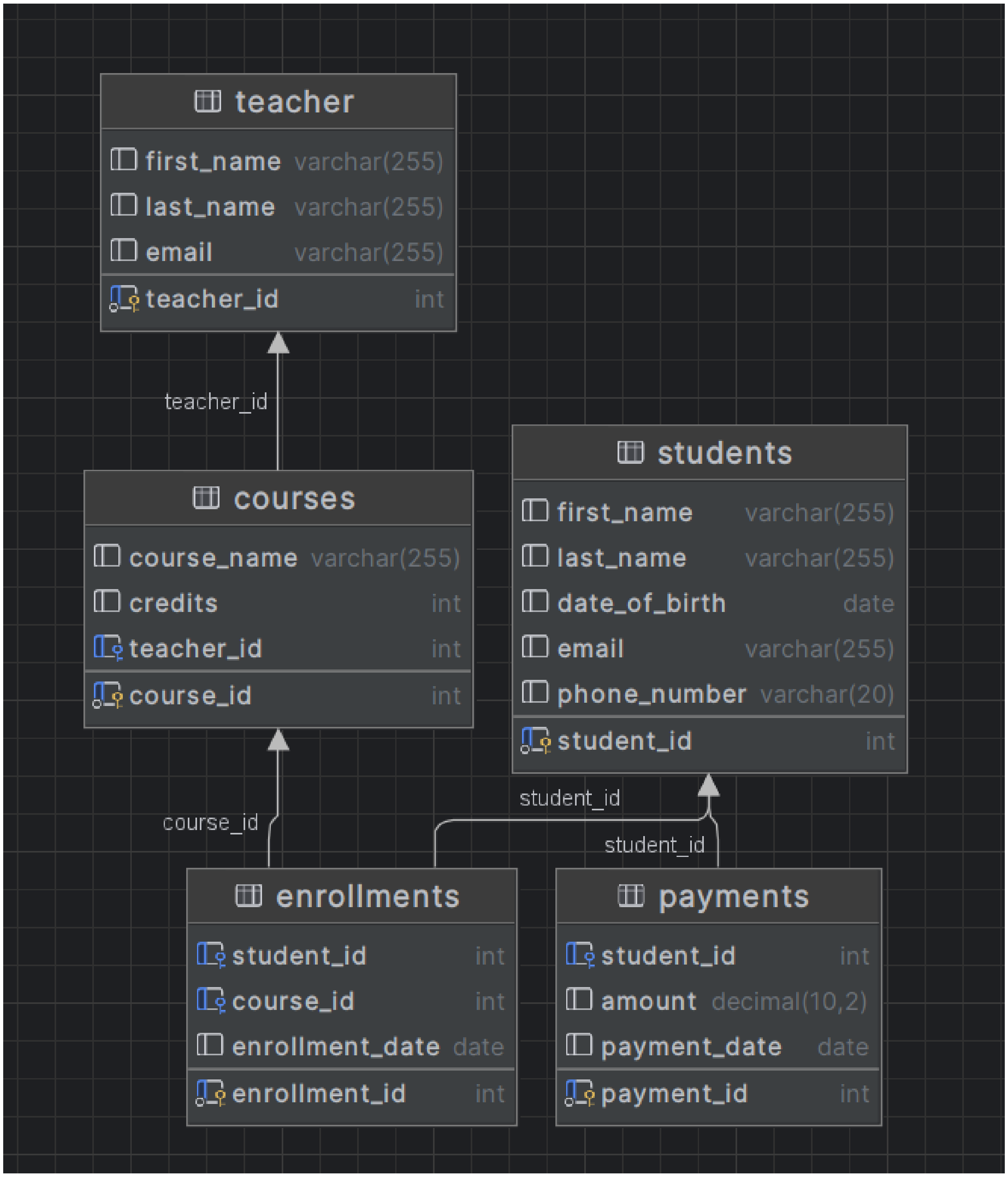
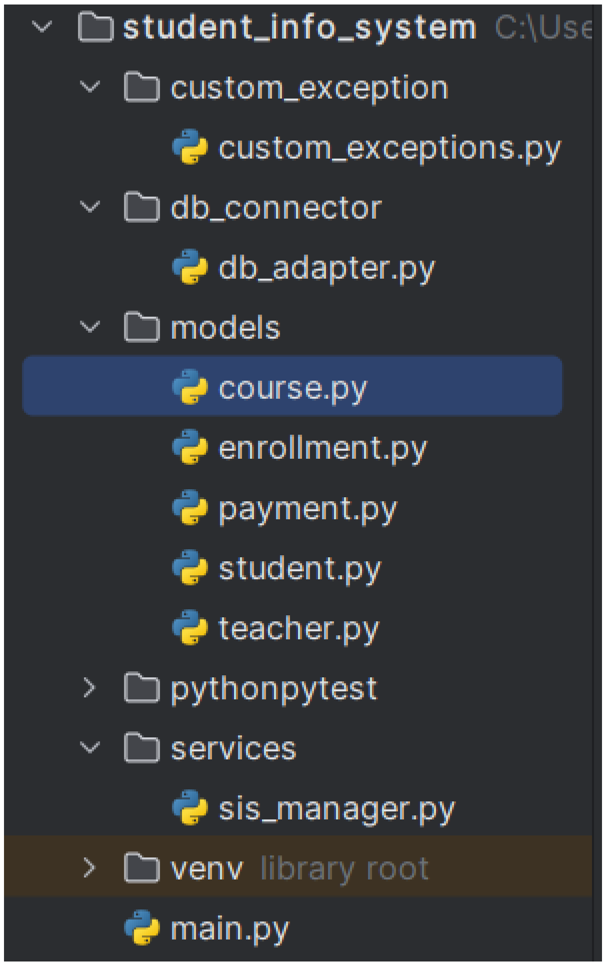
Student Information System

