

project_main.py

```
1 import os
2 import csv
3
4 class Person:
5     def __init__(self, id, name, age, address) -> None:
6         self.id = id
7         self.name = name
8         self.age = age
9         self.address = address
10
11     data = {
12         "id": id,
13         "name": self.name,
14         "age": self.age,
15         "address": self.address
16     }
17
18     try:
19         # Check if file exists
20         file_exists = os.path.isfile("person.csv")
21
22         # Open the file in append mode and write data
23         with open("person.csv", "a", newline="") as file:
24             writer = csv.DictWriter(file, fieldnames=["id", "name", "age", "address"])
25
26             if not file_exists:
27                 writer.writeheader()
28
29             writer.writerow(data)
30
31     except Exception as e:
32         print(f"An unexpected error occurred: {e}")
33
34 @staticmethod
35 def display_person_info(id):
36     try:
37         # Initialize variables to avoid 'referenced before assignment' error
38         re_id = re_name = re_age = re_add = course_name = grade_num = None
39
40         # Read the person.csv file
41         with open("person.csv", "r") as pr:
42             person_reader = csv.DictReader(pr)
43
44             for p in person_reader:
45                 if p["id"].strip() == id.strip():
46                     re_id = p["id"]
47                     re_name = p["name"]
48                     re_age = p["age"]
```

```

49         re_add = p["address"]
50         break # Exit loop once the person is found
51
52     # Check if person data was found
53     if not re_id:
54         print("Person not found.")
55         return
56
57     # Read the course and grade files
58     with open("en_course.csv", "r") as enc, open("course_grade.csv", "r") as gpa:
59         course_reader = csv.DictReader(enc)
60         grade_reader = csv.DictReader(gpa)
61
62     # Find the enrolled course for the given ID
63     for c in course_reader:
64         if c["id"].strip() == id.strip():
65             course_name = c["course"]
66             break # Exit loop once the course is found
67
68     # Check if course was found
69     if not course_name:
70         print("Course not found.")
71         return
72
73     # Find the grade for the course and ID
74     for g in grade_reader:
75         if g["id"].strip() == id.strip() and g["course"].strip() == course_name:
76             grade_num = g["grade"]
77             break # Exit loop once the grade is found
78
79     # Display the information
80     print(f"""
81 Student Information:
82 Name: {re_name}
83 ID: {re_id}
84 Age: {re_age}
85 Address: {re_add}
86 Enrolled Course: {course_name}
87 Grade: {{{course_name}: {grade_num}}}
88 """)
89
90     except Exception as e:
91         print(f"An error occurred: {e}")
92
93
94 class Student():
95     # Class attributes to store grades and enrolled courses.
96     grade_list = {}
97     course_list = []
98

```

```

99 def add_grade(self, sid, course, grade):
100     try:
101         # Open the files containing student and course data.
102         with open("person.csv", "r") as pr, open("course.csv", "r") as cr:
103             person_reader = csv.DictReader(pr)
104             course_reader = csv.DictReader(cr)
105
106             # Find the student by ID and print a confirmation message.
107             for p in person_reader:
108                 if p["id"].strip() == sid.strip():
109                     print(f"Grade {grade} added for {p['name']} in ", end="")
110
111             # Find the course by code and complete the message.
112             for c in course_reader:
113                 if c["Course Code"].strip() == course.strip():
114                     print(f"{c['Course Name']}")
115
116                     # Add the grade to the class's grade list.
117                     key = c["Course Name"]
118                     value = grade
119                     g = {key: value}
120                     self.grade_list.update(g)
121
122             # Prepare data for saving in the course-grade file.
123             course_grade = {
124                 "id": sid,
125                 "course": key,
126                 "grade": grade
127             }
128
129             # Append to 'course_grade.csv', creating a header if needed.
130             file_exists = os.path.isfile("course_grade.csv")
131             with open("course_grade.csv", "a", newline="") as file:
132                 writer = csv.DictWriter(file, fieldnames=["id", "course", "grade"])
133                 if not file_exists:
134                     writer.writeheader() # Add headers if the file is new.
135                 writer.writerow(course_grade) # Add the new entry.
136
137     except Exception as e:
138         print(f"{e}") # Handle and print any exceptions that occur.
139
140 def enroll_course(self, sid, course):
141     try:
142         # Open the files containing student and course data.
143         with open("person.csv", "r") as pr, open("course.csv", "r") as cr:
144             person_reader = csv.DictReader(pr)
145             course_reader = csv.DictReader(cr)
146
147             # Find the student by ID and print a confirmation message.
148             for p in person_reader:

