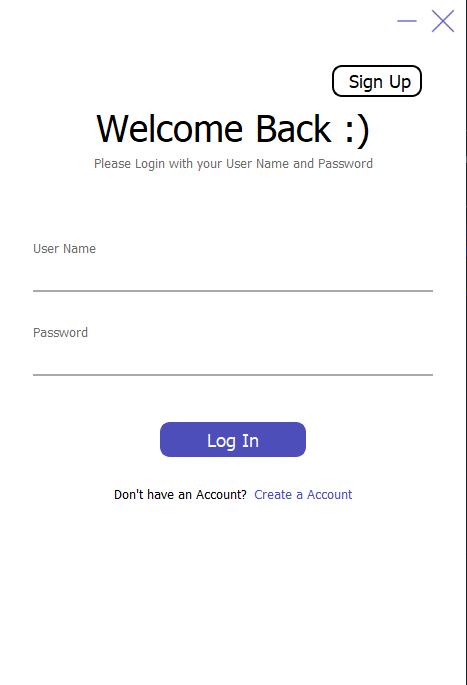
**Class LoginSignUpWindow:**

It is a sign in and sign-up page. It has stacked widget(stacked widget is used to open new tab in the same window. For Example, In this class when signup button is clicked it does not open new window instead it shows signup page on the same window). It has login button. It has signup button. It has link of create account.

by clicking Login button we can login user. The register button name is btnLogInTo

# Connect the btnLogInTo button to the LoggingIn() method when it is clicked

self.btnLogInTo.clicked.connect(self.LoggingIn) line-52

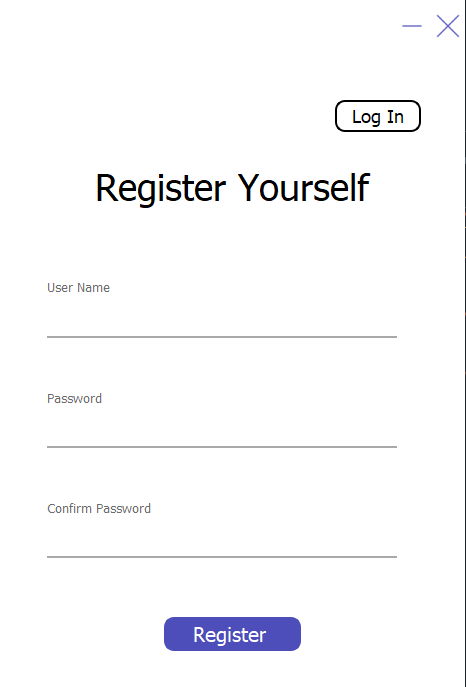
On clicking SignUp button or Create a Account Link, we go to the signup window by this code

# Connect the btnSignUp button to a lambda function that sets the current widget to the sign-up page

self.btnSignUp.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.SignUpPage)) Line-38

On Link clicking:

self.lblCreateAccount.mousePressEvent = self.on\_lblCreateAccount\_clicked Line-43

Similarly, here we can register ourselves and we can move it login page again by clicking on login page by following code.

# Connect the btnLogIn\_3 button to a lambda function that sets the current widget to the sign-in page

self.btnLogIn\_3.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.SignInPage)) Line-40

and by clicking register button we can register user. The register button name is btnRegister

# Connect the btnRegister button to the addUserToDB() method when it is clicked

self.btnRegister.clicked.connect(self.addUserToDB) Line-50

**Class TeacherWindow:**

It has teacher window all teachers modules will handle. One important thing is that we are passing one parameter to this class.

