self.pfst)
    contact.setText('Contact')
    contact.setStyleSheet('padding-left:150')

    self.conLine=QLineEdit(self.pfst)
    self.conLine.setText(str(profileData[0][2]))

```

```

self.conLine.setEnabled(False)

self.edit=QPushButton(self.pfst)
self.edit.setText('Edit')
self.edit.adjustSize()
self.edit.clicked.connect(self.editProfile)

self.cancel=QPushButton(self.pfst)
self.cancel.setText('Cancel')
self.cancel.adjustSize()
self.cancel.setEnabled(False)
self.cancel.clicked.connect(self.canProfile)

self.submit=QPushButton(self.pfst)
self.submit.setText('Submit')
self.submit.adjustSize()
self.submit.setEnabled(False)
self.submit.clicked.connect(self.subProfile)

umainGrid=QGridLayout()
self.pfst.setLayout(umainGrid)

for i in reversed(range(umainGrid.count())):
    umainGrid.itemAt(i).widget().setParent(None)

umainGrid.addWidget(userPic,0,0,3,0)
umainGrid.addWidget(uName,3,0)
umainGrid.addWidget(uDesg,4,0)
umainGrid.addWidget(uContact,5,0)
umainGrid.addWidget(name,0,1)
umainGrid.addWidget(desg,1,1)
umainGrid.addWidget(contact,2,1)
umainGrid.addWidget(self.nameLine,0,2,1,4)
umainGrid.addWidget(self.desgLine,1,2,1,4)
umainGrid.addWidget(self.conLine,2,2,1,4)
umainGrid.addWidget(self.edit,3,2)
umainGrid.addWidget(self.cancel,3,4)
umainGrid.addWidget(self.submit,3,5)

def userMenu(self):
    if self.userTable.currentIndex()==1:
        self.createUser()
        self.userTable.setCurrentIndex(0)
    elif self.userTable.currentIndex()==2:
        self.showUser()
        self.userTable.setCurrentIndex(0)
    elif self.userTable.currentIndex()==3:
        self.updateUser()
        self.userTable.setCurrentIndex(0)
    elif self.userTable.currentIndex()==4:
        self.deleteUser()
        self.userTable.setCurrentIndex(0)

def createUser(self):
    self.uiMain.setCurrentWidget(self.crUser)

```

```

self.crUser.setFixedSize(700,380)

self.crUserMsg=QWidget()
self.crUserMsg.setFixedSize(700,100)

self.crUserName=QLabel(self.crUser)
self.crUserName.setText('UserName')

crPass=QLabel(self.crUser)
crPass.setText('Password')

crName=QLabel(self.crUser)
crName.setText('Full Name')

crRole=QLabel(self.crUser)
crRole.setText('Role')

crDesg=QLabel(self.crUser)
crDesg.setText('Designation')

crCont=QLabel(self.crUser)
crCont.setText('Contact')

self.crUserLine=QLineEdit(self.crUser)
self.crPassLine=QLineEdit(self.crUser)
self.crPassLine.setEchoMode(QLineEdit.Password)
self.crNameLine=QLineEdit(self.crUser)
self.crDesgLine=QLineEdit(self.crUser)
self.crContLine=QLineEdit(self.crUser)

self.crRoleList=QComboBox(self.crUser)
roles=['Admin','DEO','Evaluator','Role4']
self.crRoleList.addItem(roles)
self.crRoleText=self.crRoleList.itemText(0)
self.crRoleList.activated.connect(self.createRole)

crCreate=QPushButton(self.crUser)
crCreate.setText('Create')
crCreate.setStyleSheet('font-weight:bold')
crCreate.clicked.connect(self.newUser)

crCancel=QPushButton(self.crUser)
crCancel.setText('Cancel')
crCancel.setStyleSheet('font-weight:bold')
crCancel.clicked.connect(self.cancelCrUser)

self.crUserGrid=QGridLayout()
self.crUser.setLayout(self.crUserGrid)

self.crUserMsgGrid=QGridLayout()
self.crUserMsg.setLayout(self.crUserMsgGrid)

self.crUserGrid.addWidget(self.crUserName,0,0)
self.crUserGrid.addWidget(self.crUserLine,1,0,1,2)
self.crUserGrid.addWidget(crPass,0,2,1,2)
self.crUserGrid.addWidget(self.crPassLine,1,2,1,2)
self.crUserGrid.addWidget(crName,2,0,1,2)

```



```

self.crUserGrid.addWidget(self.crNameLine,3,0,1,2)
self.crUserGrid.addWidget(crDesg,2,2,1,2)
self.crUserGrid.addWidget(self.crDesgLine,3,2,1,2)
self.crUserGrid.addWidget(crCont,4,0)
self.crUserGrid.addWidget(self.crContLine,5,0,1,2)
self.crUserGrid.addWidget(crRole,4,2,1,2)
self.crUserGrid.addWidget(self.crRoleList,5,2,1,2)
self.crUserGrid.addWidget(crCancel,6,0)
self.crUserGrid.addWidget(crCreate,6,3)
self.crUserGrid.addWidget(self.crUserMsg,7,0,1,5)

def createRole(self):
    if self.crRoleList.currentIndex()==0:
        self.crRoleText=self.crRoleList.itemText(0)
    elif self.crRoleList.currentIndex()==1:
        self.crRoleText=self.crRoleList.itemText(1)
    elif self.crRoleList.currentIndex()==2:
        self.crRoleText=self.crRoleList.itemText(2)
    elif self.crRoleList.currentIndex()==3:
        self.crRoleText=self.crRoleList.itemText(3)
    return

def cancelCrUser(self):
    self.crUserLine.clear()
    self.crPassLine.clear()
    self.crNameLine.clear()
    self.crDesgLine.clear()
    self.crContLine.clear()
    return

def newUser(self):
    try:
        for i in reversed(range(self.crUserMsgGrid.count())):
            self.crUserMsgGrid.itemAt(i).widget().setParent(None)
        user=self.crUserLine.text()
        pas=self.crPassLine.text()
        name=self.crNameLine.text()
        name=name.title()
        desg=self.crDesgLine.text()
        desg=desg.title()
        cont=self.crContLine.text()
        role=self.crRoleText
        crInsTab='insert into projectuser
values(:username,:password,:fullname,:designation,:role,:contact)'
        self.cur.execute(crInsTab,(user,pas,name,desg,role,cont))
    except cx_Oracle.IntegrityError as I:
        if str(I)=='ORA-00001: unique constraint (SUPERVISOR.SYS_C007058)
violated':
            print('erro')
            self.uMsg=QLabel()
            self.uMsg.setText('UserName Exists!')
            self.uMsg.setStyleSheet('color:red')
            self.crUserMsgGrid.addWidget(self.uMsg,0,0)
        if str(I)=='ORA-02290: check constraint (SUPERVISOR.SYS_C007066)
violated':

```

```

        self.cMsg=QLabel()
        self.cMsg.setText('Check Contact!')
        self.cMsg.setStyleSheet('color:red')
        self.crUserMsgGrid.addWidget(self.cMsg,1,0)
        if str(I)=='ORA-01400: cannot insert NULL into
("SUPERVISOR"."PROJECTUSER"."USERNAME")':
            self.nMsg=QLabel()
            self.nMsg.setText('Please fill the Input Fields!')
            self.nMsg.setStyleSheet('color:red')
            self.crUserMsgGrid.addWidget(self.nMsg,2,0)
        print(str(I))
    except cx_Oracle.DatabaseError:
        cMsg=QLabel()
        cMsg.setText('Contact contains characters rather than numbers!')
        cMsg.setStyleSheet('color:red')
        self.crUserMsgGrid.addWidget(cMsg,3,0)
    else:
        crQuery='create user '+user+' identified by '+pas
        self.cur.execute(crQuery)
        if self.crRoleText=='Admin':
            crGrant='grant all privileges to '+user
            self.cur.execute(crGrant)
        self.con.commit()
        self.cancelCrUser()
    return

```

```

def showUser(self):
    self.uiMain.setCurrentWidget(self.shUser)
    self.shUser.setFixedSize(700,450)

    self.shUserMain=QWidget()
    self.shUserMain.setMinimumSize(700,400)

    UserName=QLabel(self.shUser)
    UserName.setText('UserName')
    UserName.setStyleSheet('font-weight:bold')

    Role=QLabel(self.shUser)
    Role.setText('Role')
    Role.setStyleSheet('font-weight:bold')

    Name=QLabel(self.shUser)
    Name.setText('Full Name')
    Name.setStyleSheet('font-weight:bold')

    Desg=QLabel(self.shUser)
    Desg.setText('Designation')
    Desg.setStyleSheet('font-weight:bold')

    Cont=QLabel(self.shUser)
    Cont.setText('Contact')
    Cont.setStyleSheet('font-weight:bold')

    usersList=QListWidget(self.shUser)
    usersList.setGeometry(50,600,700,400)

```

```

self.shUserGrid=QGridLayout()
self.shUser.setLayout(self.shUserGrid)

self.shUserMainGrid=QGridLayout()
self.shUserMain.setLayout(self.shUserMainGrid)

self.shUserGrid.addWidget(UserName,0,0)
self.shUserGrid.addWidget(Name,0,1)
self.shUserGrid.addWidget(Desg,0,2)
self.shUserGrid.addWidget(Role,0,3)
self.shUserGrid.addWidget(Cont,0,4)
self.shUserGrid.addWidget(self.shUserMain,1,0,1,5)

cur=self.con.cursor()
cur.execute('select username,name,designation,role,contact from projectuser')
rec=cur.fetchall()
row=1
col=0

for i in reversed(range(self.shUserMainGrid.count())):
    self.shUserMainGrid.itemAt(i).widget().setParent(None)