Database Schema



File system


```
student_info_system C:\Use
    custom_exception
        custom_exceptions.py
    db_connector
        db_adapter.py
    models
        course.py
        enrollment.py
        payment.py
        student.py
        teacher.py
    pythonpytest
    services
        sis_manager.py
    venv library root
    main.py
```

Custom_exceptions.py

```
class CourseNotFoundException(Exception):
   pass


class StudentNotFoundException(Exception):
   pass


class TeacherNotFoundException(Exception):
   pass


class PaymentValidationException(Exception):
    pass


class InvalidStudentDataException(Exception):
    pass


class InvalidCourseDataException(Exception):
    pass


class InvalidEnrollmentDataException(Exception):
    pass


class InvalidTeacherDataException(Exception):
    pass


class InsufficientFundsException(Exception):
    pass
```

Db_adapter.py

```
import mysql.connector


def get_db_connection():
    # Replace the following values with your MySQL server credentials
    config = {
        'user': 'root',
        'password': 'prakhar123',
        'host': 'localhost',
        'database': 'sisdb'
    }

    try:
        connection = mysql.connector.connect(**config)
        # print("Connected to the database")
        # print('hello World')
        return connection
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None


def get_ids(table_name, id_column_name):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT ' + id_column_name + ' FROM ' + table_name + ' ORDER BY ' +
id_column_name + ' DESC LIMIT 1'
    print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x) + 1
```

Course.py

```
from db_connector.db_adapter import get_db_connection
from models.teacher import Teacher
from models.enrollment import Enrollment


class Course:

    def __init__(self, course_id, course_name, credit, teacher: Teacher):
        self.connection = get_db_connection()
        self.__course_id = course_id
        self.__course_name = course_name
        self.__credits = credit
        self.__teacher = teacher

    def get_course_id(self):
        return self.__course_id

    def get_course_name(self):
        return self.__course_name

    def get_teacher(self):
        return self.__teacher

    def get_course_credits(self):
        return self.__credits

    def assign_teacher(self, teacher: Teacher):
        my_cursor = self.connection.cursor()
        sql = '''
            UPDATE Courses SET teacher_id = %s WHERE course_id = %s
            '''
        para = (teacher.get_teacher_id(), self.__course_id)
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Teacher Assigned Successfully')

    def update_course_info(self, course_name=None, credit=None):
        my_cursor = self.connection.cursor()
        if course_name:
            sql = '''
                                    UPDATE Courses SET course_name = %s WHERE course_id = %s
                                '''
            para = (course_name, self.__course_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__course_name = course_name

        if credit:
            sql = '''
                                    UPDATE Students SET credits = %s WHERE course_id = %s
                                '''
            para = (credit, self.__course_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__credits = credit

        print('Teacher Details Updated Successfully')

    def get_enrollments(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * Enrollments WHERE course_id = %s
            '''
            para = (self.__course_id,)
            my_cursor.execute(sql, para)
            x = [Enrollment(*list(i)) for i in list(my_cursor.fetchall())]
        except Exception as e:
            print(f'An error occurred {e}')

    def display_course_info(self):
        print('CourseID', self.__course_id)
        print('Course Name', self.__course_name)
        print('Course Credits', self.__credits)
        print('Teacher Details:-')
        print(self.__teacher)
```

