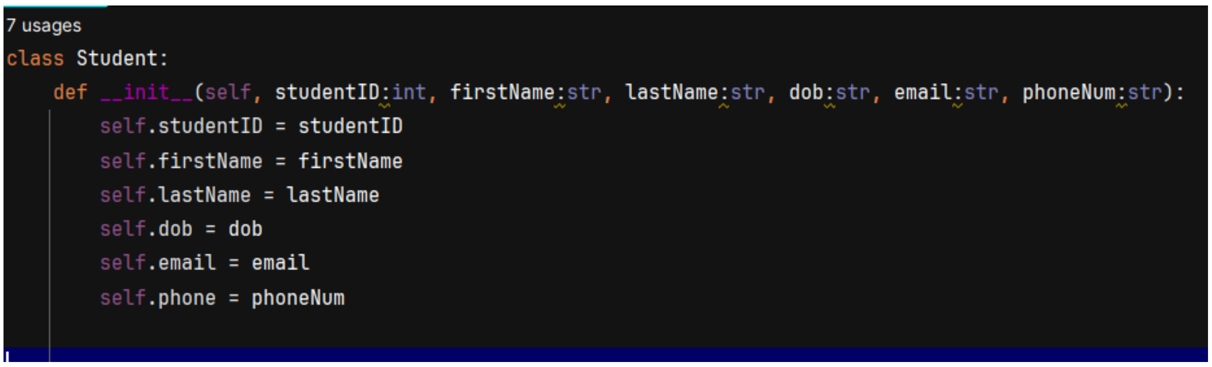
Name: Ayush Padhy

Assignment: 3-Student Information Service

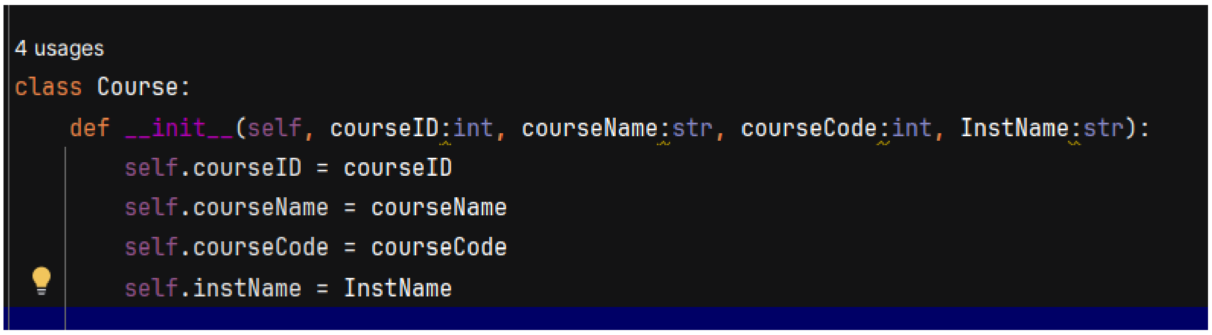
TASK 1: CREATING CLASSES

1. Student:



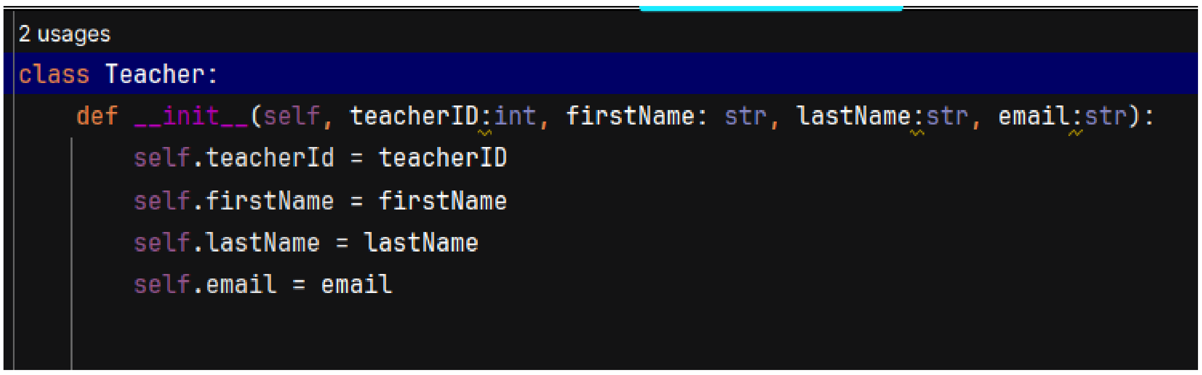
```
7 usages
class Student:
    def __init__(self, studentID:int, firstName:str, lastName:str, dob:str, email:str, phoneNum:str):
        self.studentID = studentID
        self.firstName = firstName
        self.lastName = lastName
        self.dob = dob
        self.email = email
        self.phone = phoneNum
```

2. Course:



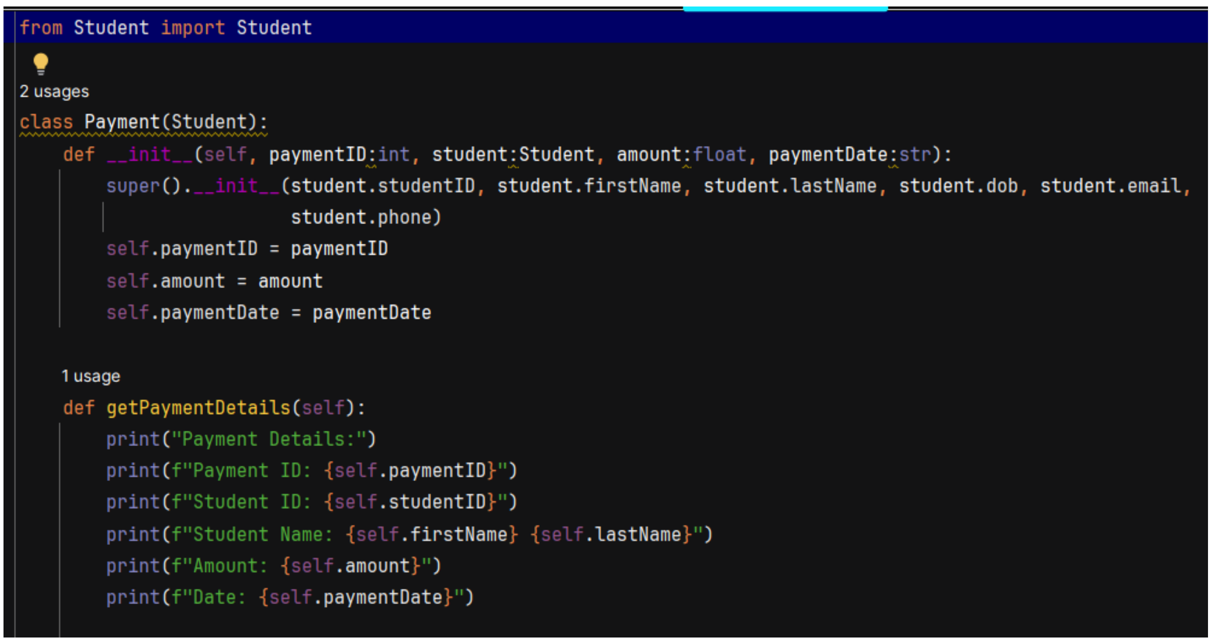
```
4 usages
class Course:
    def __init__(self, courseID:int, courseName:str, courseCode:int, InstName:str):
        self.courseID = courseID
        self.courseName = courseName
        self.courseCode = courseCode
        self.instName = InstName
```