#self.shUserMainGrid.addWidget(usersList,0,0)
for i in rec:
    col=0
    for j in i:
        self.shUserMainGrid.addWidget(QLabel(str(j)),row,col)
        col=col+1
    row=row+1
return

def updateUser(self):
    self.uiMain.setCurrentWidget(self.upUser)
    self.upUser.setFixedSize(700,300)

    search=QLabel(self.upUser)
    search.setText('Enter UserName')

    self.upSearchLine=QLineEdit(self.upUser)

    self.upSearchButton=QPushButton(self.upUser)
    self.upSearchButton.setText('Search')
    self.upSearchButton.adjustSize()
    self.upSearchButton.clicked.connect(self.upSearchUser)

    upName=QLabel(self.upUser)
    upName.setText('Full Name')

    oldName=QLabel(self.upUser)
    oldName.setText('Full Name')
    oldName.setStyleSheet('font-weight:bold')

    oldCont=QLabel(self.upUser)
    oldCont.setText('Contact')
    oldCont.setStyleSheet('font-weight:bold')

```

```

upDesg=QLabel(self.upUser)
upDesg.setText('Designation')

oldDesg=QLabel(self.upUser)
oldDesg.setText('Designation')
oldDesg.setStyleSheet('font-weight:bold')

upRole=QLabel(self.upUser)
upRole.setText('Role')

oldRole=QLabel(self.upUser)
oldRole.setText('Role')
oldRole.setStyleSheet('font-weight:bold')

self.upRoleList=QComboBox(self.upUser)
roles=['Admin','DEO','Evaluator','Role4']
self.upRoleList.addItem(roles)
self.upRoleList.activated.connect(self.upSelectRole)
self.upRoleList.setEnabled(False)

self.upRoleText=self.upRoleList.itemText(0)

self.upNameLine=QLineEdit(self.upUser)
self.upNameLine.setEnabled(False)

self.upDesgLine=QLineEdit(self.upUser)
self.upDesgLine.setEnabled(False)

self.upUpdate=QPushButton(self.upUser)
self.upUpdate.setText('Update')
self.upUpdate.clicked.connect(self.changeUser)

self.upCancel=QPushButton(self.upUser)
self.upCancel.setText('Cancel')
self.upCancel.adjustSize()
self.upCancel.setEnabled(False)
self.upCancel.clicked.connect(self.cancelUpUser)

self.upUserGrid=QGridLayout()
self.upUser.setLayout(self.upUserGrid)

self.upUserGrid.addWidget(search,0,0,1,2)
self.upUserGrid.addWidget(self.upSearchLine,0,2,1,3)
self.upUserGrid.addWidget(self.upSearchButton,0,5)
self.upUserGrid.addWidget(oldName,1,0,1,2)
self.upUserGrid.addWidget(oldDesg,2,0,1,2)
self.upUserGrid.addWidget(oldRole,3,0,1,2)
self.upUserGrid.addWidget(oldCont,4,0,1,2)
self.upUserGrid.addWidget(upName,5,0,1,2)
self.upUserGrid.addWidget(upDesg,5,3,1,3)
self.upUserGrid.addWidget(upRole,5,2)
self.upUserGrid.addWidget(self.upNameLine,6,0,1,2)
self.upUserGrid.addWidget(self.upRoleList,6,2)
self.upUserGrid.addWidget(self.upDesgLine,6,3,1,3)
self.upUserGrid.addWidget(self.upUpdate,7,5)
self.upUserGrid.addWidget(self.upCancel,7,0)

```

```

def upSearchUser(self):
    user=self.upSearchLine.text()
    self.cur.execute('select name,designation,role,contact from projectuser where
username=(%s)',(user,))
    rec=self.cur.fetchall()
    print(len(rec))
    noUser=QLabel(self.upUser)
    if len(rec)==0:
        noUser.setText('*UserName does not exist!')
        noUser.setStyleSheet('color:red')
        self.upUserGrid.addWidget(noUser,8,0,1,4)
    else:
        self.upCancel.setEnabled(True)
        self.upSearchButton.setEnabled(False)
        noUser.clear()
        l=[]
        for i in rec:
            l.append(i[0])
            l.append(i[1])
            l.append(i[2])
            l.append(i[3])

        self.oldNameValue=QLabel(self.upUser)
        self.oldDesgValue=QLabel(self.upUser)
        self.oldRoleValue=QLabel(self.upUser)
        self.oldContValue=QLabel(self.upUser)

        if self.oldNameValue.text()!=None:
            self.oldNameValue.clear()
            self.oldDesgValue.clear()
            self.oldRoleValue.clear()
            self.oldContValue.clear()

        self.oldNameValue.setText(l[0])
        self.oldDesgValue.setText(l[1])
        self.oldRoleValue.setText(l[2])
        self.oldContValue.setText(str(l[3]))

        self.upUserGrid.addWidget(self.oldNameValue,1,2)
        self.upUserGrid.addWidget(self.oldDesgValue,2,2)
        self.upUserGrid.addWidget(self.oldRoleValue,3,2)
        self.upUserGrid.addWidget(self.oldContValue,4,2)

        self.upNameLine.setEnabled(True)
        self.upDesgLine.setEnabled(True)
        self.upRoleList.setEnabled(True)

        self.upNameLine.setText(l[0])
        self.upDesgLine.setText(l[1])

        if self.oldRoleValue.text()=='Admin':
            self.upRoleList.setCurrentIndex(0)
        elif self.oldRoleValue.text()=='DEO':
            self.upRoleList.setCurrentIndex(1)
        elif self.oldRoleValue.text()=='Evaluator':

```

```

        self.upRoleList.setCurrentIndex(2)
    elif self.oldRoleValue.text()=='Role4':
        self.upRoleList.setCurrentIndex(3)
    return

def upSelectRole(self):
    if self.upRoleList.currentIndex()==0:
        self.upRoleText=self.upRoleList.itemText(0)
    elif self.upRoleList.currentIndex()==1:
        self.upRoleText=self.upRoleList.itemText(1)
    elif self.upRoleList.currentIndex()==2:
        self.upRoleText=self.upRoleList.itemText(2)
    elif self.upRoleList.currentIndex()==3:
        self.upRoleText=self.upRoleList.itemText(3)
    return

def changeUser(self):
    try:
        user=self.upSearchLine.text()
        name=self.upNameLine.text()
        name=name.title()
        desg=self.upDesgLine.text()
        desg=desg.title()
        role=self.upRoleText
        self.cur.execute('update projectuser set
name=:fullname,designation=:designation,role=:role where
username=:username',(name,desg,role,user))
    except:
        print('error occured')
    else:
        self.con.commit()
        self.upSearchButton.setEnabled(True)
        self.upCancel.setEnabled(False)
        self.upSearchLine.clear()
        self.oldNameValue.clear()
        self.oldDesgValue.clear()
        self.oldRoleValue.clear()
        self.oldContValue.clear()
        self.upNameLine.clear()
        self.upDesgLine.clear()
        self.upRoleList.clear()
    return

def cancelUpUser(self):
    self.upSearchButton.setEnabled(True)
    self.upCancel.setEnabled(False)
    self.upSearchLine.clear()
    self.oldNameValue.clear()
    self.oldDesgValue.clear()
    self.oldRoleValue.clear()
    self.oldContValue.clear()
    self.upNameLine.clear()
    self.upDesgLine.clear()
    self.upRoleList.clear()

```

```

return

def deleteUser(self):
    self.uiMain.setCurrentWidget(self.deUser)
    self.deUser.setFixedSize(700,300)

    search=QLabel(self.deUser)
    search.setText('Enter UserName')

    self.deSearchLine=QLineEdit(self.deUser)

    self.deSearchButton=QPushButton(self.deUser)
    self.deSearchButton.setText('Search')
    self.deSearchButton.adjustSize()
    self.deSearchButton.clicked.connect(self.deSearchUser)

    oldName=QLabel(self.deUser)
    oldName.setText('Full Name')
    oldName.setStyleSheet('font-weight:bold')

    oldCont=QLabel(self.deUser)
    oldCont.setText('Contact')
    oldCont.setStyleSheet('font-weight:bold')

    oldDesg=QLabel(self.deUser)
    oldDesg.setText('Designation')
    oldDesg.setStyleSheet('font-weight:bold')

    oldRole=QLabel(self.deUser)
    oldRole.setText('Role')
    oldRole.setStyleSheet('font-weight:bold')

    self.deDelete=QPushButton(self.deUser)
    self.deDelete.setText('Delete')
    self.deDelete.clicked.connect(self.delUser)

    self.deCancel=QPushButton(self.deUser)
    self.deCancel.setText('Cancel')
    self.deCancel.adjustSize()
    self.deCancel.setEnabled(False)
    self.deCancel.clicked.connect(self.deCancelUser)

    self.deUserGrid=QGridLayout()
    self.deUser.setLayout(self.deUserGrid)

    self.deUserGrid.addWidget(search,0,0)
    self.deUserGrid.addWidget(self.deSearchLine,0,1)
    self.deUserGrid.addWidget(self.deSearchButton,0,2)
    self.deUserGrid.addWidget(oldName,1,0)
    self.deUserGrid.addWidget(oldDesg,2,0)
    self.deUserGrid.addWidget(oldRole,3,0)
    self.deUserGrid.addWidget(oldCont,4,0)
    self.deUserGrid.addWidget(self.deDelete,5,2)
    self.deUserGrid.addWidget(self.deCancel,5,0)

```