Enrollment.py

```
from db_connector.db_adapter import get_db_connection


class Enrollment:
    def __init__(self, enrollment_id, student, course, enrollment_date):
        self.connection = get_db_connection()
        self.__enrollment_id = enrollment_id
        self.__student = student
        self.__course = course
        self.__enrollment_date = enrollment_date

    def get_enrollment_id(self):
        return self.__enrollment_id

    def get_student(self):
        return self.__student

    def get_course(self):
        return self.__course

    def get_enrollment_date(self):
        return self.__enrollment_date

    def display_enrollment_info(self):
        print('Enrollment ID: ', self.__enrollment_id)
        print('Student Details: ', self.__student)
        print('Course Details: ', self.__course)
        print('Enrollment Date: ',self.__enrollment_date)
```

Payment.py

```
from db_connector.db_adapter import get_db_connection


class Payment:

    def __init__(self, payment_id, student, amount, payment_date):
        self.connection = get_db_connection()
        self.__payment_id = payment_id
        self.__student = student
        self.__amount = amount
        self.__payment_date = payment_date

    def get_payment_id(self):
        return self.__payment_id

    def get_student(self):
        return self.__student

    def get_payment_amount(self):
        return self.__amount

    def get_payment_date(self):
        return self.__payment_date
    def display_payment_info(self):
        print('Payment ID: ', self.__payment_id)
        print('Student Details: ', self.__student)
        print('Payment Amount: ', self.__amount)
        print('Payment Date: ', self.__payment_date)
```

Student.py

```
from datetime import date
from db_connector.db_adapter import get_db_connection
from db_connector.db_adapter import get_ids


class Student:

    def __init__(self, student_id, firstname, lastname, dob, email,
phone_number):
        self.connection = get_db_connection()
        self.__student_id = student_id
        self.__firstname = firstname
        self.__lastname = lastname
        self.__dob = dob
        self.__email = email
        self.__phone_number = phone_number

    def get_student_id(self):
        return self.__student_id

    def get_first_name(self):
        return self.__firstname

    def get_last_name(self):
        return self.__lastname

    def get_dob(self):
        return self.__dob

    def get_email(self):
        return self.__email

    def get_phone_number(self):
        return self.__phone_number

    def enroll_in_course(self, course):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
```

```python
                    INSERT INTO Enrollments(enrollment_id, student_id,
course_id, enrollment_date)
                    VALUES (%s, %s, %s, %s)
                    '''
            para = (get_ids('enrollments', 'enrollment_id'), self.__student_id,
course.get_course_id(), date.today())
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Student Enrolled Successfully')
        except Exception as e:
            print(f'An error occurred {e}')

    def make_payment(self, amount, payment_date):
        try:
            my_cursor = self.connection.cursor()
            sql = '''INSERT INTO Payments(payment_id, student_id, amount,
payment_date)
            VALUES (%s, %s, %s, %s)
            '''
            para = (get_ids('payments', 'payment_id'), self.__student_id,
amount, payment_date)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Payment Made successfully')
        except Exception as e:
            print(f'An error occurred {e}')

    def get_enrolled_courses(self):
        my_cursor = self.connection.cursor()
        sql = '''
            SELECT * FROM Courses WHERE student_id = %s
        '''
        para = (self.__student_id,)
        my_cursor.execute(sql, para)
        t = list(my_cursor.fetchall())
        x = [list(i) for i in t]
        print(*x, sep='\n')

    def get_payment_history(self):
        my_cursor = self.connection.cursor()
        sql = '''
                    SELECT * FROM Payments WHERE student_id = %s
                '''
        para = (self.__student_id,)
        my_cursor.execute(sql, para)
        t = list(my_cursor.fetchall())
        x = [list(i) for i in t]
        print(*x, sep='\n')
    def display_student_info(self):
        print('StudentID', self.__student_id)
        print('Student First Name', self.__firstname)
        print('Student Last Name', self.__lastname)
        print('Student DOB', self.__dob)
        print('Student Email', self.__email)
        print('Student Phone Number', self.__phone_number)

    def update_student_info(self, first_name=None, last_name=None,
date_of_birth=None, email=None, phone_number=None):
        my_cursor = self.connection.cursor()

        if first_name:
            sql = '''
                UPDATE Students SET first_name = %s WHERE student_id = %s
            '''
            para = (first_name, self.__student_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__firstname = first_name

        if last_name:
            sql = '''
                UPDATE Students SET last_name = %s WHERE student_id = %s
            '''
            para = (last_name, self.__student_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__lastname = last_name

        if date_of_birth:
            sql = '''
                UPDATE Students SET date_of_birth = %s WHERE student_id = %s
            '''
            para = (date_of_birth, self.__student_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__dob = date_of_birth

        if email:
            sql = '''
                UPDATE Students SET email = %s WHERE student_id = %s
            '''
            para = (email, self.__student_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__email = email

        if phone_number:
            sql = '''
                UPDATE Students SET phone_number = %s WHERE student_id = %s
            '''
            para = (phone_number, self.__student_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            self.__phone_number = phone_number

        print('Student Details Updated Successfully')
```