3. Teacher:



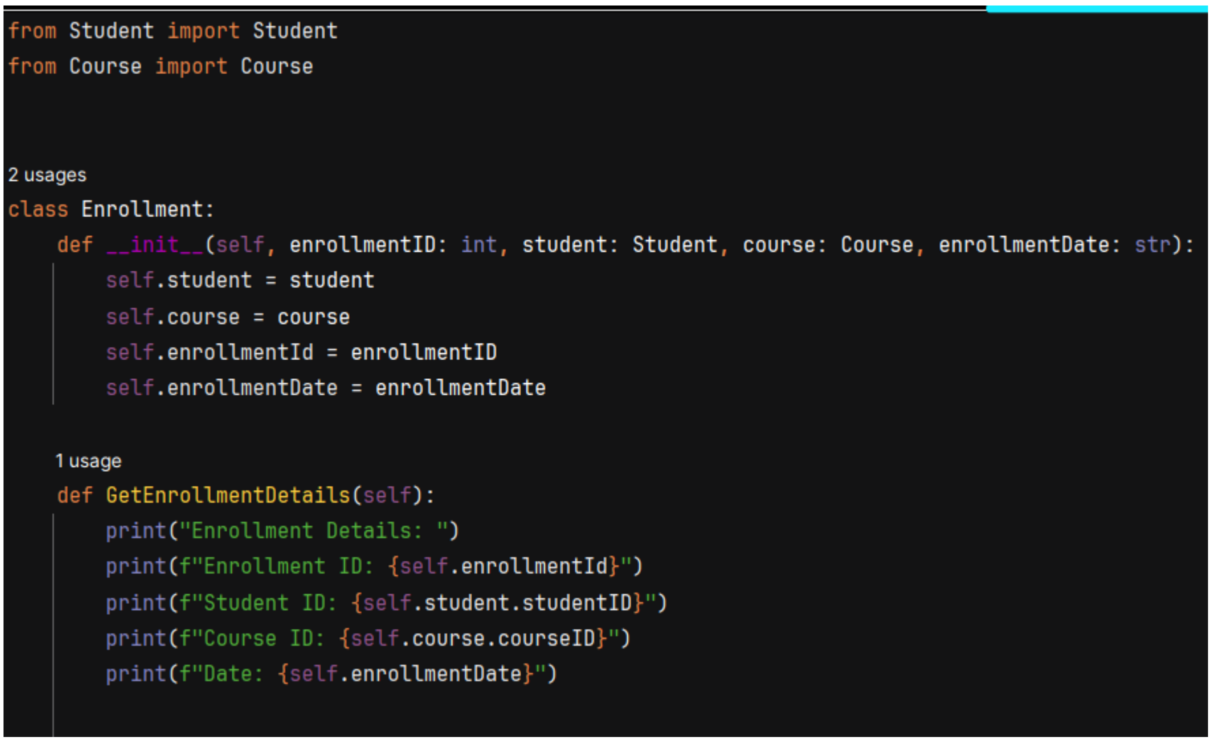
```
2 usages
class Teacher:
    def __init__(self, teacherID:int, firstName: str, lastName:str, email:str):
        self.teacherId = teacherID
        self.firstName = firstName
        self.lastName = lastName
        self.email = email
```

4. Payment:



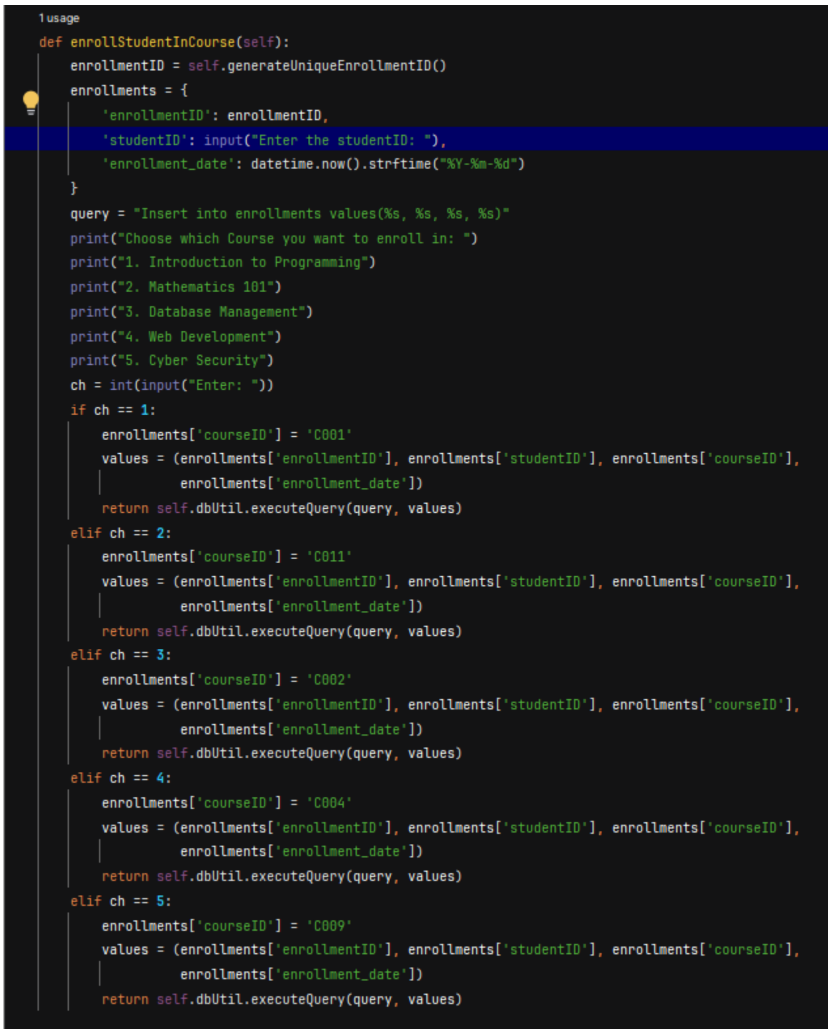
```
from Student import Student

2 usages
class Payment(Student):
    def __init__(self, paymentID:int, student:Student, amount:float, paymentDate:str):
        super().__init__(student.studentID, student.firstName, student.lastName, student.dob, student.email,
                         student.phone)
        self.paymentID = paymentID
        self.amount = amount
        self.paymentDate = paymentDate

    1 usage
    def getPaymentDetails(self):
        print("Payment Details:")
        print(f"Payment ID: {self.paymentID}")
        print(f"Student ID: {self.studentID}")
        print(f"Student Name: {self.firstName} {self.lastName}")
        print(f"Amount: {self.amount}")
        print(f"Date: {self.paymentDate}")
```