```

def deSearchUser(self):
    user=self.deSearchLine.text()
    self.cur.execute('select name,designation,role,contact from projectuser where
username=(%s)',(user,))
    rec=self.cur.fetchall()
    print(len(rec))
    noUser=QLabel(self.deUser)
    if len(rec)==0:
        noUser.setText('*UserName does not exist!')
        noUser.setStyleSheet('color:red')
        self.deUserGrid.addWidget(noUser,8,0,1,4)
    else:
        self.deCancel.setEnabled(True)
        self.deSearchButton.setEnabled(False)
        noUser.clear()
        l=[]
        for i in rec:
            l.append(i[0])
            l.append(i[1])
            l.append(i[2])
            l.append(i[3])

        self.deNameValue=QLabel(self.deUser)
        self.deDesgValue=QLabel(self.deUser)
        self.deRoleValue=QLabel(self.deUser)
        self.deContValue=QLabel(self.deUser)

        if self.deNameValue.text()!=None:
            self.deNameValue.clear()
            self.deDesgValue.clear()
            self.deRoleValue.clear()
            self.deContValue.clear()

        self.deNameValue.setText(l[0])
        self.deDesgValue.setText(l[1])
        self.deRoleValue.setText(l[2])
        self.deContValue.setText(str(l[3]))

        self.deUserGrid.addWidget(self.deNameValue,1,1)
        self.deUserGrid.addWidget(self.deDesgValue,2,1)
        self.deUserGrid.addWidget(self.deRoleValue,3,1)
        self.deUserGrid.addWidget(self.deContValue,4,1)

def delUser(self):
    try:
        user=self.deSearchLine.text()
        self.cur.execute('delete projectuser where username=%s',(user,))
    except:
        print('error occured')
    else:
        self.con.commit()
        self.deSearchButton.setEnabled(True)
        self.deCancel.setEnabled(False)
        self.deSearchLine.clear()
        self.deNameValue.clear()
        self.deDesgValue.clear()
        self.deRoleValue.clear()

```



```

        self.deContValue.clear()
    return

```

```

def deCancelUser(self):
    self.deCancel.setEnabled(False)
    self.deSearchButton.setEnabled(True)
    self.deSearchLine.clear()
    self.deNameValue.clear()
    self.deDesgValue.clear()
    self.deRoleValue.clear()
    self.deContValue.clear()
    return

```

```

def stuMenu(self):
    if self.stuTable.currentIndex() == 1:
        self.showStudent()
        self.stuTable.setCurrentIndex(0)
    return

```

```

def showStudent(self):
    self.uiMain.setCurrentWidget(self.shStudent)
    self.shStudent.setFixedSize(700,480)

    self.shStudentMain=QWidget()
    self.shStudent.setFixedSize(700,380)

    self.stuCourseList=QComboBox(self.shStudent)
    self.stuSemList=QComboBox(self.shStudent)

    courses=['Course','CSE','BPharmacy']
    self.stuCourseList.addItem(courses)
    self.stuCourseList.activated.connect(self.stuSelectCrs)

```

Sem']

```

sem=['Semester','I Sem','II Sem','III Sem','IV Sem','V Sem','VI Sem','VII Sem','VIII

self.stuSemList.addItem(sem)
self.stuSemList.setEnabled(False)
self.stuSemList.activated.connect(self.stuSelectSem)

self.showStuButton=QPushButton(self.shStudent)
self.showStuButton.setText('SHOW')
self.showStuButton.setEnabled(False)
self.showStuButton.clicked.connect(self.searchStudents)

self.shStudentGrid=QGridLayout()
self.shStudent.setLayout(self.shStudentGrid)

self.shStudentMainGrid=QGridLayout()
self.shStudentMain.setLayout(self.shStudentMainGrid)

self.shStudentGrid.addWidget(self.stuCourseList,0,0)
self.shStudentGrid.addWidget(self.stuSemList,0,1)
self.shStudentGrid.addWidget(self.showStuButton,0,3)
self.shStudentGrid.addWidget(self.shStudentMain,1,0,1,4)

```

```

        return

def stuSelectCrs(self):
    if self.stuCourseList.currentIndex()!=0:
        self.stuSemList.setEnabled(True)
    else:
        self.stuSemList.setEnabled(False)
        self.showStuButton.setEnabled(False)
    return

def stuSelectSem(self):
    if self.stuSemList.currentIndex()!=0:
        self.showStuButton.setEnabled(True)
    else:
        self.showStuButton.setEnabled(False)
    return

def searchStudents(self):
    studentList=QListWidget(self.shStudentMain)
    studentList.setGeometry(50,600,700,380)

    self.shStudentMainGrid.addWidget(studentList,0,0)

    scroll=QScrollBar(self.shStudentMain)
    studentList.setVerticalScrollBar(scroll)
    return

def subMenu(self):
    if self.subTable.currentIndex()==1:
        self.showSubjects()
        self.subTable.setCurrentIndex(0)
    elif self.subTable.currentIndex()==2:
        self.updateSubjects()
        self.subTable.setCurrentIndex(0)
    return

def showSubjects(self):
    self.uiMain.setCurrentWidget(self.shSub)
    self.shSub.setFixedSize(700,450)

    self.shSubMain=QWidget(self.shSub)
    self.shSubMain.setFixedSize(700,420)

    self.courseList=QComboBox(self.shSub)
    self.semList=QComboBox(self.shSub)

    courses=['Course','CSE','BPharmacy']
    self.courseList.addItem(courses)
    self.courseList.activated.connect(self.subSelectCrs)

    sem=['Semester','I Sem','II Sem','III Sem','IV Sem','V Sem','VI Sem','VII Sem','VIII
Sem']

```

```

self.semList.addItem(sem)
self.semList.setEnabled(False)
self.semList.activated.connect(self.subSelectSem)

self.showSubButton=QPushButton(self.shSub)
self.showSubButton.setText('SHOW')
self.showSubButton.setEnabled(False)
self.showSubButton.clicked.connect(self.searchSubjects)

self.shSubGrid=QGridLayout()
self.shSub.setLayout(self.shSubGrid)

self.shMainGrid=QGridLayout()
self.shSubMain.setLayout(self.shMainGrid)

self.shSubGrid.addWidget(self.courseList,0,0,1,2)
self.shSubGrid.addWidget(self.semList,0,2,1,2)
self.shSubGrid.addWidget(self.showSubButton,0,4)
self.shSubGrid.addWidget(self.shSubMain,1,0,1,5)

return

def subSelectCrs(self):
    if self.courseList.currentIndex() != 0:
        self.semList.setEnabled(True)
    else:
        self.semList.setEnabled(False)
        self.showSubButton.setEnabled(False)
    return

def subSelectSem(self):
    if self.courseList.currentIndex() != 0 and self.semList.currentIndex() != 0:
        self.showSubButton.setEnabled(True)
    else:
        self.showSubButton.setEnabled(False)
    return

def searchSubjects(self):
    sname=QLabel(self.shSubMain)
    sname.setText('Subject Name')
    sname.setStyleSheet('font-weight:bold')

    scode=QLabel(self.shSubMain)
    scode.setText('Subject Code')
    scode.setStyleSheet('font-weight:bold')

    for i in reversed(range(self.shMainGrid.count())):
        self.shMainGrid.itemAt(i).widget().setParent(None)

    self.shMainGrid.addWidget(sname,0,0)
    self.shMainGrid.addWidget(scode,0,1)
    if self.courseList.currentText() == 'CSE':

```

```

        curSub=self.con.cursor()
        curSub.execute('select sub,subcode from
cse'+str(self.semList.currentIndex()))
        rec=curSub.fetchall()
        row=1
        for i in rec:
            col=0
            for j in i:
                self.shMainGrid.addWidget(QLabel(str(j)),row,col)
                col=col+1
            row=row+1
        elif self.courseList.currentText()=='Pharmacy':
            curSub=self.con.cursor()
            curSub.execute('select sub,subcode from
phm'+str(self.semList.currentIndex()))
            rec=curSub.fetchall()
            row=1
            for i in rec:
                col=0
                for j in i:
                    self.shMainGrid.addWidget(QLabel(str(j)),row,col)
                    col=col+1
                row=row+1
    return

```

```

def dateMenu(self):
    if self.datesheet.currentIndex()==1:
        self.genDate()
        self.datesheet.setCurrentIndex(0)
    if self.datesheet.currentIndex()==2:
        self.showDate()
        self.datesheet.setCurrentIndex(0)
    if self.datesheet.currentIndex()==3:
        pass
    return

```