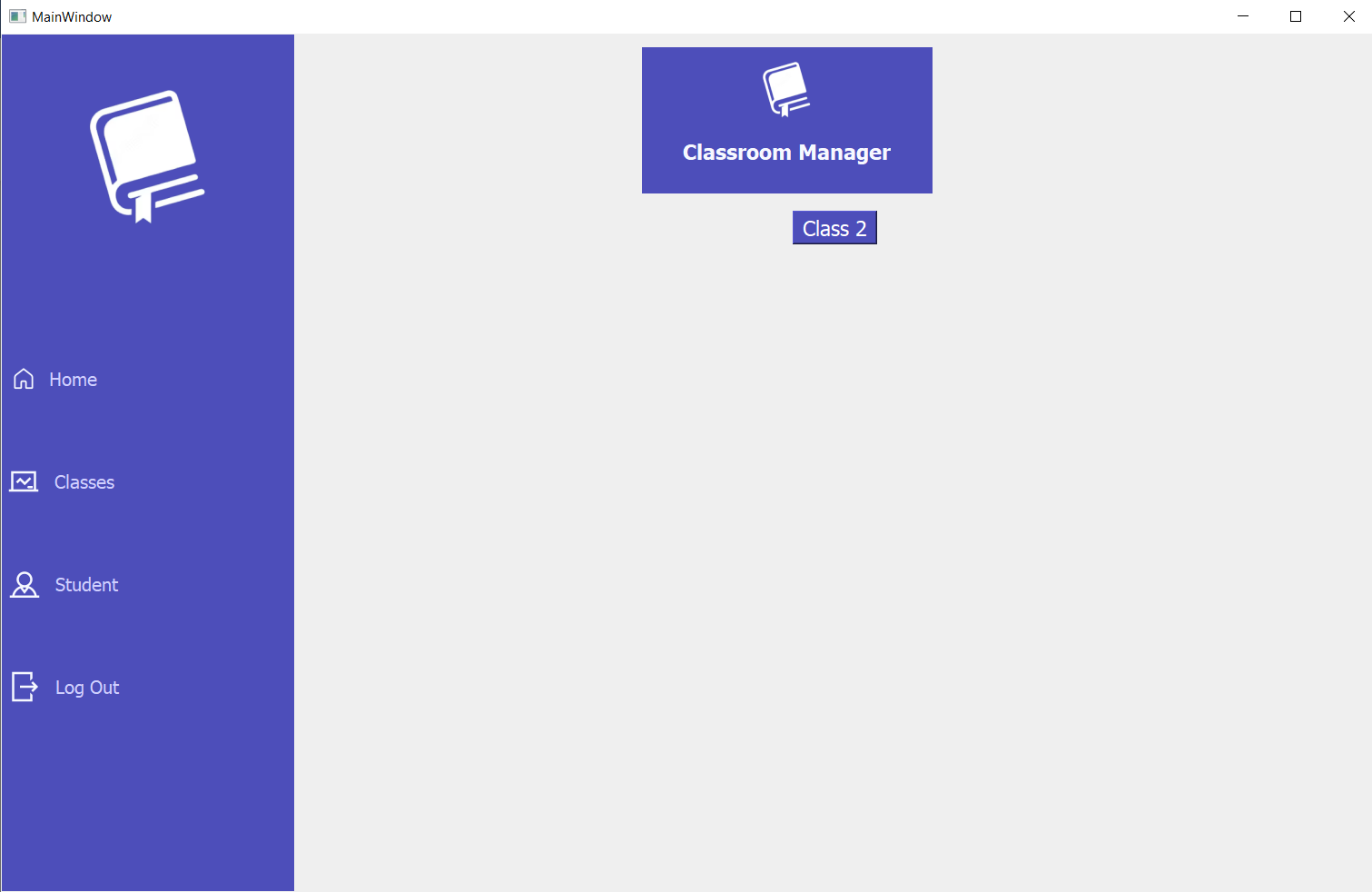
def \_\_init\_\_(self,teacherId):

CurrentId=teacherId

As we should know the who is current user so when logging in user we authenticate the user and after that we pass that user id

Teacher Page has menu in which there are two buttons 1) Home 2)Class 3)Student 4)Logout . Logout button will logout the current user from the application.

self.btnLogOut.clicked.connect(lambda:self.logOut()) Line-217

In default teacher homepage will load where that particular teacher classes will be shown which are generating dynamically by this function