5. Enrollment:

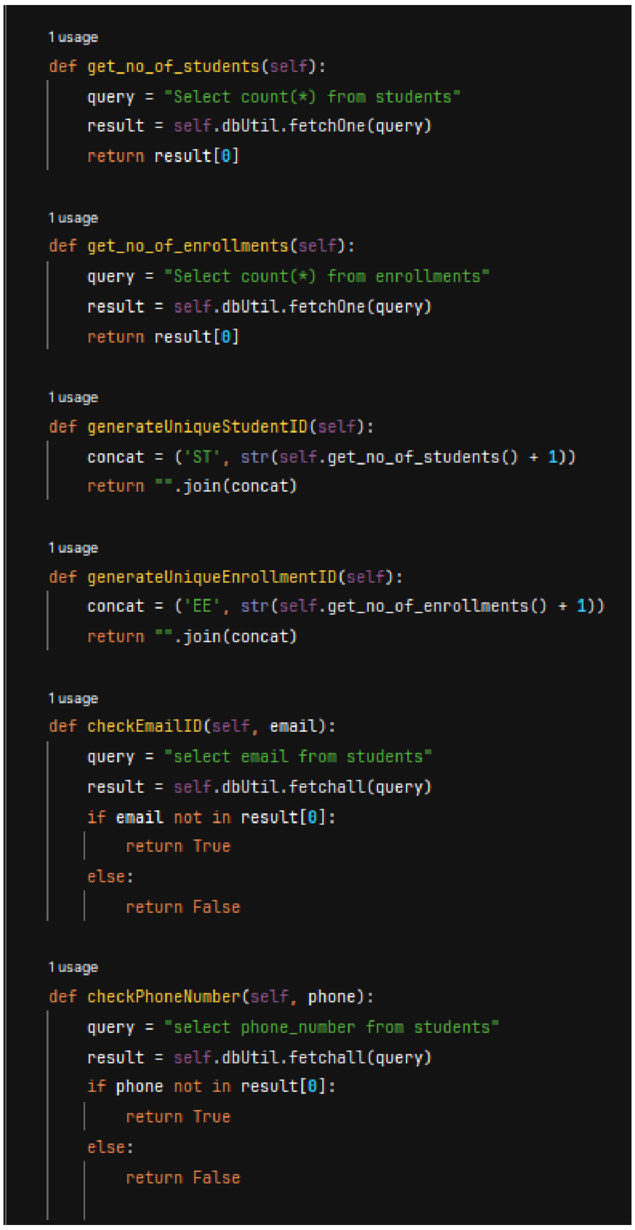


```
from Student import Student
from Course import Course


2 usages
class Enrollment:
    def __init__(self, enrollmentID: int, student: Student, course: Course, enrollmentDate: str):
        self.student = student
        self.course = course
        self.enrollmentId = enrollmentID
        self.enrollmentDate = enrollmentDate

    1 usage
    def GetEnrollmentDetails(self):
        print("Enrollment Details: ")
        print(f"Enrollment ID: {self.enrollmentId}")
        print(f"Student ID: {self.student.studentID}")
        print(f"Course ID: {self.course.courseID}")
        print(f"Date: {self.enrollmentDate}")
```