```

def genDate(self):
    self.uiMain.setCurrentWidget(self.gDate)
    self.gDate.setFixedSize(980,450)

    self.gDateMain=QWidget(self.gDate)
    self.gDateMain.setFixedSize(980,300)
    #scroll=QScrollArea(self.gDate)

    start=QLabel()
    start.setText('Start Date')
    start.setStyleSheet('font-weight:bold')
    start.setAlignment(Qt.AlignCenter)

    crsLabel=QLabel()
    crsLabel.setText('Course:')
    crsLabel.setStyleSheet('font-weight:bold')
    crsLabel.setAlignment(Qt.AlignCenter)

    semLabel=QLabel()

```

```

semLabel.setText('Semester:')
semLabel.setStyleSheet('font-weight:bold')
semLabel.setAlignment(Qt.AlignCenter)

self.genDateCrsList=QComboBox()
courses=['Courses','CSE','BPharmacy']
self.genDateCrsList.addItem(courses)
self.genDateCrsList.setCurrentIndex(0)
self.genDateCrsList.activated.connect(self.genSelectCourse)

self.genDateSemList=QComboBox()
sem=['Semester','Even Semester','Odd Semester']
self.genDateSemList.addItem(sem)
self.genDateSemList.setEnabled(False)
self.genDateSemList.setCurrentIndex(0)
self.genDateSemList.activated.connect(self.genSelectSem)

self.genDateStart=QDateEdit()
self.genDateStart.setMinimumDate(QDate(date.today()))
self.genDateStart.setMaximumDate(QDate(date(date.today()).year,12,31)))

self.genDateSubmit=QPushButton()
self.genDateSubmit.setText('Submit')
self.genDateSubmit.setEnabled(False)
self.genDateSubmit.clicked.connect(self.generateDatesheet)

self.genDateGrid=QGridLayout()
self.gDate.setLayout(self.genDateGrid)

self.genDateMainGrid=QGridLayout()
self.gDateMain.setLayout(self.genDateMainGrid)

for i in reversed(range(self.genDateGrid.count())):
    self.genDateGrid.itemAt(i).widget().setParent(None)

self.genDateGrid.addWidget(start,0,0)
self.genDateGrid.addWidget(self.genDateStart,0,1,1,2)
self.genDateGrid.addWidget(crsLabel,0,3)
self.genDateGrid.addWidget(self.genDateCrsList,0,4,1,2)
self.genDateGrid.addWidget(semLabel,0,6)
self.genDateGrid.addWidget(self.genDateSemList,0,7,1,2)
self.genDateGrid.addWidget(self.genDateSubmit,0,10)
self.genDateGrid.addWidget(self.gDateMain,2,0,1,11)

'''scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)
scroll.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
#scroll.setWidgetResizable(True)
scroll.setWidget(self.gDate)'''
return

def genSelectCourse(self):
    if self.genDateCrsList.currentIndex()!=0:
        self.genDateSemList.setEnabled(True)
    else:
        self.genDateSemList.setEnabled(False)
        self.genDateSubmit.setEnabled(False)

```

```

return

def genSelectSem(self):
    if self.genDateSemList.currentIndex() != 0 and self.genDateCrsList.currentIndex() != 0:
        self.genDateSubmit.setEnabled(True)
    else:
        self.genDateSubmit.setEnabled(False)
    return

def generateDatesheet(self):
    fetCourse=self.con.cursor()
    createSheet=self.con.cursor()
    insertValues=self.con.cursor()
    msg=QLabel()
    msg.setText('Datesheet has already been released for the selected Course/Semester!')
    try:
        if self.genDateCrsList.currentText()=='CSE':
            if self.genDateSemList.currentText()=='Even Semester':
                sem=['cse2','cse4','cse6','cse8']
            elif self.genDateSemList.currentText()=='Odd Semester':
                sem=['cse1','cse3','cse5','cse7']
            dateY=self.genDateStart.date().toPyDate().year
            for i in sem:
                semQuery='create table '+i+str(dateY)+'_dsheet (edate date
not null, subcode varchar2(10), sub varchar2(20))'
                createSheet.execute(semQuery)
            elif self.genDateCrsList.currentText()=='BPharmacy':
                if self.genDateSemList.currentText()=='Even Semester':
                    sem=['Ph2','Ph4','Ph6','Ph8']
                elif self.genDateSemList.currentText()=='Odd Semester':
                    sem=['Ph1','Ph3','Ph5','Ph7']
                dateY=self.genDateStart.date().toPyDate().year
                for i in sem:
                    semQuery='create table '+i+str(dateY)+'_dsheet (edate date
not null, subcode varchar2(10), sub varchar2(20))'
                    createSheet.execute(semQuery)
        except:
            for i in reversed(range(self.genDateMainGrid.count())):
                self.genDateMainGrid.itemAt(i).widget().setParent(None)
            self.genDateMainGrid.addWidget(msg,0,0)
            self.genDateSubmit.setEnabled(False)
        else:
            for i in reversed(range(self.genDateMainGrid.count())):
                self.genDateMainGrid.itemAt(i).widget().setParent(None)
            newDate=self.con.cursor()
            if self.genDateCrsList.currentText()=='CSE':
                if self.genDateSemList.currentText()=='Even Semester':
                    sem=['cse2','cse4','cse6','cse8']
                elif self.genDateSemList.currentText()=='Odd Semester':
                    sem=['cse1','cse3','cse5','cse7']
                dateY=self.genDateStart.date().toPyDate().year
                for i in sem:
                    query='select *from '+i
                    fetCourse.execute(query)
                    rec=fetCourse.fetchall()

```

```

        shuffle(rec)
        d=self.genDateStart.date().toPyDate()
        for j in rec:
            insert='insert into '+i+str(dateY)+'_dsheet

values(:edate,:code,:sub)'

            insertValues.execute(insert,(d,j[0],j[1]))
            if d.weekday()==5:
                d+=timedelta(days=2)
            else:
                d+=timedelta(days=1)
        self.con.commit()
        self.genDateSubmit.setEnabled(False)

        if self.genDateSemList.currentText()=='Even Semester':
            generate='select substr(cse2'+str(dateY)+'_dsheet.edate,1,9),
cse2'+str(dateY)+'_dsheet.subcode, cse2'+str(dateY)+'_dsheet.sub,
cse4'+str(dateY)+'_dsheet.subcode, cse4'+str(dateY)+'_dsheet.sub,cse6'+str(dateY)+'_dsheet.subcode,
cse6'+str(dateY)+'_dsheet.sub, cse8'+str(dateY)+'_dsheet.subcode, cse8'+str(dateY)+'_dsheet.sub
from cse2'+str(dateY)+'_dsheet left join cse4'+str(dateY)+'_dsheet on
cse2'+str(dateY)+'_dsheet.edate=cse4'+str(dateY)+'_dsheet.edate left join cse6'+str(dateY)+'_dsheet
on cse6'+str(dateY)+'_dsheet.edate=cse2'+str(dateY)+'_dsheet.edate right join
cse8'+str(dateY)+'_dsheet on cse8'+str(dateY)+'_dsheet.edate=cse2'+str(dateY)+'_dsheet.edate'
            elif self.genDateSemList.currentText()=='Odd Semester':
                generate='select cse1'+str(dateY)+'_dsheet.edate,
cse1'+str(dateY)+'_dsheet.subcode, cse1'+str(dateY)+'_dsheet.sub,
cse3'+str(dateY)+'_dsheet.subcode, cse3'+str(dateY)+'_dsheet.sub,cse5'+str(dateY)+'_dsheet.subcode,
cse5'+str(dateY)+'_dsheet.sub, cse7'+str(dateY)+'_dsheet.subcode, cse7'+str(dateY)+'_dsheet.sub
from cse1'+str(dateY)+'_dsheet left join cse3'+str(dateY)+'_dsheet on
cse1'+str(dateY)+'_dsheet.edate=cse3'+str(dateY)+'_dsheet.edate left join cse5'+str(dateY)+'_dsheet
on cse5'+str(dateY)+'_dsheet.edate=cse1'+str(dateY)+'_dsheet.edate right join
cse7'+str(dateY)+'_dsheet on cse7'+str(dateY)+'_dsheet.edate=cse1'+str(dateY)+'_dsheet.edate'
            elif self.genDateCrsList.currentText()=='BPharmacy':
                if self.genDateSemList.currentText()=='Even Semester':
                    sem=['Ph2','Ph4','Ph6','Ph8']
                elif self.genDateSemList.currentText()=='Odd Semester':
                    sem=['Ph1','Ph3','Ph5','Ph7']
                dateY=self.genDateStart.date().toPyDate().year
                for i in sem:
                    query='select *from '+i
                    fetCourse.execute(query)
                    rec=fetCourse.fetchall()
                    shuffle(rec)
                    d=self.genDateStart.date().toPyDate()
                    for j in rec:
                        insert='insert into '+i+str(dateY)+'_dsheet

values(:edate,:code,:sub)'

                        insertValues.execute(insert,(d,j[0],j[1]))
                        d+=timedelta(days=1)
                    self.con.commit()
                    self.genDateSubmit.setEnabled(False)
                newDate.execute(generate)
                dateRec=newDate.fetchall()
                row=0
                col=0
                for i in dateRec:
                    for j in i:
                        #if j!='None'

```