# Call the showingAllClasses method to populate the ClassList widget with all the classes

self.showingAllClasses(CurrentId) Line-202

I will explain this function as it is import one

def showingAllClasses(self,CurrentId):

We have passed current teacher Id

# Clear the home page

self.clearHomePage()

This clearHomePage function is use to clear all the layout so new classes can be loaded otherwise by clicking home button same classes will added again and again

# Create a label to display if there are no classes found

lbl = QLabel("No classes found")

lbl.setStyleSheet('color: rgb(77, 78, 186);font-size: 20pt;')

It is simply creating a layout to show no classes found(Note : This is not working)

# Connect to the database

conn = sqlite3.connect("DB.db")

cur = conn.cursor()

# Retrieve all the class names for the current teacher

cur.execute("SELECT ClassName FROM Class WHERE TeacherId = ?", (CurrentId,))

classes = cur.fetchall()

cur.close()

conn.close()

Selecting all classes which current teacher have created from database

# Add buttons to the layout

layout = self.HomePage.layout()

layout.setAlignment(Qt.AlignTop | Qt.AlignHCenter)

layout.setSpacing(20)

Getting homepage layout in the layout and Setting its alignment

for cls in classes:

Creating a loop for number of classes found

# Clear the label if classes are found

lbl.setText("")

# Create a button for each class

btn = QPushButton(cls[0])

btn.setStyleSheet('background-color: rgb(77, 78, 186); color: white;font-size: 14pt;')

creating a button setting its name as className using cls[0] and setting its style sheet

# Add the button to the layout

layout.addWidget(btn)

Adding a btn to layout

# Function to handle the button click event

As we do not know the names of the functions so we create a function inside a function and checking which button is call and connecting a class details function

def on\_button\_clicked():

self.btnClicked = self.sender()

self.classDetails(CurrentId,self.btnClicked.text())

# Connect all buttons to the same function

Connecting all button to same function if any button of this is clicked we can connect the function

for i in range(layout.count()):

widget = layout.itemAt(i).widget()