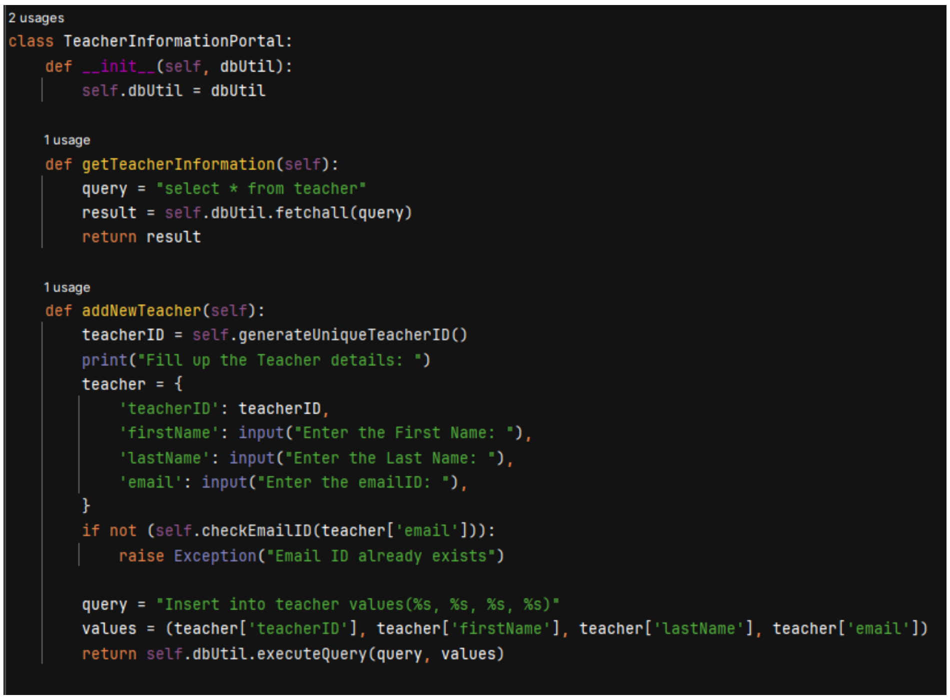
TASK 2: STUDENT INFORMATION PORTAL



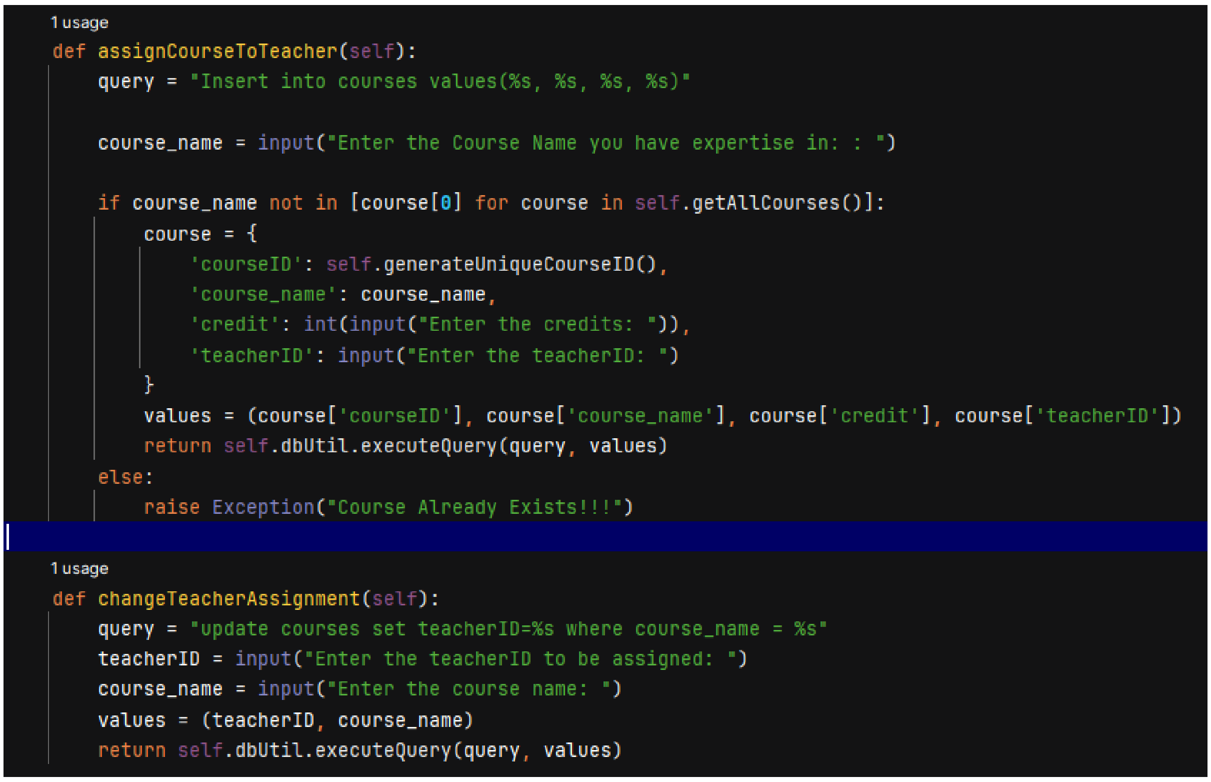
```
1 usage
def enrollStudentInCourse(self):
    enrollmentID = self.generateUniqueEnrollmentID()
    enrollments = {
        'enrollmentID': enrollmentID,
        'studentID': input("Enter the studentID: "),
        'enrollment_date': datetime.now().strftime("%Y-%m-%d")
    }
    query = "Insert into enrollments values(%s, %s, %s, %s)"
    print("Choose which Course you want to enroll in: ")
    print("1. Introduction to Programming")
    print("2. Mathematics 101")
    print("3. Database Management")
    print("4. Web Development")
    print("5. Cyber Security")
    ch = int(input("Enter: "))
    if ch == 1:
        enrollments['courseID'] = 'C001'
        values = (enrollments['enrollmentID'], enrollments['studentID'], enrollments['courseID'],
                  enrollments['enrollment_date'])
        return self.dbUtil.executeQuery(query, values)
    elif ch == 2:
        enrollments['courseID'] = 'C011'
        values = (enrollments['enrollmentID'], enrollments['studentID'], enrollments['courseID'],
                  enrollments['enrollment_date'])
        return self.dbUtil.executeQuery(query, values)
    elif ch == 3:
        enrollments['courseID'] = 'C002'
        values = (enrollments['enrollmentID'], enrollments['studentID'], enrollments['courseID'],
                  enrollments['enrollment_date'])
        return self.dbUtil.executeQuery(query, values)
    elif ch == 4:
        enrollments['courseID'] = 'C004'
        values = (enrollments['enrollmentID'], enrollments['studentID'], enrollments['courseID'],
                  enrollments['enrollment_date'])
        return self.dbUtil.executeQuery(query, values)
    elif ch == 5:
        enrollments['courseID'] = 'C009'
        values = (enrollments['enrollmentID'], enrollments['studentID'], enrollments['courseID'],
                  enrollments['enrollment_date'])
        return self.dbUtil.executeQuery(query, values)
```


```
1 usage
def get_no_of_students(self):
    query = "Select count(*) from students"
    result = self.dbUtil.fetchOne(query)
    return result[0]

1 usage
def get_no_of_enrollments(self):
    query = "Select count(*) from enrollments"
    result = self.dbUtil.fetchOne(query)
    return result[0]

1 usage
def generateUniqueStudentID(self):
    concat = ('ST', str(self.get_no_of_students() + 1))
    return "".join(concat)

1 usage
def generateUniqueEnrollmentID(self):
    concat = ('EE', str(self.get_no_of_enrollments() + 1))
    return "".join(concat)

1 usage
def checkEmailID(self, email):
    query = "select email from students"
    result = self.dbUtil.fetchall(query)
    if email not in result[0]:
        return True
    else:
        return False

1 usage
def checkPhoneNumber(self, phone):
    query = "select phone_number from students"
    result = self.dbUtil.fetchall(query)
    if phone not in result[0]:
        return True
    else:
        return False
```

