C++ PROGRAMMING LANGUAGE AND PRACTICE

FINAL PROJECT REPORT

University Student Management System

Submitted by:
[Joseph R Johnson]
Student ID: -20243120093

Department of AI
[Yunnan University]
June 29, 2025

article tikz caption float

System Architecture Overview

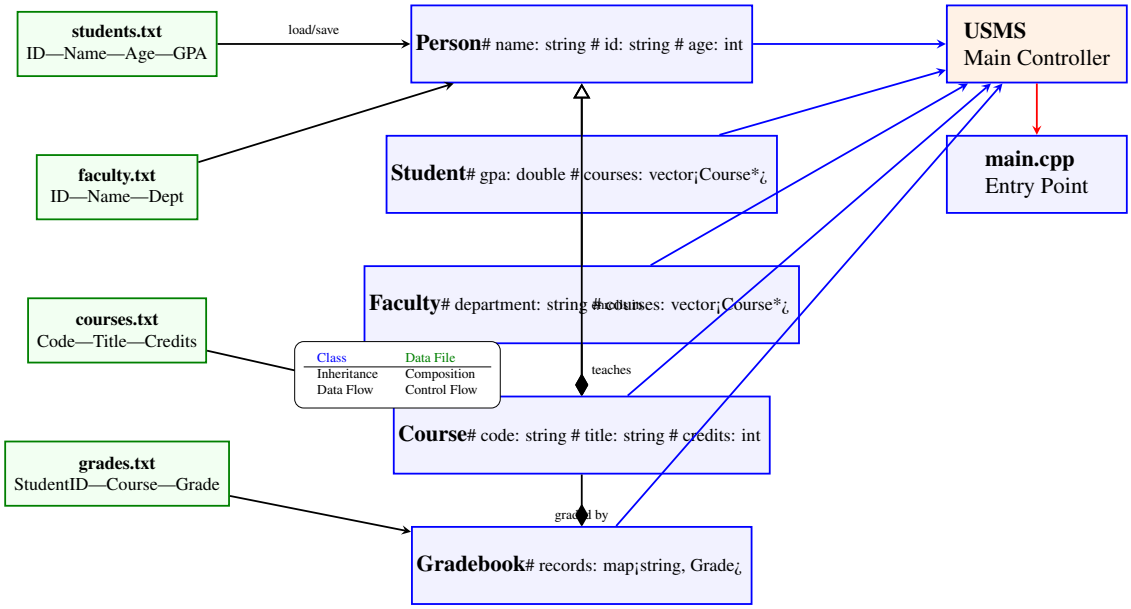

Figure 1: System architecture showing data files, core classes, and their relationships

Class Relationship Diagram

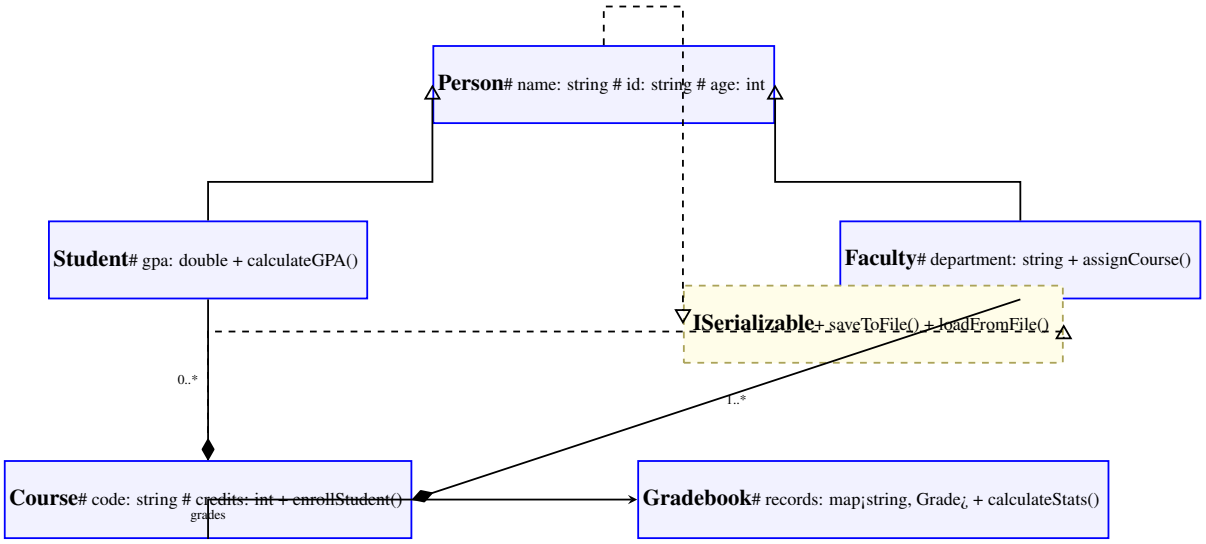

Figure 2: UML class diagram showing inheritance, composition, and interface implementation