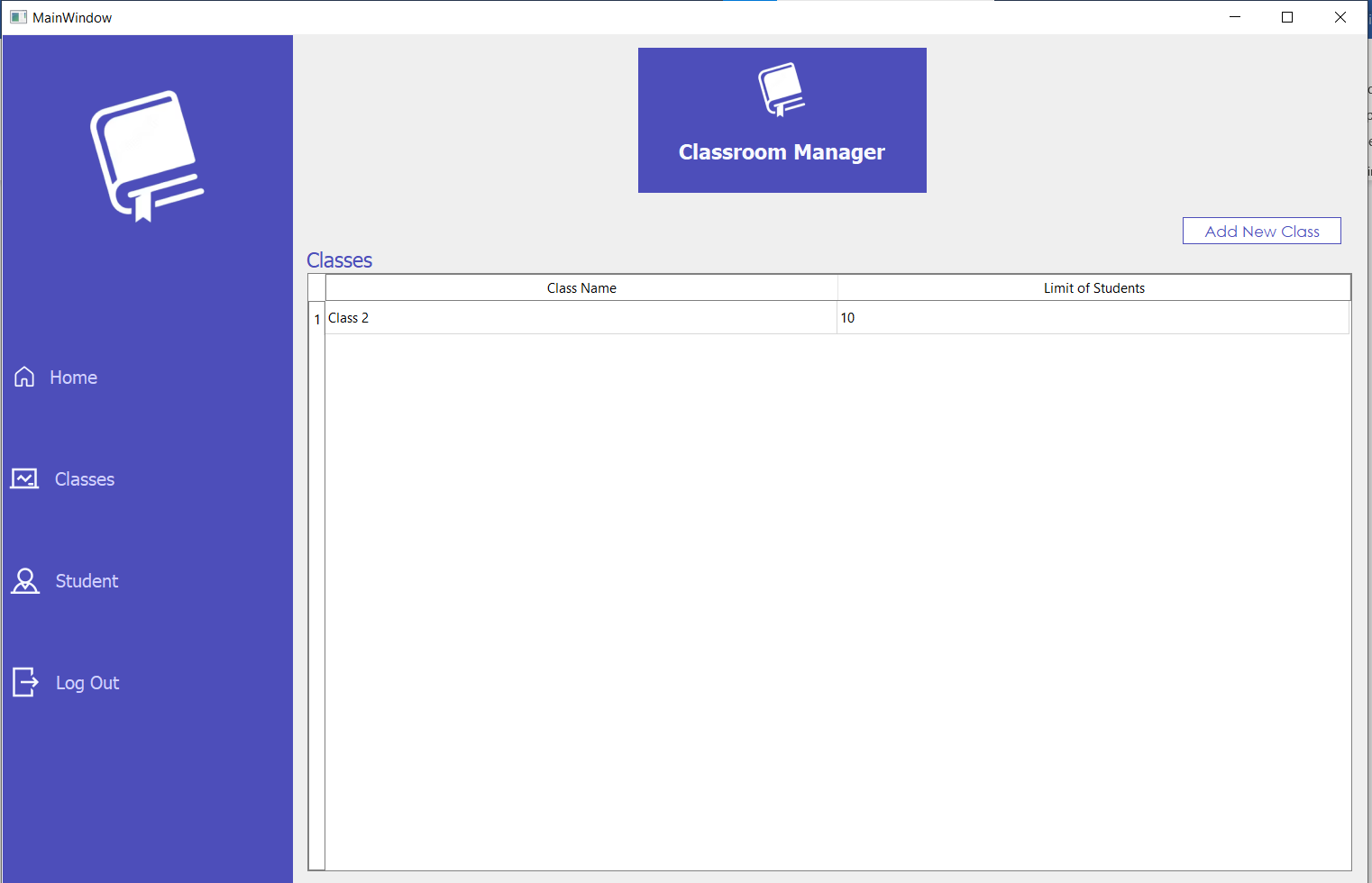
if isinstance(widget, QPushButton):

widget.clicked.connect(on\_button\_clicked)

Function End-------------------------------------------------------------------------------------

By clicking on class button menu, we connect this page and also call function to fill the table in this class

self.btnHomeMenu.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.HomePage))

 self.btnMenuClass.clicked.connect(lambda:self.fillClassTable(CurrentId))

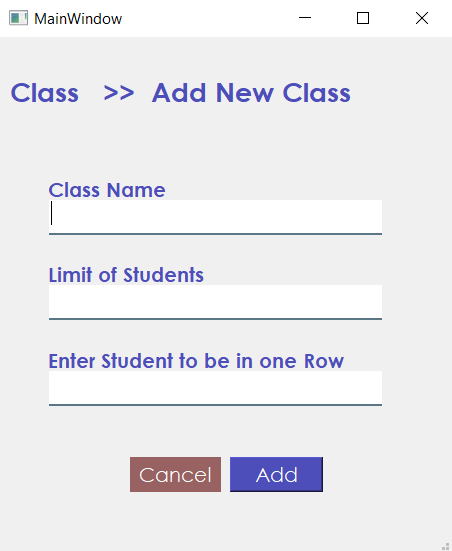
By Clicking on Add new class we connect AddNewClass Class

**Class AddNewClass:**

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

# idx: current user id

currentId=idx

Current id Is also pass so we can know which teacher have added class. 