Teacher.py

```
from db_connector.db_adapter import get_db_connection


class Teacher:

    def __init__(self, teacher_id, firstname, lastname, email):
        self.connection = get_db_connection()
        self.__teacher_id = teacher_id
        self.__firstname = firstname
        self.__lastname = lastname
        self.__email = email

    def get_teacher_id(self):
        return self.__teacher_id

    def get_first_name(self):
        return self.__firstname

    def get_last_name(self):
        return self.__lastname

    def get_email(self):
        return self.__email

    def display_teacher_info(self):
        print('TeacherID', self.__teacher_id)
        print('Teacher First Name', self.__firstname)
        print('Teacher Last Name', self.__lastname)
        print('Teacher Email', self.__email)

    def get_assigned_course(self):
        try:
            my_cursor = self.connection.cursor()
            sql = 'SELECT * FROM Course WHERE teacher_id = %s'
            para = (self.__teacher_id,)
            my_cursor.execute(sql, para)
            return list(my_cursor.fetchone())
        except Exception as e:
            print(f'An error occurred: {e}')

    def update_teacher_info(self, first_name=None, last_name=None, email=None):
        my_cursor = self.connection.cursor()

        try:
            sql1 = '''
                UPDATE Teacher SET first_name = %s WHERE teacher_id = %s
            '''
            para1 = (first_name, self.__teacher_id)
            my_cursor.execute(sql1, para1)
            self.connection.commit()
            self.__firstname = first_name
            print('First Name Updated Successfully')
        except Exception as e:
            print(f'An error occurred: {e}')

        try:
            sql2 = '''
                            UPDATE Teacher SET last_name = %s WHERE teacher_id = %s
                        '''
            para2 = (last_name, self.__teacher_id)
            my_cursor.execute(sql2, para2)
            self.connection.commit()
            self.__lastname = last_name
            print('Last Name Updated Successfully')
        except Exception as e:
            print(f'An error occurred: {e}')

        try:
            sql3 = '''
                            UPDATE Teacher SET email = %s WHERE teacher_id = %s
                        '''
            para3 = (email, self.__teacher_id)
            my_cursor.execute(sql3, para3)
            self.connection.commit()
            self.__email = email
            print('Email Updated Successfully')
        except Exception as e:
            print(f'An error occurred: {e}')
```

Sis_manager.py

```
from datetime import date
from db_connector.db_adapter import get_db_connection
from models.course  import Course
from models.payment import Payment
from models.student import Student
from models.enrollment import Enrollment
from models.teacher import Teacher
from db_connector.db_adapter import get_ids
from custom_exception.custom_exceptions import *

class SISManager:

    def __init__(self):
        self.connection = get_db_connection()

    def enroll_student_in_course(self, course: Course, student: Student):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                INSERT INTO Enrollments(enrollment_id, student_id, course_id,
enrollment_date)
                VALUES (%s, %s, %s, %s)
            '''
            para = (
            get_ids('enrollments', 'enrollment_id'), student.get_student_id(),
course.get_course_id(), date.today())
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Student Enrolled successfully')
        except Exception as e:
            print(f'An error occurred: {e}')

    def assign_teacher_to_course(self, teacher: Teacher, course: Course):
        try:
            course.assign_teacher(teacher)
        except Exception as e:
            print(f'An error occurred: {e}')

    def record_payment(self, student: Student, amount, payment_date):
        try:
            student.make_payment(amount, payment_date)
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_payment_report(self, student: Student):
        try:
            student.get_payment_history()
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_enrollment_report(self, course: Course):
```

```python
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT Students.* FROM Students JOIN Enrollments
                ON Students.student_id = Enrollments.student_id
                WHERE Enrollments.course_id = %s
            '''
            para = (course.get_course_id(),)
            # print(sql, para)
            my_cursor.execute(sql, para)
            x = [list(i) for i in list(my_cursor.fetchall())]
            print(*x)
        except Exception as e:
            print(f'An error occurred: {e}')

    def calculate_course_statistics(self, course: Course):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT SUM(Payments.amount) AS TotalAmount,
COUNT(Students.student_id) AS Enrollments FROM Students JOIN Enrollments
                ON Students.student_id = Enrollments.student_id
                JOIN Payments ON Payments.student_id = Students.student_id
                WHERE Enrollments.course_id = %s
            '''
            para = (course.get_course_id(),)
            my_cursor.execute(sql, para)
            x = [list(i) for i in list(my_cursor.fetchall())]
            print(*x, sep='\n')
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_student_by_id(self, student_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * FROM Students WHERE student_id = %s
            '''
            para = (student_id,)
            my_cursor.execute(sql, para)
            x = my_cursor.fetchone()
            if x is None:
                raise StudentNotFoundException('Invalid Student ID')
            else:
                return Student(*x)
        except StudentNotFoundException as snfe:
            print(f'An error occurred: ', snfe)
        except Exception as e:
            print(f'An error occurred: {e}')
```