```

        self.genDateMainGrid.addWidget(QLabel(str(j)),row,col)
        col+=1
    row+=1
    col=0
return

def showDate(self):
    self.uiMain.setCurrentWidget(self.sDate)
    self.sDate.setFixedSize(980,450)

    self.sDateMain=QWidget(self.sDate)
    self.sDateMain.setFixedSize(980,300)

    crsLabel=QLabel()
    crsLabel.setText('Course:')
    crsLabel.setStyleSheet('font-weight:bold')
    crsLabel.setAlignment(Qt.AlignCenter)

    semLabel=QLabel()
    semLabel.setText('Semester:')
    semLabel.setStyleSheet('font-weight:bold')
    semLabel.setAlignment(Qt.AlignCenter)

    self.shDateCrsList=QComboBox()
    courses=['Courses','CSE','BPharmacy']
    self.shDateCrsList.addItem(courses)
    self.shDateCrsList.setCurrentIndex(0)
    self.shDateCrsList.activated.connect(self.shSelectCourse)

    self.shDateSemList=QComboBox()
    sem=['Semester','Even Semester','Odd Semester']
    self.shDateSemList.addItem(sem)
    self.shDateSemList.setEnabled(False)
    self.shDateSemList.setCurrentIndex(0)
    self.shDateSemList.activated.connect(self.shSelectSem)

    self.shDateSubmit=QPushButton()
    self.shDateSubmit.setText('Submit')
    self.shDateSubmit.setEnabled(False)
    self.shDateSubmit.clicked.connect(self.showDatesheet)

    self.shDateGrid=QGridLayout()
    self.sDate.setLayout(self.shDateGrid)

    self.shDateMainGrid=QGridLayout()
    self.sDateMain.setLayout(self.shDateMainGrid)

    for i in reversed(range(self.shDateGrid.count())):
        self.shDateGrid.itemAt(i).widget().setParent(None)

    self.shDateGrid.addWidget(crsLabel,0,0)
    self.shDateGrid.addWidget(self.shDateCrsList,0,1,1,2)
    self.shDateGrid.addWidget(semLabel,0,3)
    self.shDateGrid.addWidget(self.shDateSemList,0,4,1,2)
    self.shDateGrid.addWidget(self.shDateSubmit,0,8)
    self.shDateGrid.addWidget(self.sDateMain,2,0,1,9)

```



```

        return

def shSelectCourse(self):
    if self.shDateCrsList.currentIndex()!=0:
        self.shDateSemList.setEnabled(True)
    else:
        self.shDateSemList.setEnabled(False)
        self.shDateSubmit.setEnabled(False)
    return

def shSelectSem(self):
    if self.shDateSemList.currentIndex()!=0 and self.shDateCrsList.currentIndex()!=0:
        self.shDateSubmit.setEnabled(True)
    else:
        self.shDateSubmit.setEnabled(False)
    return

def showDatesheet(self):
    dateY=datetime.now().date().year
    try:
        datesheet=self.con.cursor()
        if self.shDateCrsList.currentText()=='CSE':
            if self.shDateSemList.currentText()=='Even Semester':
                show='select substr(cse2'+str(dateY)+'_dsheet.edate,1,9),
cse2'+str(dateY)+'_dsheet.subcode, cse2'+str(dateY)+'_dsheet.sub,
cse4'+str(dateY)+'_dsheet.subcode, cse4'+str(dateY)+'_dsheet.sub,cse6'+str(dateY)+'_dsheet.subcode,
cse6'+str(dateY)+'_dsheet.sub, cse8'+str(dateY)+'_dsheet.subcode, cse8'+str(dateY)+'_dsheet.sub
from cse2'+str(dateY)+'_dsheet left join cse4'+str(dateY)+'_dsheet on
cse2'+str(dateY)+'_dsheet.edate=cse4'+str(dateY)+'_dsheet.edate left join cse6'+str(dateY)+'_dsheet
on cse6'+str(dateY)+'_dsheet.edate=cse2'+str(dateY)+'_dsheet.edate right join
cse8'+str(dateY)+'_dsheet on cse8'+str(dateY)+'_dsheet.edate=cse2'+str(dateY)+'_dsheet.edate'
            elif self.shDateSemList.currentText()=='Odd Semester':
                show='select cse1'+str(dateY)+'_dsheet.edate,
cse1'+str(dateY)+'_dsheet.subcode, cse1'+str(dateY)+'_dsheet.sub,
cse3'+str(dateY)+'_dsheet.subcode, cse3'+str(dateY)+'_dsheet.sub,cse5'+str(dateY)+'_dsheet.subcode,
cse5'+str(dateY)+'_dsheet.sub, cse7'+str(dateY)+'_dsheet.subcode, cse7'+str(dateY)+'_dsheet.sub
from cse1'+str(dateY)+'_dsheet left join cse3'+str(dateY)+'_dsheet on
cse1'+str(dateY)+'_dsheet.edate=cse3'+str(dateY)+'_dsheet.edate left join cse5'+str(dateY)+'_dsheet
on cse5'+str(dateY)+'_dsheet.edate=cse1'+str(dateY)+'_dsheet.edate right join
cse7'+str(dateY)+'_dsheet on cse7'+str(dateY)+'_dsheet.edate=cse1'+str(dateY)+'_dsheet.edate'
            elif self.shDateCrsList.currentText()=='BPharmacy':
                if self.shDateSemList.currentText()=='Even Semester':
                    pass
                elif self.shDateSemList.currentText()=='Odd Semester':
                    pass
            datesheet.execute(show)
    except:
        msg=QLabel()
        msg.setText('Datesheet has not been generated yet.')
        for i in reversed(range(self.shDateMainGrid.count())):
            self.shDateMainGrid.itemAt(i).widget().setParent(None)
        self.shDateMainGrid.addWidget(msg,0,0)

```

```

        self.shDateSubmit.setEnabled(False)
        pass
    else:
        for i in reversed(range(self.shDateMainGrid.count())):
            self.shDateMainGrid.itemAt(i).widget().setParent(None)
        dateRec=datesheet.fetchall()
        row=0
        col=0
        for i in dateRec:
            for j in i:
                #if j!='None'
                self.shDateMainGrid.addWidget(QLabel(str(j)),row,col)
                col+=1
            row+=1
            col=0
        pass
    return

def resultMenu(self):
    if self.resultTable.currentIndex()==1:
        self.resultTable.setCurrentIndex(0)
        self.uploadResult()
    elif self.resultTable.currentIndex()==2:
        self.resultTable.setCurrentIndex(0)
        self.showResult()
    elif self.resultTable.currentIndex()==3:
        self.resultTable.setCurrentIndex(0)
        self.updateResult()
    return

def uploadResult(self):
    self.uiMain.setCurrentWidget(self.uplResult)
    self.uplResult.setFixedSize(980,450)

    head=QWidget(self.uplResult)
    head.setFixedSize(970,70)

    listWidget=QWidget(self.uplResult)
    listWidget.setFixedSize(250,350)

    detailsWidget=QWidget(self.uplResult)
    detailsWidget.setFixedSize(700,80)

    marksWidget=QWidget(self.uplResult)
    marksWidget.setFixedSize(700,250)

    uploadMarksGrid=QGridLayout()
    self.uplResult.setLayout(uploadMarksGrid)

    uploadMarksGrid.addWidget(head,0,0,1,3)
    uploadMarksGrid.addWidget(listWidget,1,0,2,2)
    uploadMarksGrid.addWidget(detailsWidget,1,2)
    uploadMarksGrid.addWidget(marksWidget,2,2)

```

```

self.uplResultCrsList=QComboBox()
self.uplResultSemList=QComboBox()

courses=['Course','CSE','BPharmacy']
self.uplResultCrsList.addItem(courses)
self.uplResultCrsList.activated.connect(self.uplResultSelectCrs)

Sem']
sem=['Semester','I Sem','II Sem','III Sem','IV Sem','V Sem','VI Sem','VII Sem','VIII

self.uplResultSemList.addItem(sem)
self.uplResultSemList.setEnabled(False)
self.uplResultSemList.activated.connect(self.uplResultSelectSem)

self.uplResultStuButton=QPushButton(self.head)
self.uplResultStuButton.setText('SHOW')
self.uplResultStuButton.setEnabled(False)
self.uplResultStuButton.clicked.connect(self.fetchStudents)

headGrid=QGridLayout()
head.setLayout(headGrid)

headGrid.addWidget(self.uplResultCrsList,0,0,1,2)
headGrid.addWidget(self.uplResultSemList,0,2,1,2)
headGrid.addWidget(self.uplResultStuButton,0,6)

self.studentList=QListWidget(listWidget)
self.studentList.setGeometry(50,20,250,400)
self.studentList.setEnabled(False)
scrollbar=QScrollBar()
self.studentList.addScrollBarWidget(scrollbar,Qt.AlignRight)
hbox=QHBoxLayout()
listWidget.setLayout(hbox)
hbox.addWidget(self.studentList)

self.detailsGrid=QGridLayout()
detailsWidget.setLayout(self.detailsGrid)

self.marksGrid=QGridLayout()
marksWidget.setLayout(self.marksGrid)

return

def uplResultSelectCrs(self):
    if self.uplResultCrsList.currentIndex()!=0:
        self.uplResultSemList.setEnabled(True)
    else:
        self.uplResultSemList.setEnabled(False)
        self.uplResultStuButton.setEnabled(False)
    return

def uplResultSelectSem(self):
    if self.uplResultSemList.currentIndex()!=0:
        self.uplResultStuButton.setEnabled(True)
    else:
        self.uplResultStuButton.setEnabled(False)

```