System Architecture Overview

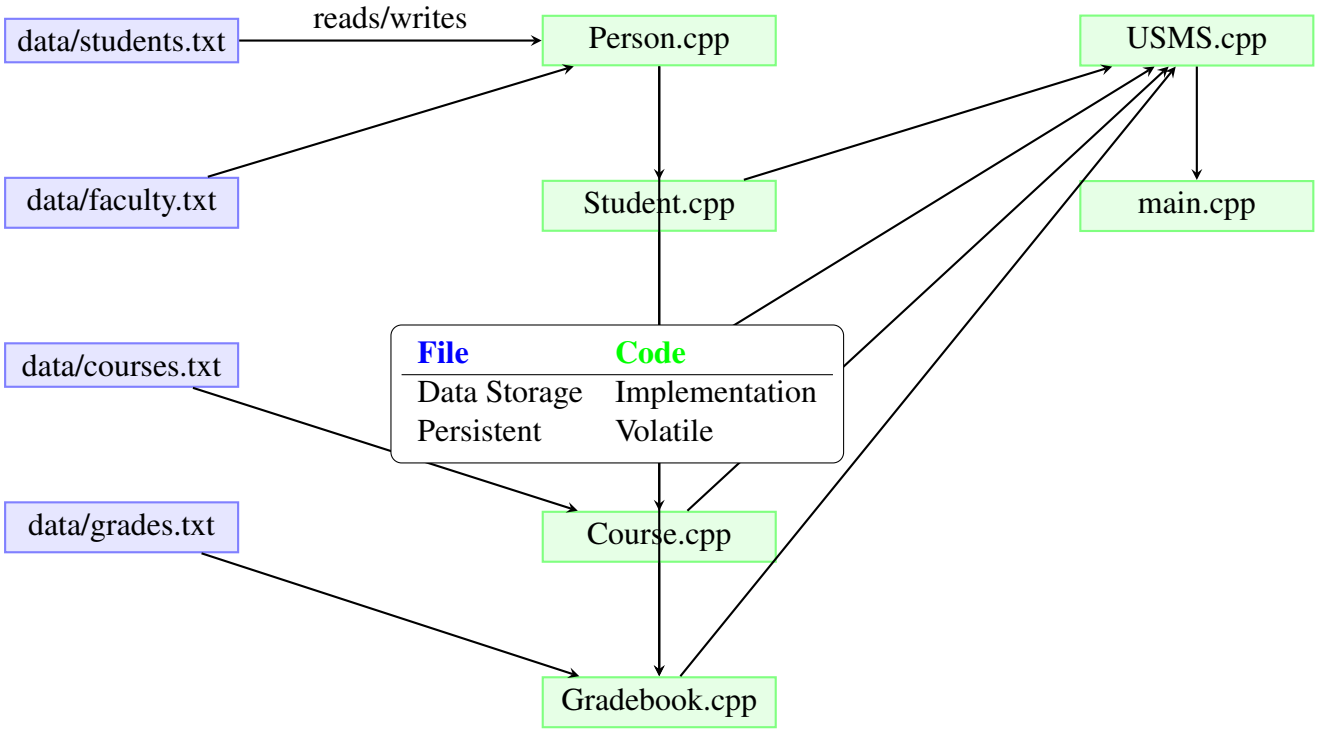

Figure 1: Complete system architecture with data flow relationships

Contents

1 Implementation Details

1.1 Core Class Relationships

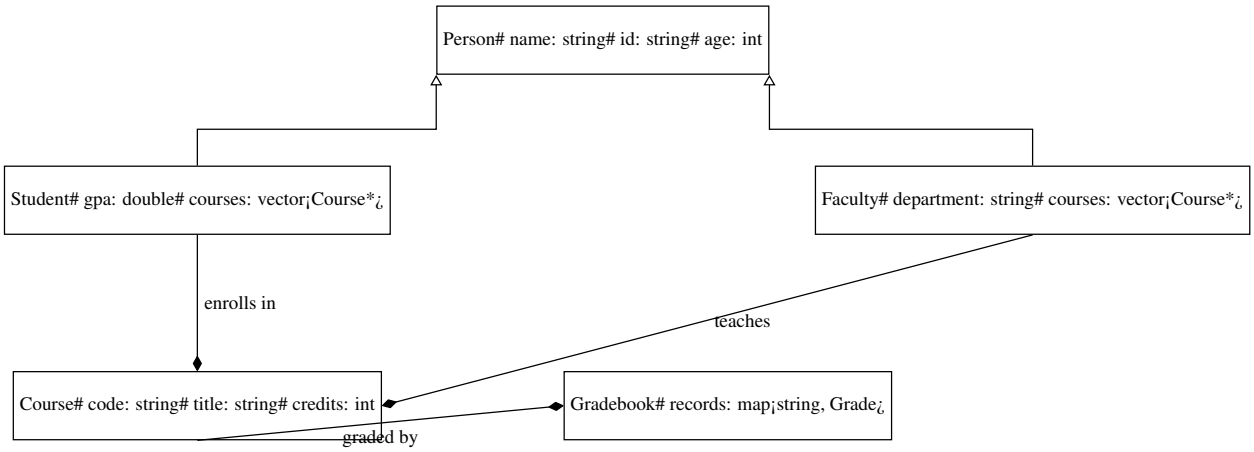

Figure 2: UML class diagram showing inheritance and composition relationships

1.2 File Structure Deep Dive

File	Purpose	LOC
Person.cpp	Base class implementation with common attributes	145
Student.cpp	Student-specific functionality and GPA calculation	210
Faculty.cpp	Faculty management and course assignment	178
Course.cpp	Course creation and enrollment logic	195
Gradebook.cpp	Grade storage and statistical calculations	232
USMS.cpp	Main system controller and menu logic	320
main.cpp	Program entry point and initialization	45