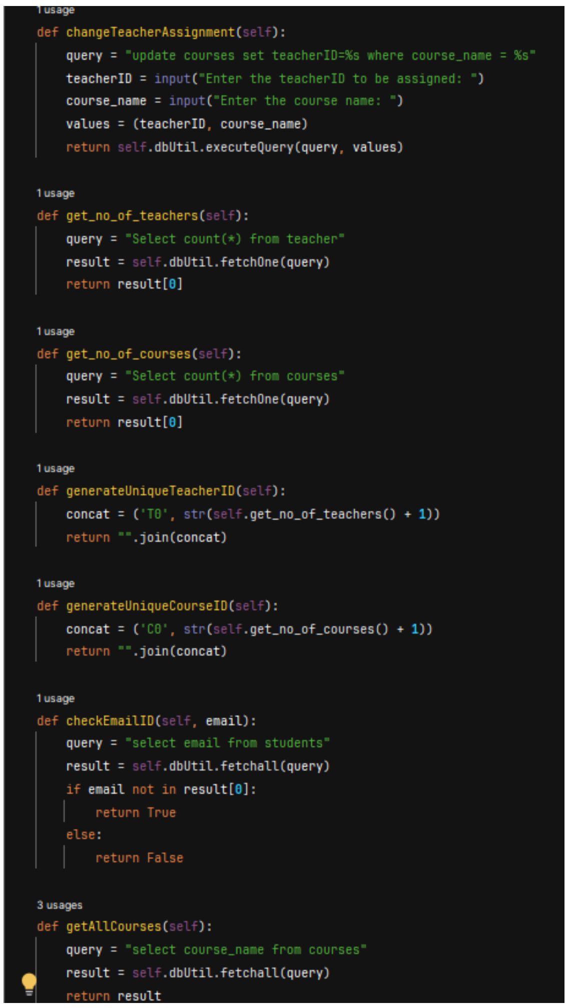
TASK 3: TEACHER INFORMATION PORTAL



```
2 usages
class TeacherInformationPortal:
    def __init__(self, dbUtil):
        self.dbUtil = dbUtil

    1 usage
    def getTeacherInformation(self):
        query = "select * from teacher"
        result = self.dbUtil.fetchall(query)
        return result

    1 usage
    def addNewTeacher(self):
        teacherID = self.generateUniqueTeacherID()
        print("Fill up the Teacher details: ")
        teacher = {
            'teacherID': teacherID,
            'firstName': input("Enter the First Name: "),
            'lastName': input("Enter the Last Name: "),
            'email': input("Enter the emailID: "),
        }
        if not (self.checkEmailID(teacher['email'])):
            raise Exception("Email ID already exists")

        query = "Insert into teacher values(%s, %s, %s, %s)"
        values = (teacher['teacherID'], teacher['firstName'], teacher['lastName'], teacher['email'])
        return self.dbUtil.executeQuery(query, values)
```

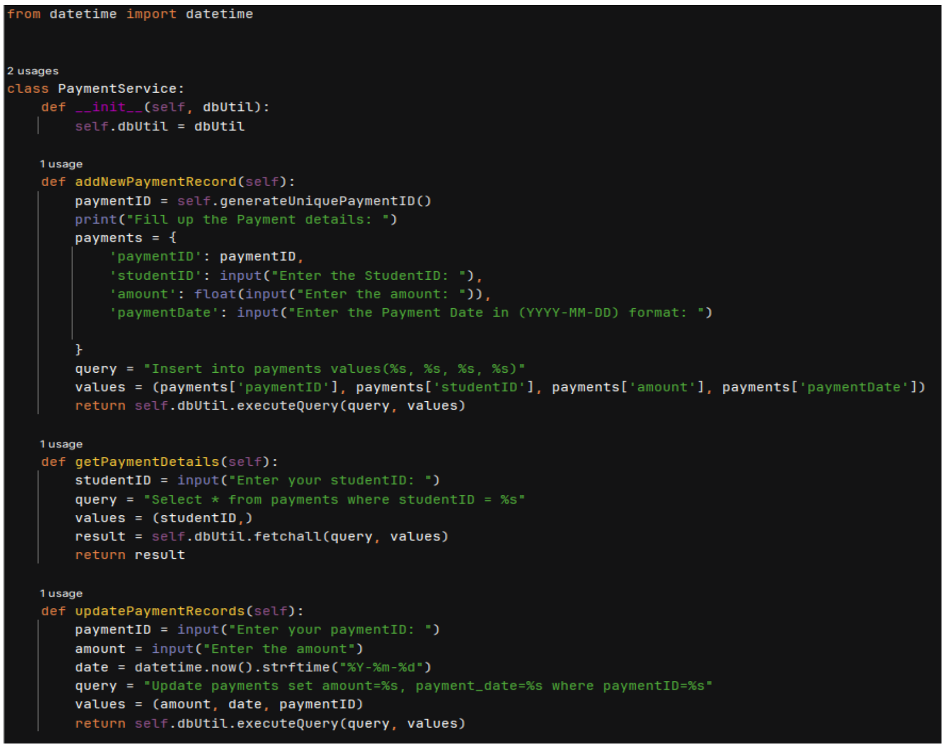


```
    1 usage
    def assignCourseToTeacher(self):
        query = "Insert into courses values(%s, %s, %s, %s)"

        course_name = input("Enter the Course Name you have expertise in: : ")

        if course_name not in [course[0] for course in self.getAllCourses()]:
            course = {
                'courseID': self.generateUniqueCourseID(),
                'course_name': course_name,
                'credit': int(input("Enter the credits: ")),
                'teacherID': input("Enter the teacherID: ")
            }
            values = (course['courseID'], course['course_name'], course['credit'], course['teacherID'])
            return self.dbUtil.executeQuery(query, values)
        else:
            raise Exception("Course Already Exists!!!")

    1 usage
    def changeTeacherAssignment(self):
        query = "update courses set teacherID=%s where course_name = %s"
        teacherID = input("Enter the teacherID to be assigned: ")
        course_name = input("Enter the course name: ")
        values = (teacherID, course_name)
        return self.dbUtil.executeQuery(query, values)
```

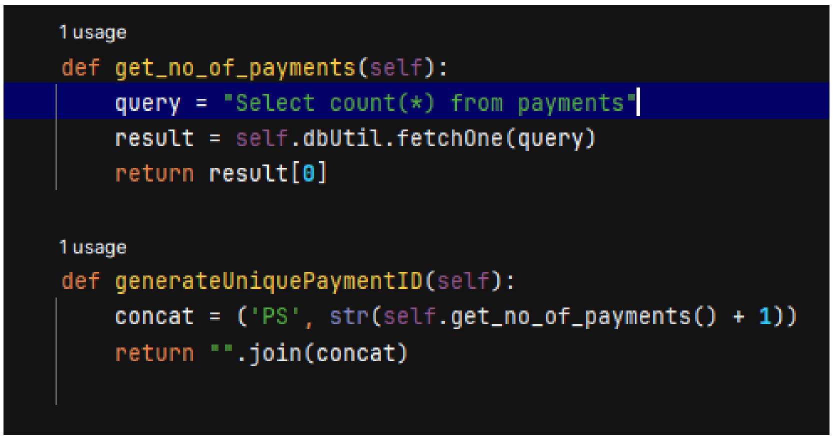


```
1 usage
def changeTeacherAssignment(self):
    query = "update courses set teacherID=%s where course_name = %s"
    teacherID = input("Enter the teacherID to be assigned: ")
    course_name = input("Enter the course name: ")
    values = (teacherID, course_name)
    return self.dbUtil.executeQuery(query, values)

1 usage
def get_no_of_teachers(self):
    query = "Select count(*) from teacher"
    result = self.dbUtil.fetchOne(query)
    return result[0]

1 usage
def get_no_of_courses(self):
    query = "Select count(*) from courses"
    result = self.dbUtil.fetchOne(query)
    return result[0]

1 usage
def generateUniqueTeacherID(self):
    concat = ('T0', str(self.get_no_of_teachers() + 1))
    return "".join(concat)

1 usage
def generateUniqueCourseID(self):
    concat = ('C0', str(self.get_no_of_courses() + 1))
    return "".join(concat)

1 usage
def checkEmailID(self, email):
    query = "select email from students"
    result = self.dbUtil.fetchall(query)
    if email not in result[0]:
        return True
    else:
        return False

3 usages
def getAllCourses(self):
    query = "select course_name from courses"
    result = self.dbUtil.fetchall(query)
    return result
```