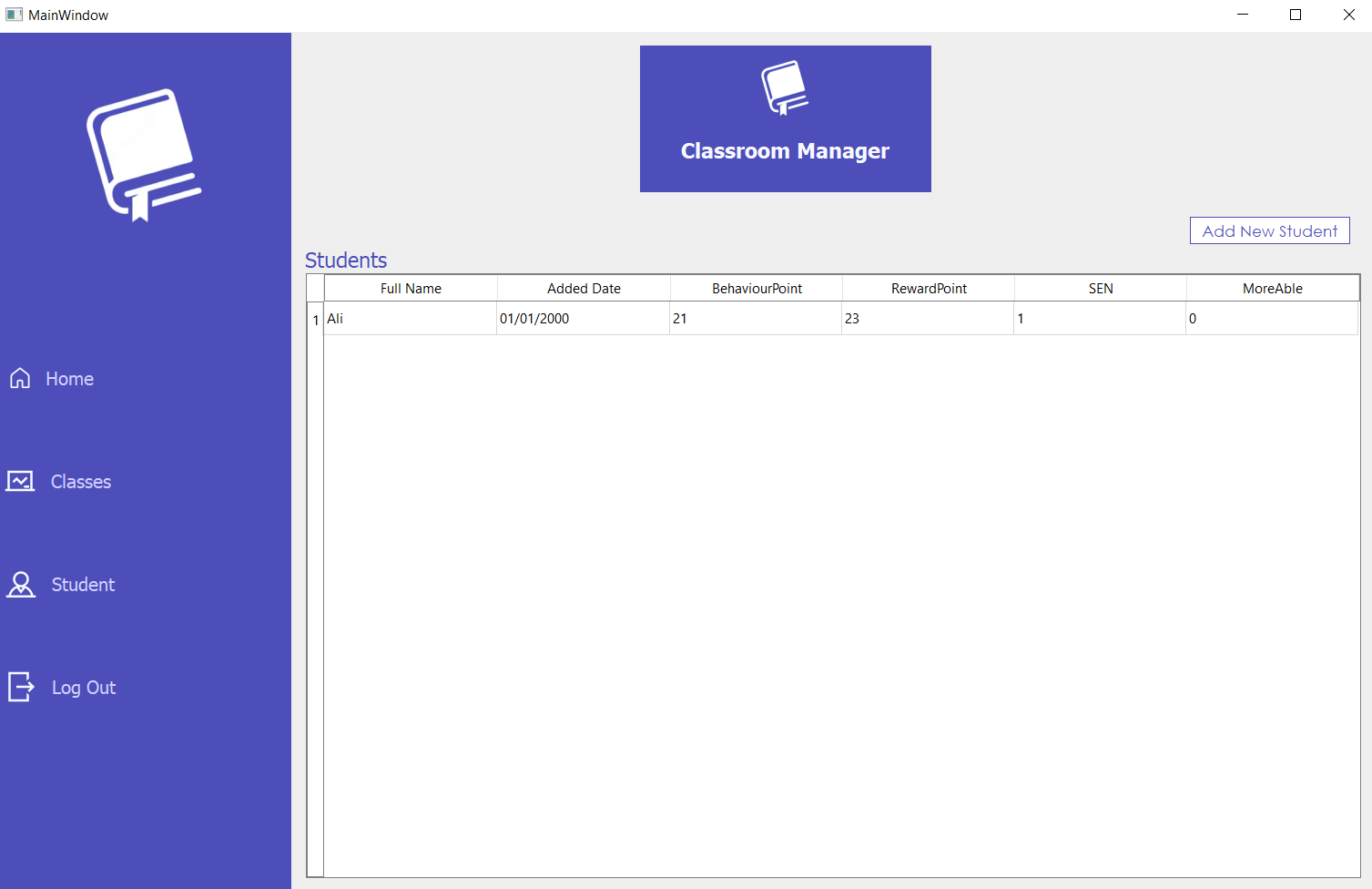
By cicking add we can add class to database

self.btnAdd.clicked.connect(lambda: self.addNewClassToDB(currentId))

By clicking on student button menu, we connect this page and also call function to fill the table

self.btnStudent.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.StudentPage))

self.btnStudent.clicked.connect(lambda:self.fillStudentTable())



By clicking on AddNewStudent we connect new class AddNewStudent

self.btnAddNewStudent.clicked.connect(lambda:self.addNewStudent(CurrentId))