```

```

149         if p["id"].strip() == sid.strip():
150             print(f"Student {p['name']} (ID: {p['id']}) ", end="")
151
152     # Find the course by code and complete the message.
153     for c in course_reader:
154         if c["Course Code"].strip() == course.strip():
155             course = c["Course Name"]
156             print(f"enrolled in {course}")
157             self.course_list.append(course) # Store enrolled course.
158
159     # Prepare data for saving in the enrolled courses file.
160     en_course = {
161         "id": sid,
162         "course": course
163     }
164
165     # Append to 'en_course.csv', creating a header if needed.
166     file_exists = os.path.isfile("en_course.csv")
167     with open("en_course.csv", "a", newline="") as file:
168         writer = csv.DictWriter(file, fieldnames=["id", "course"])
169         if not file_exists:
170             writer.writeheader() # Add headers if the file is new.
171             writer.writerow(en_course) # Add the new entry.
172
173     except Exception as e:
174         print(f"Unexpected error: {e}") # Handle and print any exceptions.
175
176
177
178
179 class Course:
180     def __init__(self, course_name, course_code, course_instructor) -> None:
181         self.coursName = course_name
182         self.coursCode = course_code
183         self.coursInstructor = course_instructor
184         self.student = []
185
186         course_data = {
187             "Course Name": self.coursName,
188             "Course Code": self.coursCode,
189             "Course Instructor": self.coursInstructor
190         }
191
192     try:
193         # Check if file exists
194         file_exists = os.path.isfile("course.csv")
195
196         # Open the file in append mode and write data
197         with open("course.csv", "a", newline="") as file:

```

```

198         writer = csv.DictWriter(file, fieldnames=["Course Name", "Course Code", "Course
Instructor"])
199
200         if not file_exists:
201             writer.writeheader()
202
203             writer.writerow(course_data)
204
205     except Exception as e:
206         print(f"An unexpected error occurred: {e}")
207
208
209     def add_student():
210         pass
211
212     @staticmethod
213     def display_course_info(course_code):
214         try:
215             # Initialize variables to avoid 'referenced before assignment' error
216             cours_name = cours_code = cours_instructor = None
217
218             # Read the course.csv file
219             with open("course.csv", "r") as cr:
220                 course_read = csv.DictReader(cr)
221
222                 for c in course_read:
223                     if c["Course Code"].strip() == course_code.strip():
224                         cours_name = c["Course Name"]
225                         cours_code = c["Course Code"]
226                         cours_instructor = c["Course Instructor"]
227                         break # Exit loop once the person is found
228
229             # Check if person data was found
230             if not course_code:
231                 print("Course not found.")
232                 return
233
234             # Read the course and grade files
235             with open("en_course.csv", "r") as enc:
236                 course_reader = csv.DictReader(enc)
237
238             # Find the enrolled course for the given ID
239             for cr in course_reader:
240                 if cr["course"] == cours_name:
241                     student_id = cr["id"]
242                     break # Exit loop once the course is found
243
244             # Check if Student id was found
245             if not student_id:
246                 print("Student do not enrolled.")

```

```

247         return
248     with open("person.csv", "r") as pr:
249         reader = csv.DictReader(pr)
250
251         for row in reader:
252             if row["id"].strip() == student_id.strip():
253                 student_name = row["name"]
254
255             # Display the information
256             print(f"""
257 Course Information:
258 Course Name: {course_name}
259 Course Code: {course_code}
260 Instructor: {course_instructor}
261 Enrolled Student: {student_name}
262
263 """)
264
265         except Exception as e:
266             print(f"An error occurred: {e}")
267
268 def main():
269     print("""
270 ==== Student Management System ====
271 1. Add New Student
272 2. Add New Course
273 3. Enroll Student in Course
274 4. Add Grade for Student
275 5. Display Student Details
276 6. Display Course Details
277 7. Save Data from File
278 8. Load Data from File
279 0. Exit
280 """)
281
282     while True:
283         try:
284             option = int(input("Choose your option from above: "))
285             if option < 0 or option > 8:
286                 print("Please choose a valid option (0-8).\n")
287             elif option == 0:
288                 print("Exiting Student Management System. Goodbye!")
289                 break
290             elif option == 1:
291                 name = input("Enter Name: ")
292                 age = input("Enter Age: ")
293                 address = input("Enter Address: ")
294                 std_id = input("Enter Student ID: ")
295
296                 Person(std_id, name, age, address)

```

```

297         print(f"Student {name} (ID: {std_id}) added successfully.")
298         break
299     elif option == 2:
300         cname = input("Enter Course Name: ")
301         ccode = input("Enter Course Code: ")
302         cinstructor = input("Enter Instructor Name: ")
303
304         Course(cname, ccode, cinstructor)
305         print(f"Course {cname} (Code: {ccode}) created with instructor {cinstructor}")
306         break
307     elif option == 3:
308         std_id = input("Enter Student ID: ")
309         ccode = input("Enter Course Code: ")
310         student = Student()
311         student.enroll_course(std_id, ccode)
312         break
313     elif option == 4:
314         std_id = input("Enter Student ID: ")
315         ccode = input("Enter Course Code: ")
316         grade = input("Enter Grade: ")
317         student = Student()
318         student.add_grade(std_id, ccode, grade)
319         break
320     elif option == 5:
321         std_id = input("Enter Student ID: ")
322         Person.display_person_info(std_id)
323         break
324     elif option == 6:
325         ccode = input("Enter Course Code: ")
326         Course.display_course_info(ccode)
327         break
328
329     except ValueError:
330         print("Invalid input! Please enter a number.")
331
332 if __name__ == "__main__":
333     main()
334

```