```
import json
from pathlib import Path


class Student:
    """Represents a single student record."""
    def __init__(self, student_id: str, name: str, grade: float):
        self.id = student_id
        self.name = name
        self.grade = grade

    def to_dict(self):
        return {"id": self.id, "name": self.name, "grade": self.grade}

    @staticmethod
    def from_dict(data: dict):
        return Student(data["id"], data["name"], data["grade"])


class StudentManager:
    """Handles all student operations + JSON file persistence."""
    def __init__(self, filename="students.json"):
        self.filename = Path(filename)
        self.students = {}
        self.load()

    # -------------------- Persistence --------------------

    def load(self):
        if self.filename.exists():
            with open(self.filename, "r", encoding="utf-8") as f:
                data = json.load(f)
            for sid, sdata in data.items():
                self.students[sid] = Student.from_dict(sdata)

    def save(self):
        data = {sid: s.to_dict() for sid, s in self.students.items()}
        with open(self.filename, "w", encoding="utf-8") as f:
            json.dump(data, f, indent=2)

    # -------------------- CRUD Operations --------------------

    def add_student(self, student_id, name, grade):
```

```
        """Add a new student (unique ID required)."""
        if student_id in self.students:
            return False
        self.students[student_id] = Student(student_id, name, grade)
        self.save()
        return True

    def update_student(self, student_id, name=None, grade=None):
        if student_id not in self.students:
            return False
        student = self.students[student_id]
        if name:
            student.name = name
        if grade is not None:
            student.grade = grade
        self.save()
        return True

    def delete_student(self, student_id):
        if student_id not in self.students:
            return False
        del self.students[student_id]
        self.save()
        return True

    def list_students(self):
        """Return students in a formatted list (not print directly)."""
        return list(self.students.values())


# -------------------- CLI Demo --------------------

def print_table(students):
    print("\n" + "-"*45)
    print(f"{'ID':<10} {'Name':<20} {'Grade':<10}")
    print("-"*45)
    for s in students:
        print(f"{s.id:<10} {s.name:<20} {s.grade:<10}")
    print("-"*45)


if __name__ == "__main__":
    sm = StudentManager()
```

```
while True:
    print("\n===== Student Management System =====")
    print("1. Add Student")
    print("2. Update Student")
    print("3. Delete Student")
    print("4. List Students")
    print("5. Exit")

    choice = input("Enter choice: ").strip()

    if choice == "1":
        sid = input("Enter ID: ")
        name = input("Enter Name: ")
        grade = float(input("Enter Grade: "))
        if sm.add_student(sid, name, grade):
            print("Student added!")
        else:
            print("ID already exists!")

    elif choice == "2":
        sid = input("Enter ID to update: ")
        name = input("Enter new name (leave blank to keep same): ")
        grade_in = input("Enter new grade (leave blank to keep same): ")

        grade = float(grade_in) if grade_in else None

        if sm.update_student(sid, name if name else None, grade):
            print("Updated!")
        else:
            print("Student not found.")

    elif choice == "3":
        sid = input("Enter ID to delete: ")
        if sm.delete_student(sid):
            print("Deleted!")
        else:
            print("Student not found.")

    elif choice == "4":
        print_table(sm.list_students())

    elif choice == "5":
```

```
    print("Exiting...")
    break

else:
    print("Invalid choice. Try again.")
```