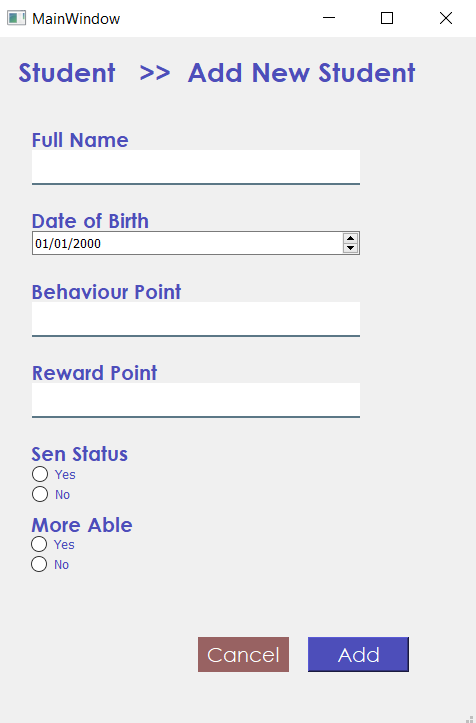
**Class AddNewStudent:**

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

# idx: current user id

currentId=idx

Current id Is also pass so we can know which teacher have added student

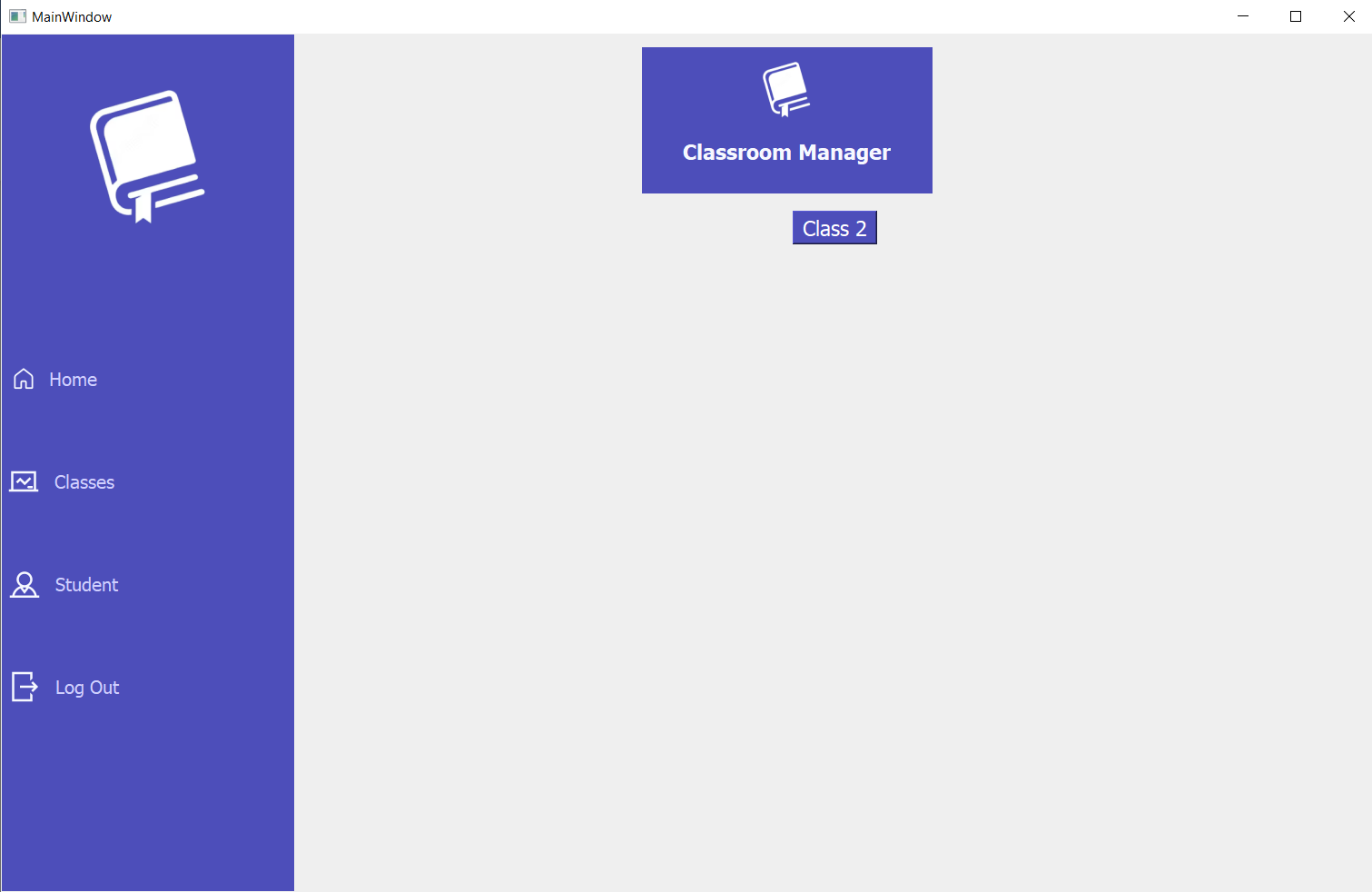


By Clicking add we connect following function

self.btnAdd.clicked.connect(lambda: self.addNewStudentToDB(currentId))

Now we return to homepage.

self.btnHomeMenu.clicked.connect(lambda: self.stackedWidget.setCurrentWidget(self.HomePage))



On clicking class we connect new window of following class

**Class ClassDetailsWindow:**

def \_\_init\_\_(self,idx,btnName):

# Initialize the object with two input arguments, idx and btnName

self.currentId=idx

self.btnName=btnName

self.ClassId=None

btnName is passed and through this button name we can find ClassId

def fillSeats(self):

First getting Id,Class Limit and student to be in one row from db

# connect to the database

conn = sqlite3.connect("DB.db")

cur = conn.cursor()

# execute a SELECT query to get the class details

cur.execute("SELECT ClassLimit,StudentInOneRow,Id FROM Class WHERE TeacherId = ? and ClassName = ?", (self.currentId,self.btnName,))

classes = cur.fetchall()

# close the cursor and the database connection

cur.close()

conn.close()

Saving them in varaibles