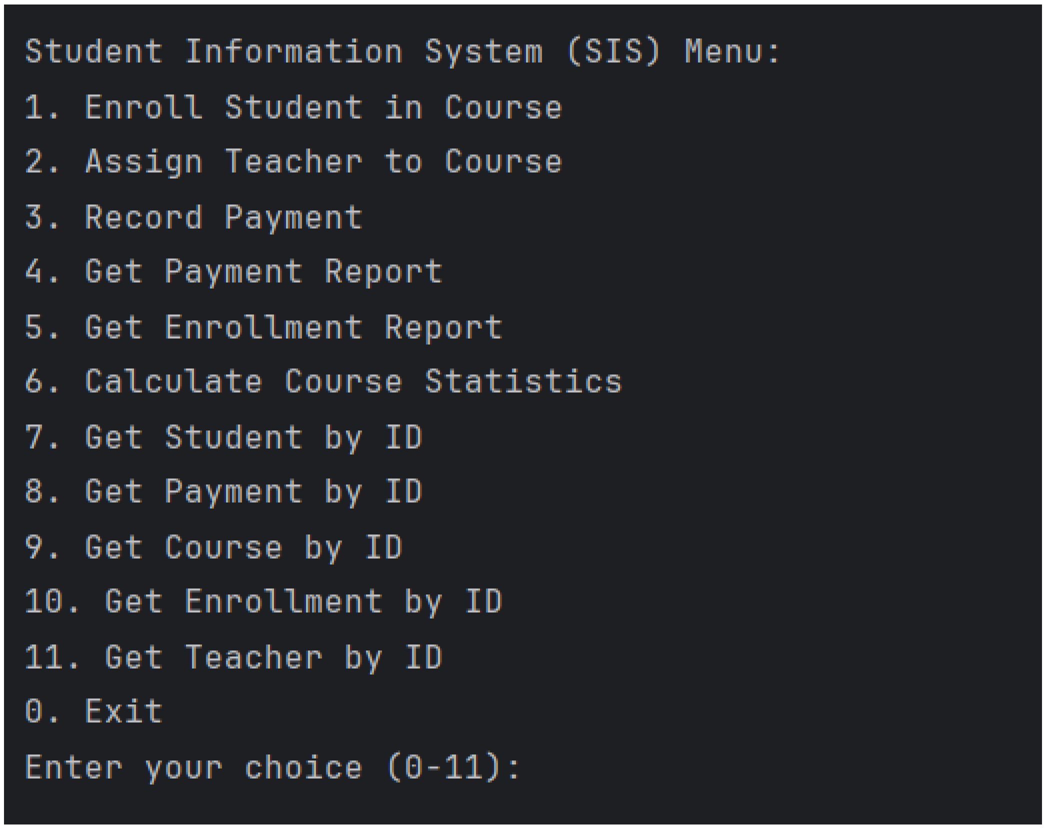
```python
def get_payment_by_id(self, payment_id):
    try:
        my_cursor = self.connection.cursor()
        sql = '''
        SELECT * FROM Payments WHERE payment_id = %s
        '''
        para = (payment_id,)
        my_cursor.execute(sql, para)
        x = my_cursor.fetchone()
        if x is None:
            raise PaymentValidationException('Invalid Payment ID')
        else:
            return Payment(*x)
    except PaymentValidationException as pve:
        print(f'An error occurred: ', pve)
    except Exception as e:
        print(f'An error occurred: {e}')

def get_course_by_id(self, course_id):
    try:
        my_cursor = self.connection.cursor()
        sql = '''
        SELECT * FROM Courses WHERE course_id = %s
        '''
        para = (course_id,)
        my_cursor.execute(sql, para)
        x = my_cursor.fetchone()
        if x is None:
            raise CourseNotFoundException('Invalid Course ID')
        else:
            return Course(*x)
    except CourseNotFoundException as cnfe:
        print(f'An error occurred: ', cnfe)
    except Exception as e:
        print(f'An error occurred: {e}')

def get_enrollment_by_id(self, enrollment_id):
    try:
        my_cursor = self.connection.cursor()
        sql = '''
        SELECT * FROM Enrollments WHERE enrollment_id = %s
        '''
        para = (enrollment_id,)
        my_cursor.execute(sql, para)
        x = my_cursor.fetchone()
        if x is None:
            raise InvalidEnrollmentDataException('Invalid Enrollment ID')
        else:
            return Enrollment(*x)
    except InvalidEnrollmentDataException as infe:
        print(f'An error occurred: ', infe)
    except Exception as e:
        print(f'An error occurred: {e}')

def get_teacher_by_id(self, teacher_id):
    try:
        my_cursor = self.connection.cursor()
        sql = '''
        SELECT * FROM Teacher WHERE teacher_id = %s
        '''
        para = (teacher_id,)
        my_cursor.execute(sql, para)
        x = my_cursor.fetchone()
        if x is None:
            raise TeacherNotFoundException('Invalid Teacher ID')
        else:
            return Teacher(*x)
    except TeacherNotFoundException as tnfe:
        print(f'An error occurred: ', tnfe)
    except Exception as e:
        print(f'An error occurred: {e}')
```