TASK4: PAYMENT SERVICE

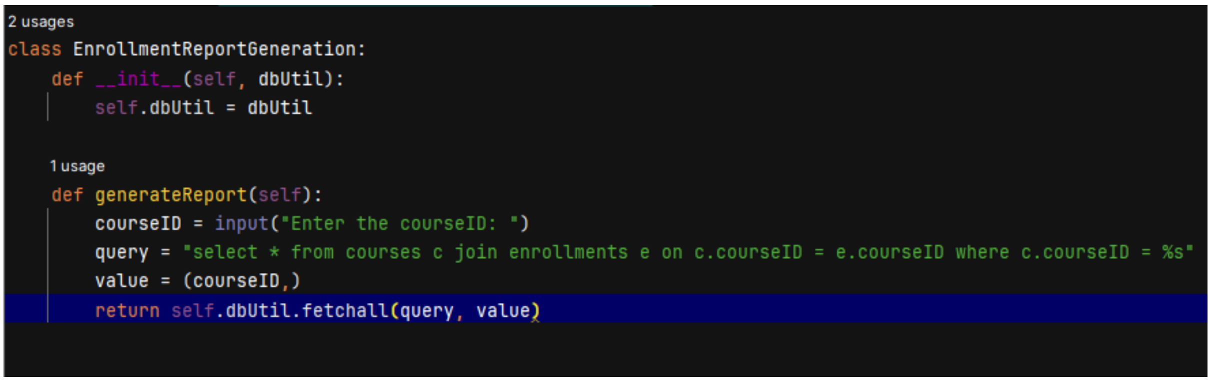


```
from datetime import datetime


2 usages
class PaymentService:
    def __init__(self, dbUtil):
        self.dbUtil = dbUtil

    1 usage
    def addNewPaymentRecord(self):
        paymentID = self.generateUniquePaymentID()
        print("Fill up the Payment details: ")
        payments = {
            'paymentID': paymentID,
            'studentID': input("Enter the StudentID: "),
            'amount': float(input("Enter the amount: ")),
            'paymentDate': input("Enter the Payment Date in (YYYY-MM-DD) format: ")

        }
        query = "Insert into payments values(%s, %s, %s, %s)"
        values = (payments['paymentID'], payments['studentID'], payments['amount'], payments['paymentDate'])
        return self.dbUtil.executeQuery(query, values)

    1 usage
    def getPaymentDetails(self):
        studentID = input("Enter your studentID: ")
        query = "Select * from payments where studentID = %s"
        values = (studentID,)
        result = self.dbUtil.fetchall(query, values)
        return result

    1 usage
    def updatePaymentRecords(self):
        paymentID = input("Enter your paymentID: ")
        amount = input("Enter the amount")
        date = datetime.now().strftime("%Y-%m-%d")
        query = "Update payments set amount=%s, payment_date=%s where paymentID=%s"
        values = (amount, date, paymentID)
        return self.dbUtil.executeQuery(query, values)
```



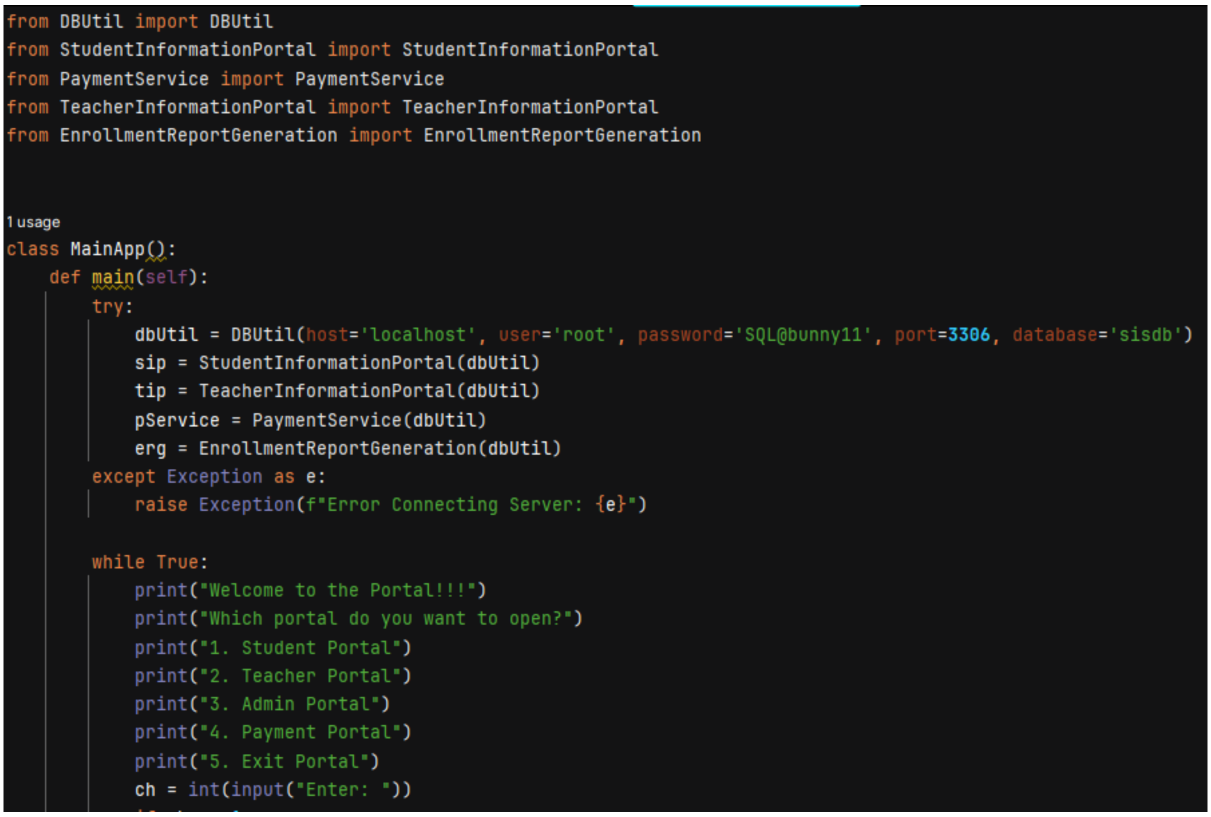
```
1 usage
def get_no_of_payments(self):
    query = "Select count(*) from payments"
    result = self.dbUtil.fetchOne(query)
    return result[0]

1 usage
def generateUniquePaymentID(self):
    concat = ('PS', str(self.get_no_of_payments() + 1))
    return "".join(concat)
```