# extract the relevant class details

total\_seats = classes[0][0]

num\_columns = classes[0][1]

self.ClassId = classes[0][2]

Creating a grid

# create a grid layout to hold the seats

grid = QGridLayout()

# create and add labels for each chair with a border

Running a loop for total seats

for i in range(total\_seats):

# calculate the row and column for this chair

row = i // num\_columns

col = i % num\_columns

# create a frame to hold the chair label

chair\_frame = QFrame()

chair\_frame.setStyleSheet("border: 2px solid #4d4eba; ")

chair\_frame.setFrameShape(QFrame.Box)

# create a layout for the chair frame

chair\_frame\_layout = QGridLayout(chair\_frame)

# create a label for the chair

chair\_label = QLabel(f"Chair {i+1}")

chair\_label.setStyleSheet("QFrame { color: #4d4eba; }")

chair\_label.setAlignment(Qt.AlignCenter)

chair\_label.setAcceptDrops(True)

# add the chair label to the chair frame layout

chair\_frame\_layout.addWidget(chair\_label, 0, 0)

# add the chair frame to the grid layout

grid.addWidget(chair\_frame, row, col)

# set the minimum height of the chair frame

chair\_frame.setMinimumHeight(70)

chair\_frame.setAcceptDrops(True)

# set the cursor to a pointing hand

chair\_frame.setCursor(Qt.PointingHandCursor)

Adding them to layout

grid\_widget = QWidget()