Main.py

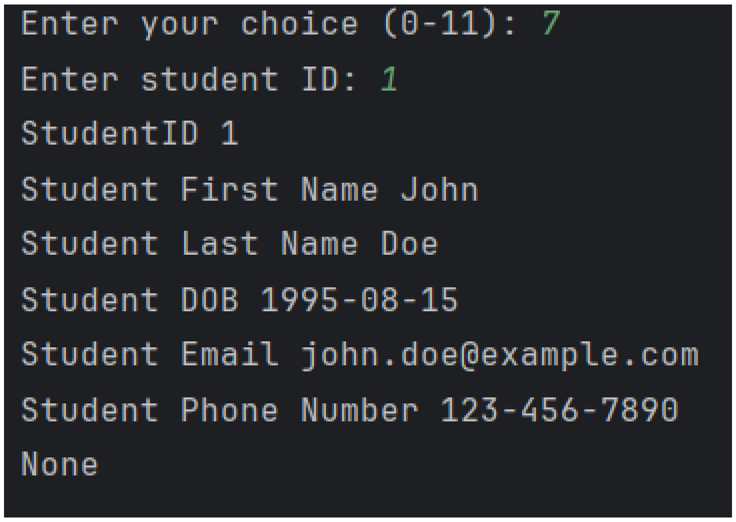
```python
from services.sis_manager import SISManager


def display_menu():
   print("\nStudent Information System (SIS) Menu:")
   print("1. Enroll Student in Course")
   print("2. Assign Teacher to Course")
   print("3. Record Payment")
   print("4. Get Payment Report")
   print("5. Get Enrollment Report")
   print("6. Calculate Course Statistics")
   print("7. Get Student by ID")
   print("8. Get Payment by ID")
   print("9. Get Course by ID")
   print("10. Get Enrollment by ID")
   print("11. Get Teacher by ID")
   print("0. Exit")

def get_user_input(prompt):
   return input(prompt).strip()
```

```python
def main():
    sis_manager = SISManager()

    while True:
        display_menu()

        choice = get_user_input("Enter your choice (0-11): ")

        if choice == "0":
            print("Exiting the Student Information System. Goodbye!")
            break
        elif choice == "1":
            # Enroll Student in Course
            student_id = get_user_input("Enter student ID: ")
            course_id = get_user_input("Enter course ID: ")
            student = sis_manager.get_student_by_id(student_id)
            course = sis_manager.get_course_by_id(course_id)
            sis_manager.enroll_student_in_course(course, student)
        elif choice == "2":
            # Assign Teacher to Course
            teacher_id = get_user_input("Enter teacher ID: ")
            course_id = get_user_input("Enter course ID: ")
            teacher = sis_manager.get_teacher_by_id(teacher_id)
            course = sis_manager.get_course_by_id(course_id)
            sis_manager.assign_teacher_to_course(teacher, course)
        elif choice == "3":
            # Record Payment
            student_id = get_user_input("Enter student ID: ")
            amount = float(get_user_input("Enter payment amount: "))
            payment_date = get_user_input("Enter payment date (YYYY-MM-DD): ")
            student = sis_manager.get_student_by_id(student_id)