TASK 5: ENROLLMENT REPORT GENERATION

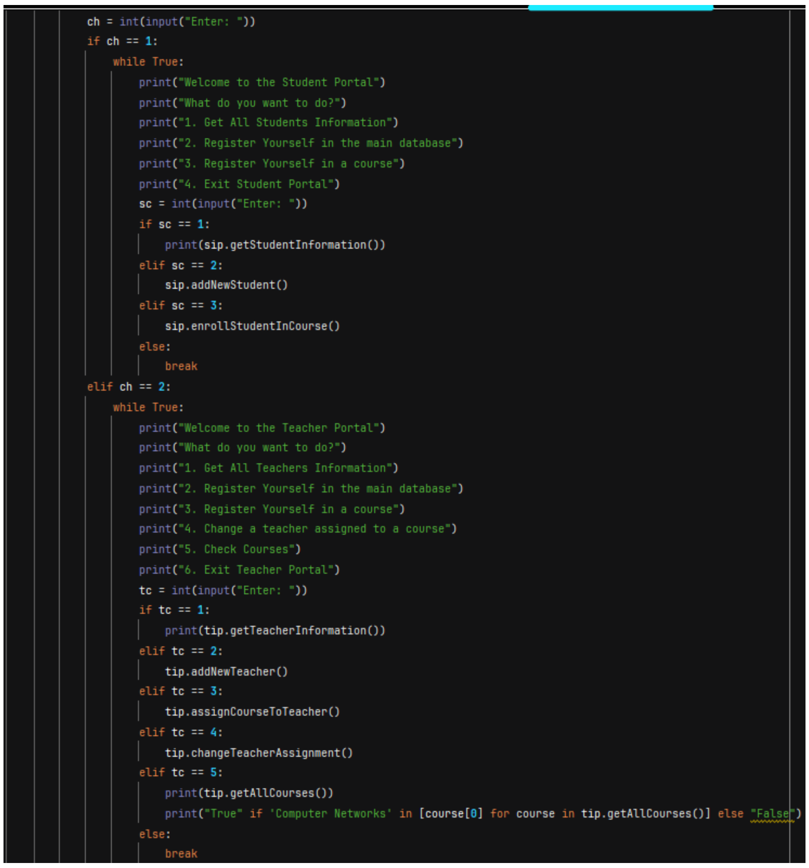


```
2 usages
class EnrollmentReportGeneration:
    def __init__(self, dbUtil):
        self.dbUtil = dbUtil

    1 usage
    def generateReport(self):
        courseID = input("Enter the courseID: ")
        query = "select * from courses c join enrollments e on c.courseID = e.courseID where c.courseID = %s"
        value = (courseID,)
        return self.dbUtil.fetchall(query, value)
```

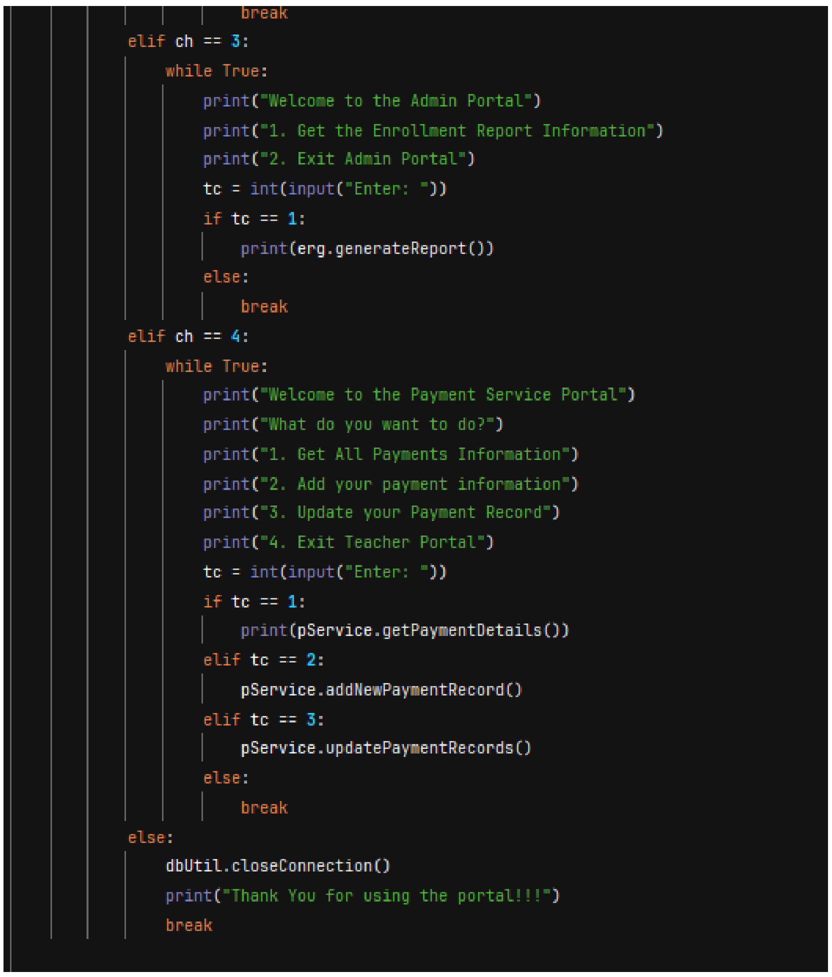
TASK 6: MAIN APP



```
from DBUtil import DBUtil
from StudentInformationPortal import StudentInformationPortal
from PaymentService import PaymentService
from TeacherInformationPortal import TeacherInformationPortal
from EnrollmentReportGeneration import EnrollmentReportGeneration


1 usage
class MainApp():
    def main(self):
        try:
            dbUtil = DBUtil(host='localhost', user='root', password='SQL@bunny11', port=3306, database='sisdb')
            sip = StudentInformationPortal(dbUtil)
            tip = TeacherInformationPortal(dbUtil)
            pService = PaymentService(dbUtil)
            erg = EnrollmentReportGeneration(dbUtil)
        except Exception as e:
            raise Exception(f"Error Connecting Server: {e}")

        while True:
            print("Welcome to the Portal!!!")
            print("Which portal do you want to open?")
            print("1. Student Portal")
            print("2. Teacher Portal")
            print("3. Admin Portal")
            print("4. Payment Portal")
            print("5. Exit Portal")
            ch = int(input("Enter: "))
```