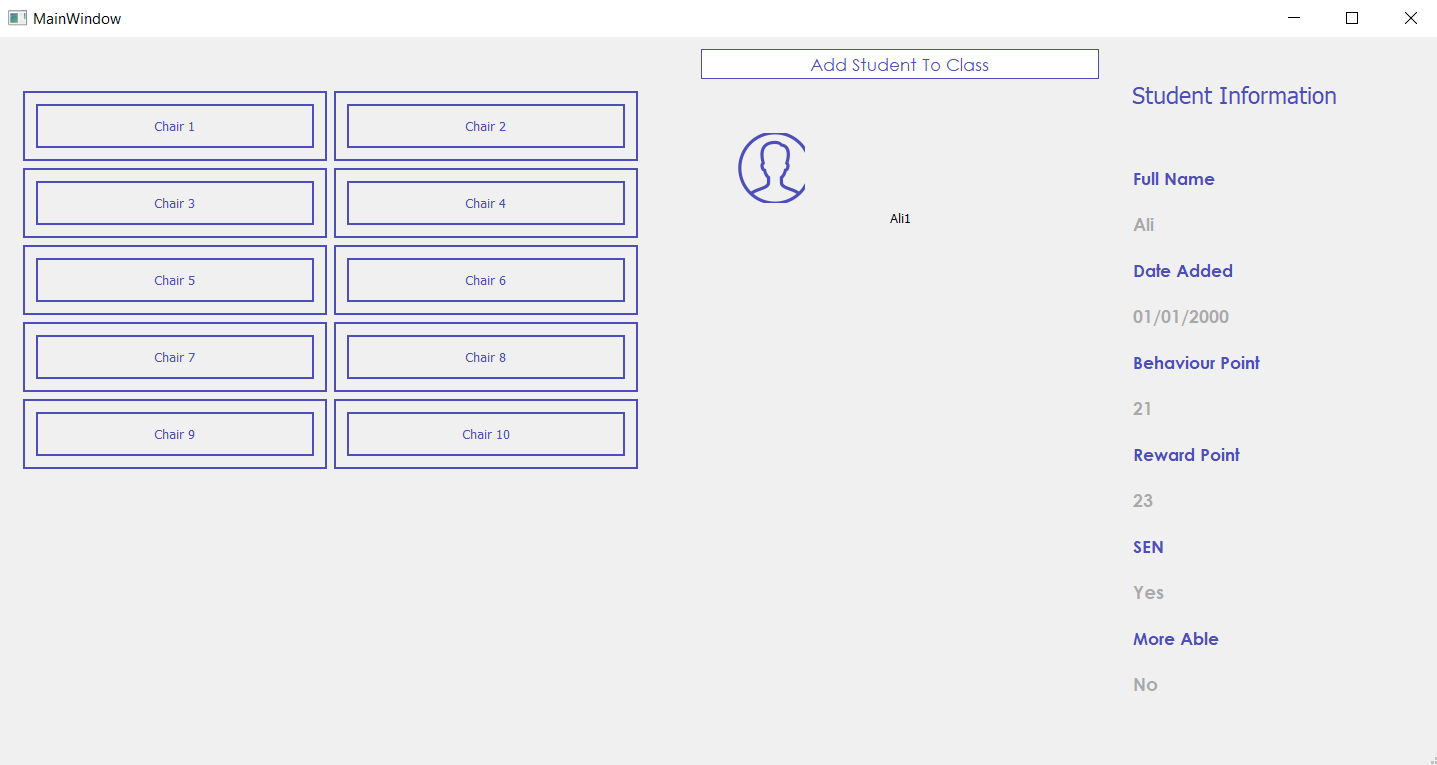
grid\_widget.setLayout(grid)

grid\_widget.setStyleSheet("border: none;")

saving seats layout in the seats\_layout and adding all grids in it

seats\_layout = self.Seats.layout()

seats\_layout.addRow(grid\_widget)



This window shows chair and all students enrolled in this class. We can add new student through button Add Student To Class

**Class AddStudentToClass:**

def \_\_init\_\_(self, teacherId, Id):

# Set class ID and teacher ID attributes

self.ClassId = Id

self.TeacherId = teacherId

ClassId is passed so we can know to which class student is added and teacherId is passed so we can know which teacher is added student

Graphical user interface

Description automatically generated

It contains all students which are not enrolled in this course we can enroll them by add button corresponding to each student