            sis_manager.record_payment(student, amount, payment_date)
        elif choice == "4":
            # Get Payment Report
            student_id = get_user_input("Enter student ID: ")
            student = sis_manager.get_student_by_id(student_id)
            sis_manager.get_payment_report(student)
        elif choice == "5":
            # Get Enrollment Report
            course_id = get_user_input("Enter course ID: ")
            course = sis_manager.get_course_by_id(course_id)
            sis_manager.get_enrollment_report(course)
        elif choice == "6":
            # Calculate Course Statistics
            course_id = get_user_input("Enter course ID: ")
            course = sis_manager.get_course_by_id(course_id)
            sis_manager.calculate_course_statistics(course)
        elif choice == "7":
            # Get Student by ID
            student_id = get_user_input("Enter student ID: ")
            student = sis_manager.get_student_by_id(student_id)
            print(student.display_student_info())
        elif choice == "8":
            # Get Payment by ID
            payment_id = get_user_input("Enter payment ID: ")
            payment = sis_manager.get_payment_by_id(payment_id)
            print(payment.display_payment_info())
        elif choice == "9":
            # Get Course by ID
            course_id = get_user_input("Enter course ID: ")
            course = sis_manager.get_course_by_id(course_id)
            print(course.display_course_info())
        elif choice == "10":
            # Get Enrollment by ID
            enrollment_id = get_user_input("Enter enrollment ID: ")
            enrollment = sis_manager.get_enrollment_by_id(enrollment_id)
            print(enrollment.display_enrollment_info())
        elif choice == "11":
            # Get Teacher by ID
            teacher_id = get_user_input("Enter teacher ID: ")
            teacher = sis_manager.get_teacher_by_id(teacher_id)