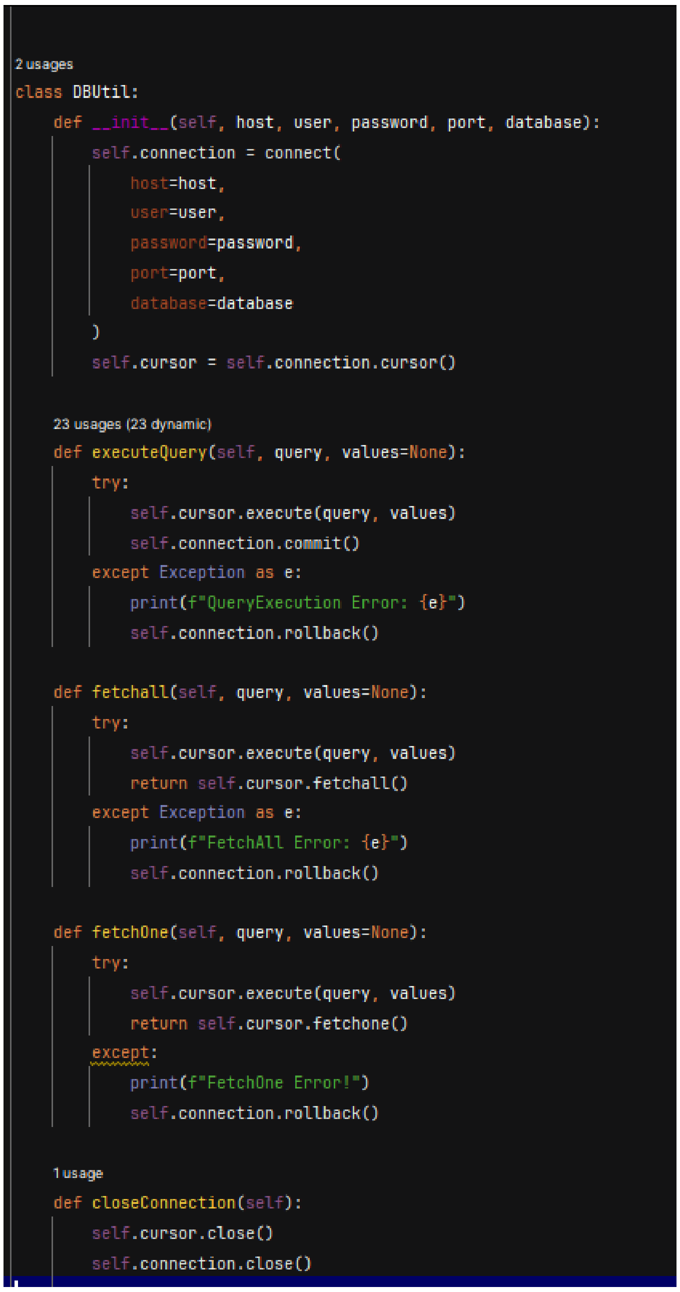


```python
ch = int(input("Enter: "))
if ch == 1:
    while True:
        print("Welcome to the Student Portal")
        print("What do you want to do?")
        print("1. Get All Students Information")
        print("2. Register Yourself in the main database")
        print("3. Register Yourself in a course")
        print("4. Exit Student Portal")
        sc = int(input("Enter: "))
        if sc == 1:
            print(sip.getStudentInformation())
        elif sc == 2:
            sip.addNewStudent()
        elif sc == 3:
            sip.enrollStudentInCourse()
        else:
            break
elif ch == 2:
    while True:
        print("Welcome to the Teacher Portal")
        print("What do you want to do?")
        print("1. Get All Teachers Information")
        print("2. Register Yourself in the main database")
        print("3. Register Yourself in a course")
        print("4. Change a teacher assigned to a course")
        print("5. Check Courses")
        print("6. Exit Teacher Portal")
        tc = int(input("Enter: "))
        if tc == 1:
            print(tip.getTeacherInformation())
        elif tc == 2:
            tip.addNewTeacher()
        elif tc == 3:
            tip.assignCourseToTeacher()
        elif tc == 4:
            tip.changeTeacherAssignment()
        elif tc == 5:
            print(tip.getAllCourses())
            print("True" if 'Computer Networks' in [course[0] for course in tip.getAllCourses()] else "False")
        else:
            break
```



```python
            break
elif ch == 3:
    while True:
        print("Welcome to the Admin Portal")
        print("1. Get the Enrollment Report Information")
        print("2. Exit Admin Portal")
        tc = int(input("Enter: "))
        if tc == 1:
            print(erg.generateReport())
        else:
            break
elif ch == 4:
    while True:
        print("Welcome to the Payment Service Portal")
        print("What do you want to do?")
        print("1. Get All Payments Information")
        print("2. Add your payment information")
        print("3. Update your Payment Record")
        print("4. Exit Teacher Portal")
        tc = int(input("Enter: "))
        if tc == 1:
            print(pService.getPaymentDetails())
        elif tc == 2:
            pService.addNewPaymentRecord()
        elif tc == 3:
            pService.updatePaymentRecords()
        else:
            break
else:
    dbUtil.closeConnection()
    print("Thank You for using the portal!!!")
    break
```

TASK 7: DATABASE CONNECTION



```
2 usages
class DBUtil:
    def __init__(self, host, user, password, port, database):
        self.connection = connect(
            host=host,
            user=user,
            password=password,
            port=port,
            database=database
        )
        self.cursor = self.connection.cursor()

    23 usages (23 dynamic)
    def executeQuery(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            self.connection.commit()
        except Exception as e:
            print(f"QueryExecution Error: {e}")
            self.connection.rollback()

    def fetchall(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            return self.cursor.fetchall()
        except Exception as e:
            print(f"FetchAll Error: {e}")
            self.connection.rollback()

    def fetchOne(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            return self.cursor.fetchone()
        except:
            print(f"FetchOne Error!")
            self.connection.rollback()

    1 usage
    def closeConnection(self):
        self.cursor.close()
        self.connection.close()
```