```
return
```

```
def fetchStudents(self):
    self.studentList.setEnabled(True)
    self.uplResultStuButton.setEnabled(False)
    course=self.uplResultCrsList.currentText()
    sem=str(self.uplResultSemList.currentIndex())
    query='select roll from projectstudent where course=:course and semester=:sem'
    detailCur=self.con.cursor()
    detailCur.execute(query,(course,sem))
    fetDetail=detailCur.fetchall()
    for i in fetDetail:
        print(i[0])
        self.studentList.addItem(str(i[0]))
    self.studentList.activated.connect(self.studentDetails)
    return
```

```
def studentDetails(self):
    print(type(int(self.studentList.currentItem().text())))
    self.studentList.setEnabled(False)
```

```
query1='select roll,name,course,semester from projectstudent where
roll='+self.studentList.currentItem().text()
detailCur=self.con.cursor()
detailCur.execute(query1)
fetDetail=detailCur.fetchall()
```

```
course=str(fetDetail[0][2])
semester=str(fetDetail[0][3])
```

```
rollLabel=QLabel('Roll No.:')
rollLabel.setStyleSheet('font-weight:bold;')
nameLabel=QLabel('Name:')
nameLabel.setStyleSheet('font-weight:bold;')
crsLabel=QLabel('Course:')
crsLabel.setStyleSheet('font-weight:bold;')
semLabel=QLabel('Semester:')
semLabel.setStyleSheet('font-weight:bold;')
```

```
roll=QLabel()
roll.setText(str(fetDetail[0][0]))
roll.setStyleSheet('background-color: #232629;'
    'border-radius:8;')
```

```
name=QLabel()
name.setText(str(fetDetail[0][1]))
name.setStyleSheet('background-color: #232629;'
    'border-radius:8;')
```

```
crs=QLabel()
crs.setText(str(fetDetail[0][2]))
crs.setStyleSheet('background-color: #232629;'
    'border-radius:8;')
```

```
sem=QLabel()
```

```

sem.setText(str(fetDetail[0][3]))
sem.setStyleSheet('background-color: #232629;'
                  'border-radius:8;')

self.detailsGrid.addWidget(rollLabel,0,0)
self.detailsGrid.addWidget(roll,1,0)
self.detailsGrid.addWidget(nameLabel,0,2)
self.detailsGrid.addWidget(name,1,2)
self.detailsGrid.addWidget(crsLabel,0,4)
self.detailsGrid.addWidget(crs,1,4)
self.detailsGrid.addWidget(semLabel,0,6)
self.detailsGrid.addWidget(sem,1,6)

subCur=self.con.cursor()
query2='select sub from '+course+semester
subCur.execute(query2)
fetSub=subCur.fetchall()
print(fetSub)

row=0
for i in fetSub:
    self.marksGrid.addWidget(QLabel(str(i[0])),row,0)
    row+=1

return

def showResult(self):
    pass
    return

def updateResult(self):
    pass
    return

def browsePhoto(self):
    pass

def editProfile(self):
    self.submit.setEnabled(True)
    self.cancel.setEnabled(True)
    self.edit.setEnabled(False)
    self.nameLine.setEnabled(True)
    self.desgLine.setEnabled(True)
    self.conLine.setEnabled(True)

def subProfile(self):
    query='update projectuser set name=:name,designation=:designation,contact=:contact
where username=:username'
```

```

        self.profileCur.execute(query,(self.nameLine.text(),self.desgLine.text(),self.conLine.text(),self.user))

        self.con.commit()
        self.submit.setEnabled(False)
        self.cancel.setEnabled(False)
        self.edit.setEnabled(True)
        self.nameLine.setEnabled(False)
        self.desgLine.setEnabled(False)
        self.conLine.setEnabled(False)

def canProfile(self):
    self.submit.setEnabled(False)
    self.cancel.setEnabled(False)
    self.edit.setEnabled(True)
    self.nameLine.setEnabled(False)
    self.desgLine.setEnabled(False)
    self.conLine.setEnabled(False)

def userSett(self):

    if self.profileMenu.currentText()=='Profile':
        self.profileSettings()
        self.profileMenu.setCurrentIndex(0)

    elif self.profileMenu.currentText()=='LogOut':
        self.con.close()
        self.close()