            print(teacher.display_teacher_info())
        else:
            print("Invalid choice. Please enter a number between 0 and 11.")

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

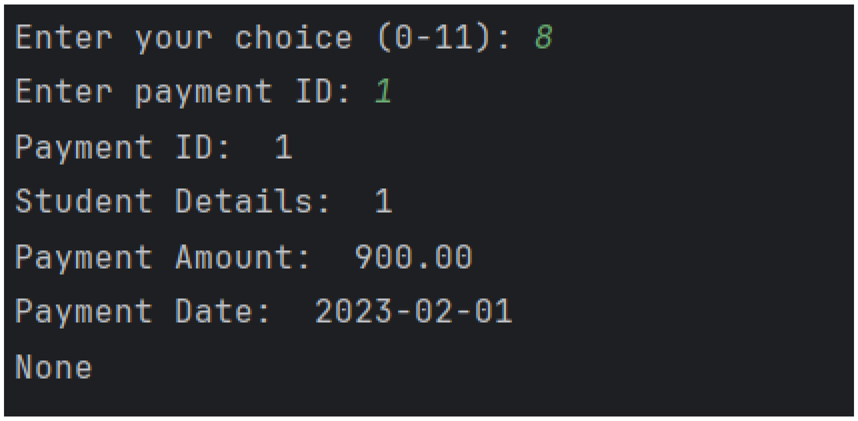
Output :



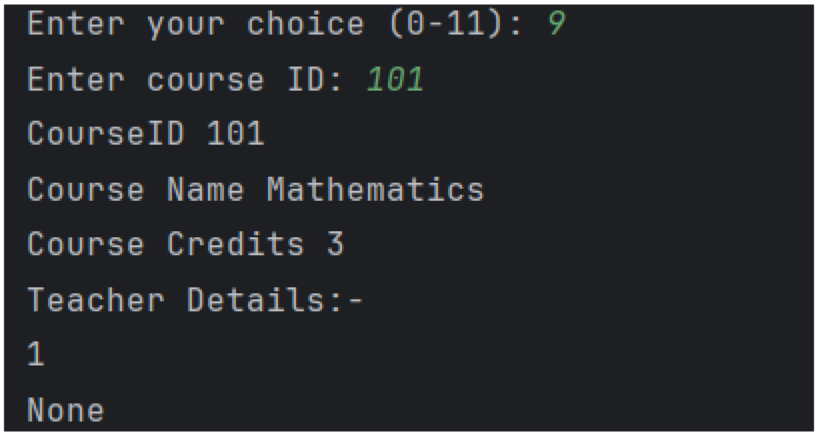
```
Student Information System (SIS) Menu:
1. Enroll Student in Course
2. Assign Teacher to Course
3. Record Payment
4. Get Payment Report
5. Get Enrollment Report
6. Calculate Course Statistics
7. Get Student by ID
8. Get Payment by ID
9. Get Course by ID
10. Get Enrollment by ID
11. Get Teacher by ID
0. Exit
Enter your choice (0-11):
```



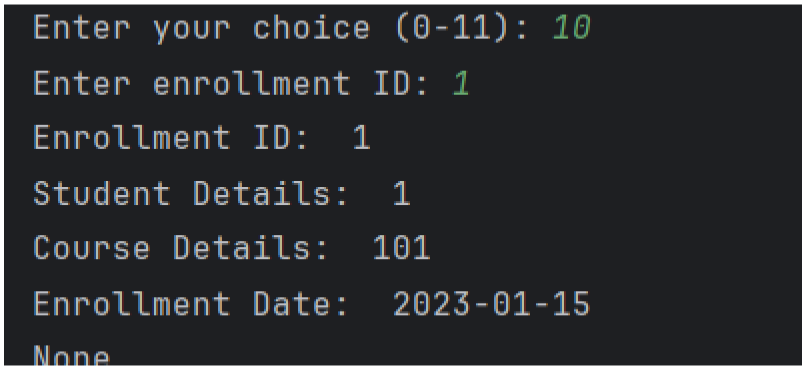
```
Enter your choice (0-11): 7
Enter student ID: 1
StudentID 1
Student First Name John
Student Last Name Doe
Student DOB 1995-08-15
Student Email john.doe@example.com
Student Phone Number 123-456-7890
None
```



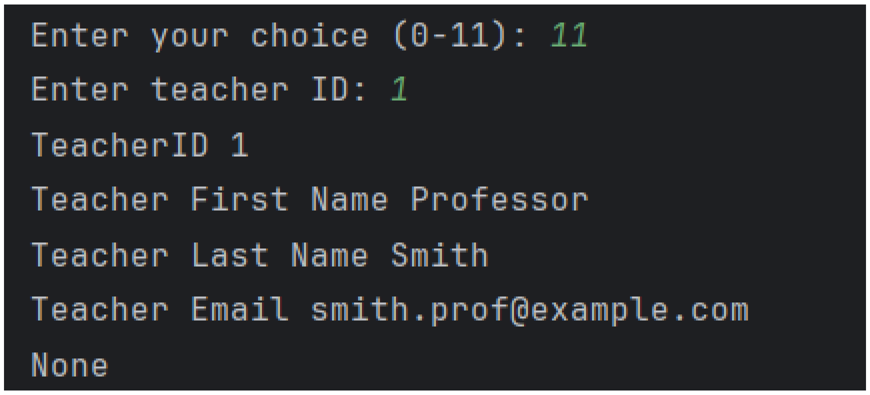
```
Enter your choice (0-11): 8
Enter payment ID: 1
Payment ID:  1
Student Details:  1
Payment Amount:  900.00
Payment Date:  2023-02-01
None
```



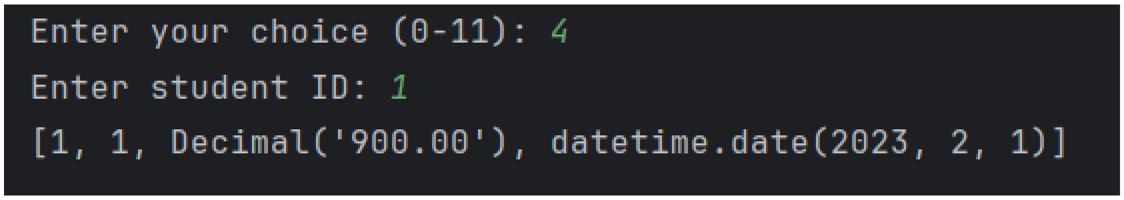
```
Enter your choice (0-11): 9
Enter course ID: 101
CourseID 101
Course Name Mathematics
Course Credits 3
Teacher Details:-
1
None
```



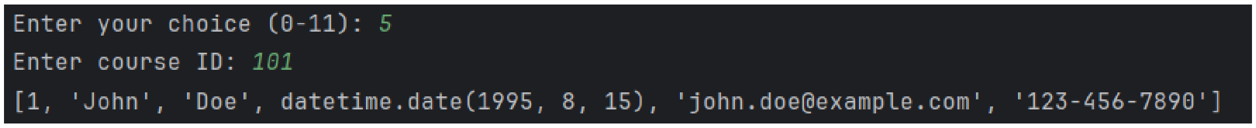
```
Enter your choice (0-11): 10
Enter enrollment ID: 1
Enrollment ID:  1
Student Details:  1
Course Details:  101
Enrollment Date:  2023-01-15
None
```



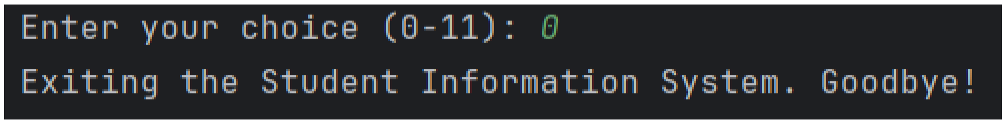
```
Enter your choice (0-11): 11
Enter teacher ID: 1
TeacherID 1
Teacher First Name Professor
Teacher Last Name Smith
Teacher Email smith.prof@example.com
None
```



```
Enter your choice (0-11): 4
Enter student ID: 1
[1, 1, Decimal('900.00'), datetime.date(2023, 2, 1)]
```



```
Enter your choice (0-11): 5
Enter course ID: 101
[1, 'John', 'Doe', datetime.date(1995, 8, 15), 'john.doe@example.com', '123-456-7890']
```



```
Enter your choice (0-11): 0
Exiting the Student Information System. Goodbye!
```