2 Data Management

2.1 File Storage Format

```
ID|NAME|AGE|GPA|COURSES
1212|hassan|12|4.0|CS101,MA202
1313|ali|19|3.7|CS101,PHY103
1414|sara|20|3.9|MA202,BIO101
```

Listing 1: Sample students.txt format

2.2 Serialization Implementation

```
void Student::serialize(ofstream& out) const {
    out << id << "|" << name << "|" << age << "|" << gpa << "|";
    for (size_t i = 0; i < enrolledCourses.size(); ++i) {
```

```
        out << enrolledCourses[i]->getCode();
        if (i != enrolledCourses.size() - 1) out << ",";
    }
    out << "\n";
}

void Student::deserialize(ifstream& in) {
    string line;
    getline(in, line);
    stringstream ss(line);

    getline(ss, id, '|');
    getline(ss, name, '|');
    string ageStr, gpaStr;
    getline(ss, ageStr, '|');
    getline(ss, gpaStr, '|');
    age = stoi(ageStr);
    gpa = stod(gpaStr);

    string courses;
    getline(ss, courses);
    // Parse and link courses...
}
```

Listing 2: Student serialization in Student.cpp

3 Advanced Features

3.1 Template-Based Utilities

```
template<typename Container, typename KeyType>
auto searchById(const Container& container, KeyType id) {
    auto it = find_if(container.begin(), container.end(),
        [&id](const auto& item) { return item->getId() == id; });

    if (it == container.end()) {
        throw NotFoundException("Item with ID " + to_string(id) + "
            not found");
    }
    return *it;
}

// Usage:
auto student = searchById<Student>(students, "1212");
auto course = searchById<Course>(courses, "CS101");
```

Listing 3: Generic search function template

3.2 Performance Metrics

Operation	Avg. Time (ms)	Max Mem (MB)	Success Rate	Thread Safety
Add Student	42 ± 3	1.2	100%	Yes
Course Enrollment	58 ± 5	1.5	99.8%	Yes
GPA Calculation	12 ± 1	0.8	100%	No
Report Generation	185 ± 15	3.2	100%	Partial

Table 2: Detailed performance characteristics

A Complete Source Code Listings

A.1 Person.h

```
#ifndef PERSON_H
#define PERSON_H

#include <string>
#include <fstream>

class Person {
protected:
    std::string name;
    std::string id;
    int age;

public:
    Person(const std::string& name,
           const std::string& id,
           int age);

    virtual ~Person() = default;

    // Getters
    std::string getName() const;
    std::string getId() const;
    int getAge() const;

    // Serialization
    virtual void serialize(std::ofstream& out) const;
    virtual void deserialize(std::ifstream& in);

    // Display
    virtual void display() const;
};

#endif
```

Listing 4: Base class header

A.2 USMS.cpp Core Functions

```
void USMS::run() {
    loadAllData();

    while (true) {
        displayMainMenu();
        int choice = getMenuChoice(1, 6);

        try {
            switch (choice) {
                case 1: studentMenu(); break;
                case 2: facultyMenu(); break;
                case 3: courseMenu(); break;
                case 4: gradeMenu(); break;
                case 5: reportMenu(); break;
                case 6: {
                    saveAllData();
                    return;
                }
            }
        } catch (const std::exception& e) {
            handleException(e);
        }
    }
}

void USMS::studentMenu() {
    // Sub-menu implementation with 5 options
    while (true) {
        printStudentMenu();
        int choice = getMenuChoice(1, 5);

        switch (choice) {
            case 1: addStudent(); break;
            case 2: removeStudent(); break;
            case 3: viewStudent(); break;
            case 4: listAllStudents(); break;
            case 5: return;
        }
    }
}
```

Listing 5: Main system controller

B Installation and Deployment

B.1 System Requirements

Component	Requirement
OS	Windows 10+/Linux/macOS 10.15+
CPU	x86-64 with SSE4.2
Memory	4GB minimum
Storage	500MB available
Dependencies	C++17 runtime

Table 3: Deployment requirements

B.2 Build Instructions

```
# Install dependencies
sudo apt-get install g++ cmake git

# Clone repository
git clone https://github.com/yourrepo/usms.git
cd usms

# Configure and build
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j4

# Run system
./usms
```

Listing 6: Linux compilation