    return

app=QApplication(sys.argv)
app.setStyle('Fusion')
apply_stylesheet(app, theme='dark_teal.xml')
ex=mainWindow()
ex.show()
sys.exit(app.exec_())

```

SCREENSHOTS

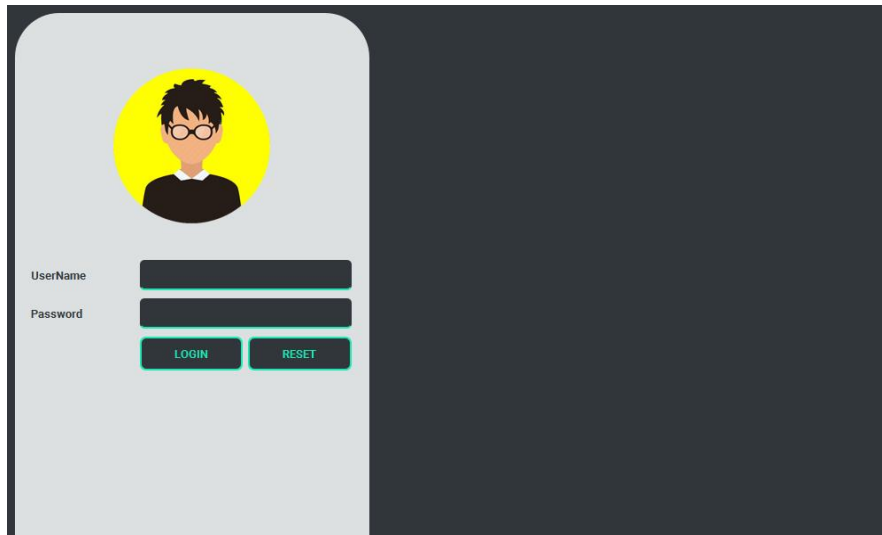


Fig. 1: LogIn Screen.

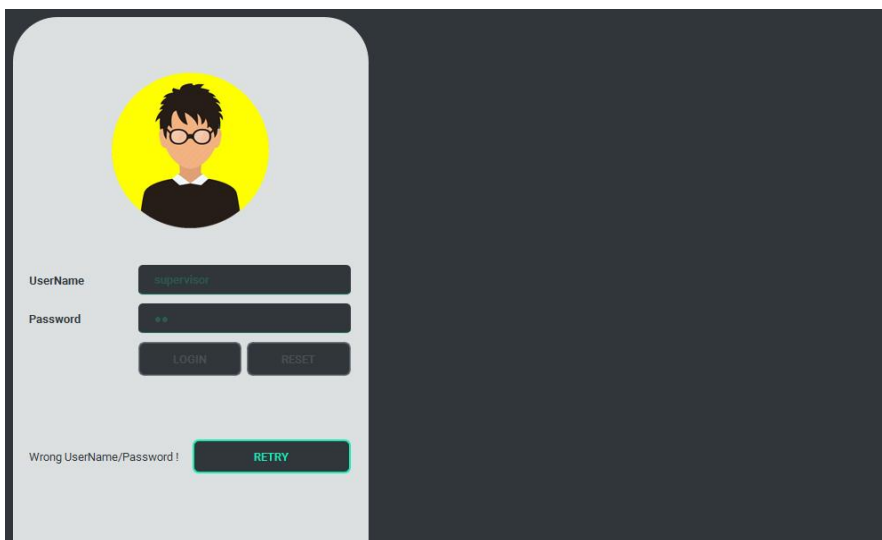


Fig. 2: Wrong password filled while logging in

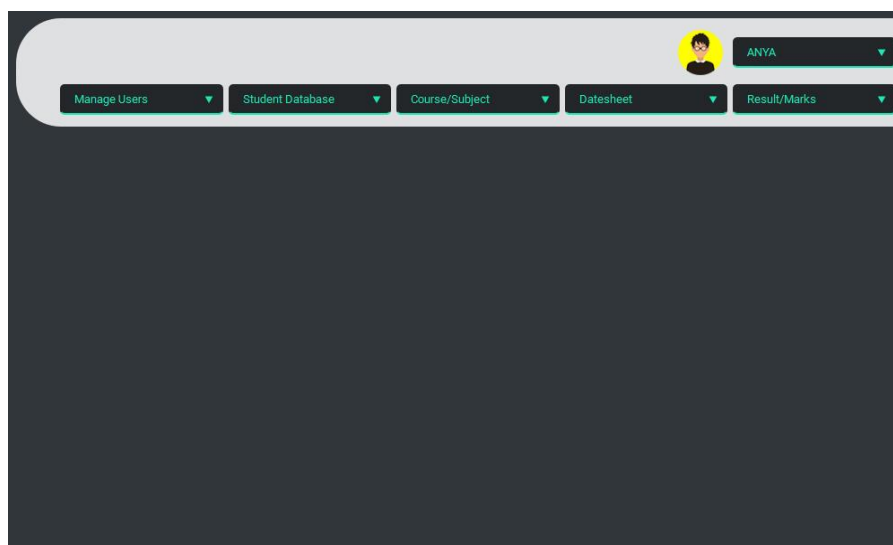


Fig. 3: Main UI Window after successful LogIn

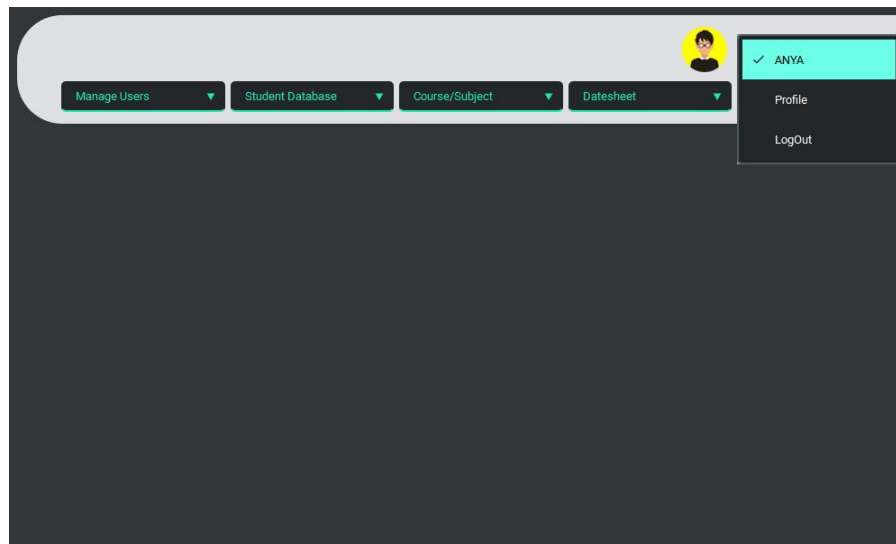


Fig. 4: Profile Menu

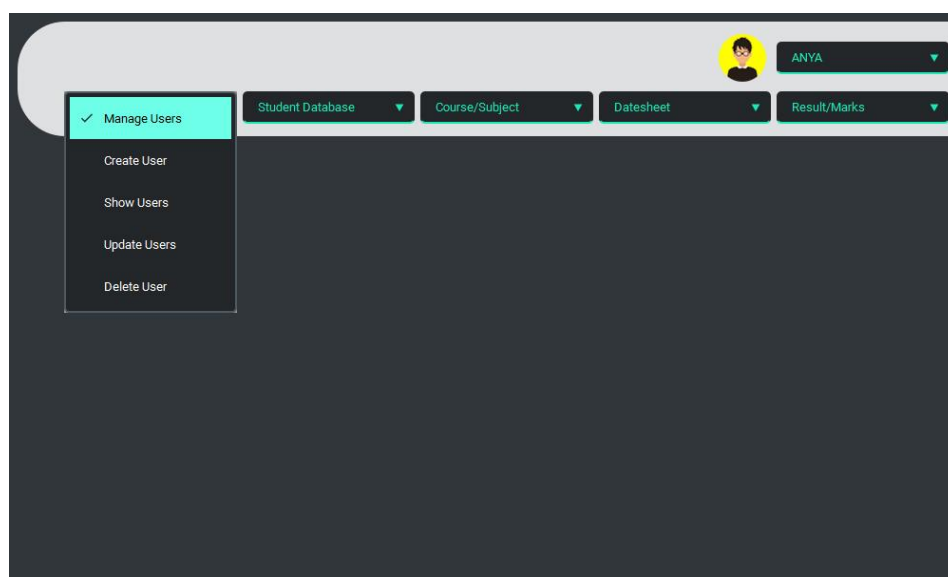


Fig. 5: User Menu

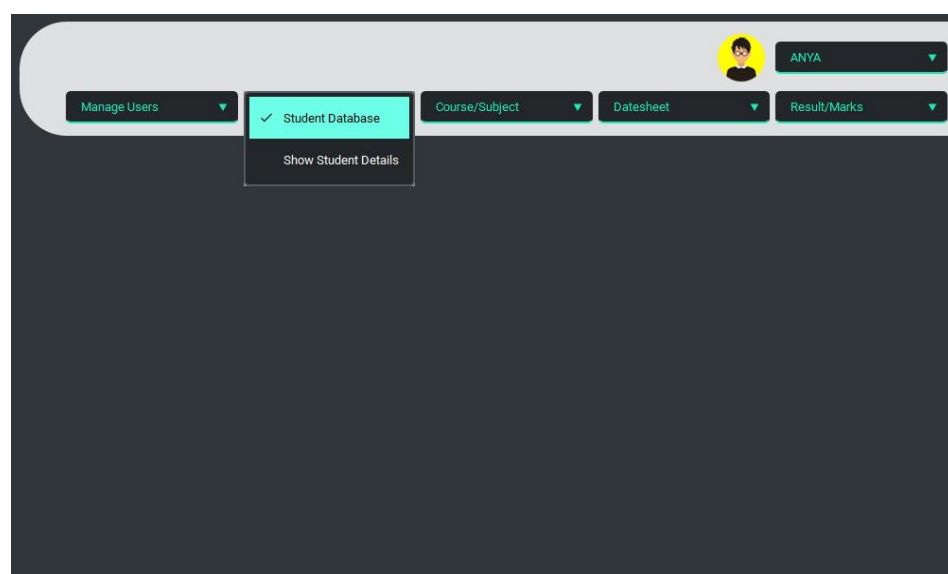


Fig. 6: Student Database Menu

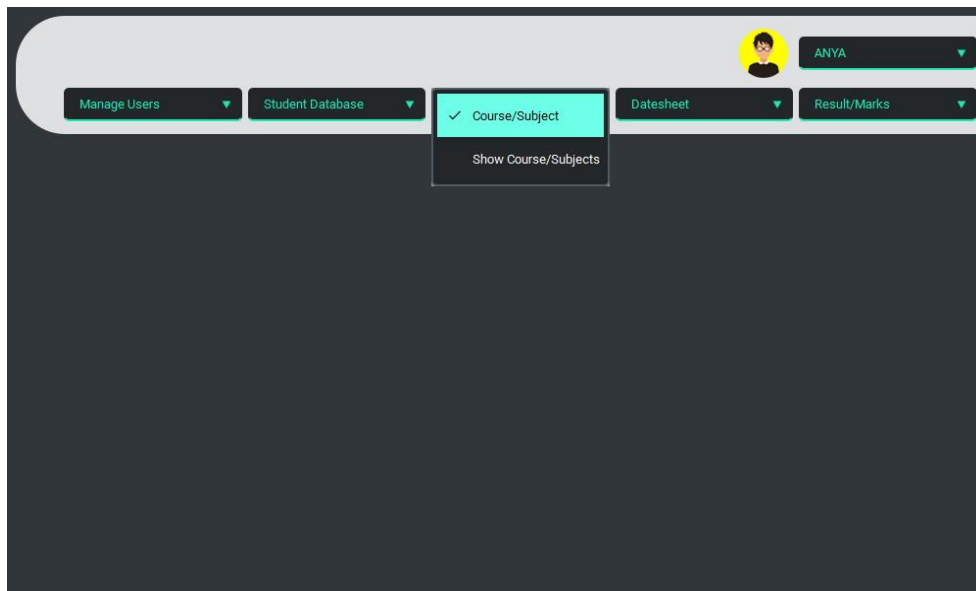


Fig. 7: Course/Subject Menu

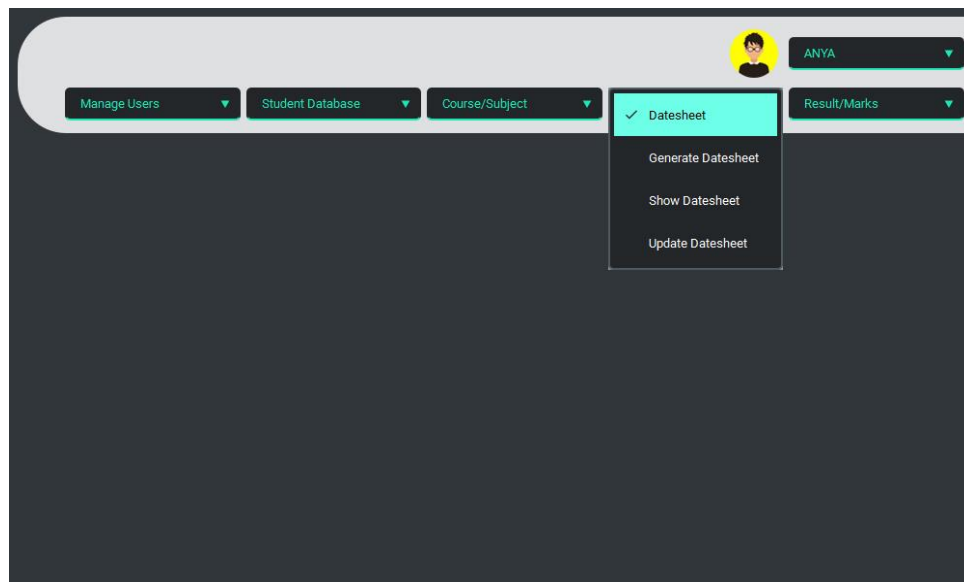


Fig. 8: Date-sheet Menu (Pre-Examination)

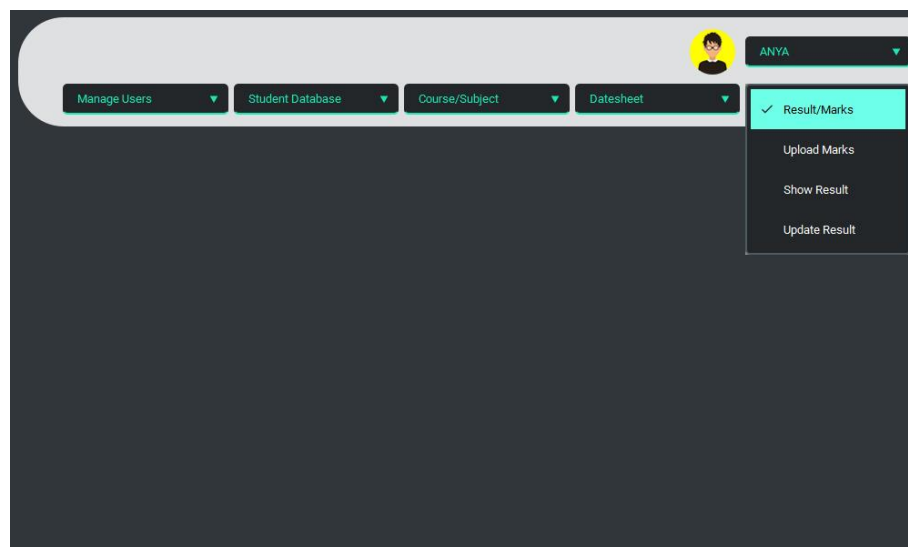
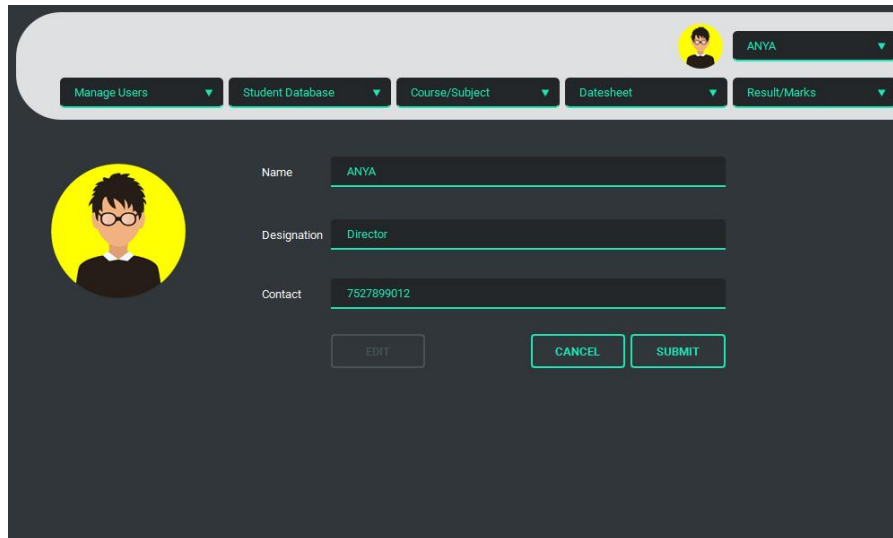



Fig. 9: Results/Marks Menu



Manage Users ▾ Student Database ▾ Course/Subject ▾ Datesheet ▾ Result/Marks ▾

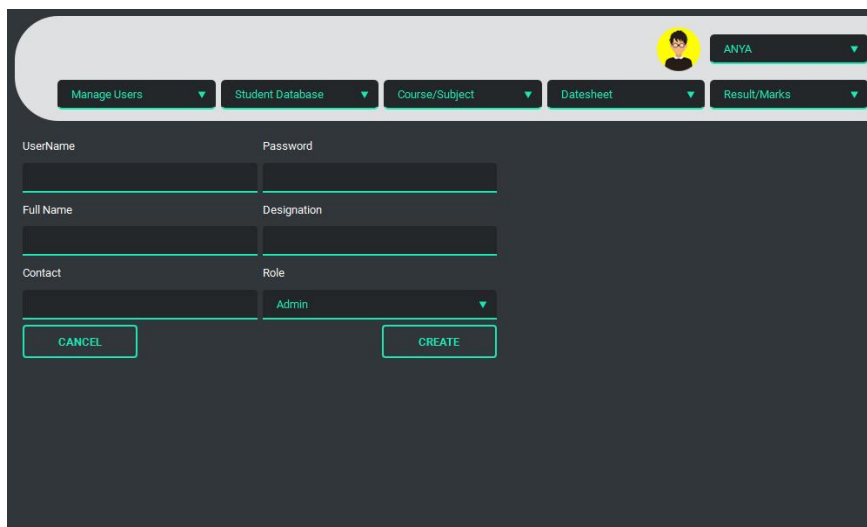
ANYA ▾

 Name:

Designation:

Contact:

Fig. 10: Show Profile



Manage Users ▾ Student Database ▾ Course/Subject ▾ Datesheet ▾ Result/Marks ▾

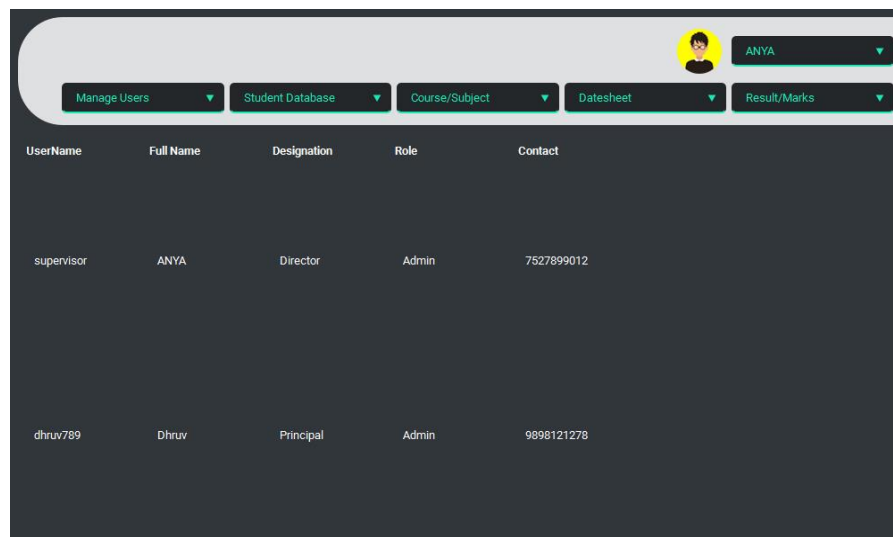
ANYA ▾

UserName: Password:

Full Name: Designation:

Contact: Role:

Fig. 11: Create User Window



UserName	Full Name	Designation	Role	Contact
supervisor	ANYA	Director	Admin	7527899012
dhruv789	Dhruv	Principal	Admin	9898121278

Fig. 12: Displaying currently present users

Enter UserName

Full Name

Designation

Role

Contact

Fig. 13: Update User Window

Enter UserName

Full Name

Designation

Role

Contact

Fig. 14: Retrieving User details from database

CSE

- ✓ Semester
- I Sem
- II Sem
- III Sem
- IV Sem
- V Sem
- VI Sem
- VII Sem
- VIII Sem

Fig. 15: Selection of Course and Semester to fetch student details

Subject Name	Subject Code
DISCRETE STRUCTURES	ACDS-16402
OPERATING SYSTEM	ACCS-16402
COMPUTER NETWORKS	ACCS-16403
PYTHON	ACCS-16404
RDBMS	ACCS-16405

Fig. 16: List of subject of specific class

Start Date: 5/16/2022 Course: CSE Semester: Even Semester SUBMIT

Datesheet has already been released for the selected Course/Semester!

Fig. 17: Datesheet Generation

Date	Course	Subject
05-MAY-22	ACPH-101	ENGG PHYSICS
06-MAY-22	ACEE-101	BEEE
07-MAY-22	ACAM-102	ENGG MATHS
09-MAY-22	ACHV-101	HUMAN VALUES
10-MAY-22	ACHU-102	COMM ENGLISH

Fig. 18: Generated Datesheet

Manage Users

Student Database

Course/Subject

Datesheet

Result/Marks

ANYA

CSE

II Sem

SHOW

2021001

2021002

Roll No.:

2021002

Name:

dhruv

Course:

CSE

Semester:

2

ENGG MATHS

COMM ENGLISH

BEEE

ENGG PHYSICS

HUMAN VALUES

Fig. 19:Marks Upload

REFERENCES

https://www.researchgate.net/publication/349265225_EXAMINATION_MANAGEMENT_SYSTEM

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3611554/A-Study-On-Web-Based-Online-Examination-System

<https://coderslegacy.com/python/getting-started-pyqt/>

<https://zetcode.com/gui/pyqt5/firstprograms/>

<https://ieeexplore.ieee.org/document/6011131>

<https://stackoverflow.com/>

FUTURE SCOPE

- Post-examination refers to generation of marks and results of each student, maybe based upon MCQs which can use OMR to generate result, or descriptive examination which needs to upload result manually by DEO.
- The Marks/Result Views may use Bars representation, Pie representation or even basic Table representation according to the user needs.
- The Marks/Result generator may use both OMR Scanner and Manual Data Entry, or only one of these, for generating each student's result.
- Marks/Result Viewer is another feature of this system where user will be provided with graphical views of any student, class, branch, course or semester results and may have feature to project out that views.