Peer Study Matching System

Final Project Report

<u>Student Name</u>
Sundar Raj Sharma, Y00889901

<u>Course Title</u>
Principles of Computer Programming, CSCI.6901.43180.2025.Fall

<u>Course Faculty</u>
Rachel Gramann

<u>Institution</u>
Youngstown State University

<u>Submission Date</u>
Dec, 05, 2025

1. BUSINESS IDEA OVERVIEW

The Peer Study Matching System addresses a common challenge in academic settings: finding compatible study partners. Students often struggle to form effective study groups due to mismatched learning styles, incompatible schedules, or complementary skill gaps. This system automates the matching process by analyzing student profiles, learning preferences, and availability to create optimized study groups.

The program solves this problem by implementing an intelligent matching algorithm that considers multiple factors including learning styles, academic strengths and weaknesses, availability, and compatibility scores. Students can enroll, manage their profiles, and be automatically grouped with peers who complement their learning needs.

2. KEY CLASSES AND OBJECT-ORIENTED DESIGN

LearningProfile Class

The LearningProfile class encapsulates a student's learning characteristics and preferences. It manages learning style preferences, academic strengths and weaknesses, subject interests, availability schedules, and study goals. The class includes a sophisticated compatibility calculation algorithm that scores potential matches based on complementary strengths/weaknesses (60 points), matching availability (25 points), and similar learning styles (15 points).

Student Class

The Student class serves as the primary entity representing each enrolled student. It contains student identification (ID, name, email), academic information (major, year), enrollment metadata (date, active status), an embedded LearningProfile object demonstrating object composition, and group assignment tracking. The class provides comprehensive display methods for both detailed profiles and summary listings, and implements compatibility scoring by delegating to the LearningProfile class.

StudyGroup Class

The StudyGroup class manages collections of matched students. It maintains group identification and metadata, a dynamic member list using vectors, meeting schedules and subject focus, maximum member capacity (default 4 students), and active/inactive status tracking. The class provides methods for adding members with capacity checks, verifying membership, and displaying group information.

JSONHandler Class

The JSONHandler class implements data persistence using JSON file storage. It provides static methods for file operations, serialization/deserialization of Student and StudyGroup objects, CRUD operations (Create, Read, Update, Delete), and file existence checking. This class separates data management concerns from business logic.

3. KEY FUNCTIONS AND ALGORITHMS

Compatibility Algorithm

The calculateCompatibility() function implements a weighted scoring system that identifies complementary learning partnerships. It awards up to 60 points for matching a student's strengths with another's weaknesses, up to 25 points for overlapping availability, and 15 points for identical learning styles. This algorithm ensures balanced study groups where members can help each other.

Group Generation Algorithm

The generateGroups() function implements an intelligent matching process. It filters active students without group assignments, iterates through unassigned students to create new groups, attempts to add compatible partners (compatibility score > 30), fills incomplete groups with remaining students to reach minimum size, and updates both student and group records in persistent storage.

Data Persistence Functions

The JSONHandler class implements comprehensive data management through saveAllStudents() and saveAllGroups() for batch serialization, loadAllStudents() and loadAllGroups() for deserialization, updateStudent() for modifying existing records, and deleteStudent() for soft deletion. These functions use the nlohmann JSON library for reliable parsing and generation.

4. TECHNICAL IMPLEMENTATION DETAILS

Dynamic Memory Management

The program extensively uses std::vector for dynamic array management, avoiding manual memory allocation and reducing memory leak risks. Vectors automatically resize for student lists, group members, profile attributes, and availability schedules.

User Interface Design

The console-based interface provides a clean menu system with numbered options for all operations, formatted display output using visual separators and aligned text, input validation and error handling, and confirmation prompts for destructive operations.

Date and Time Management

The getCurrentDate() utility function uses the C++ time library to generate ISO-formatted timestamps for enrollment dates and group creation. This ensures consistent date formatting across the system.

5. CHALLENGES AND SOLUTIONS

Challenge 1: Designing the Compatibility Algorithm

The initial challenge was determining appropriate weights for different matching criteria. I implemented a 60-25-15 split (complementary skills, availability, learning style) based on educational research suggesting that complementary strengths are the most important factor for successful peer learning.

Challenge 2: JSON Integration and Data Persistence

Implementing reliable JSON serialization required careful handling of nested objects and vector serialization. I solved this by implementing toJSON() and fromJSON() methods in each class, using the nlohmann JSON library for robust parsing, and including error handling with try-catch blocks.

Challenge 3: Input Validation and Buffer Management

Mixed use of cin and getline() caused input buffer issues. I implemented the clearInputBuffer() utility function to consistently clear the input stream after cin operations, preventing skipped inputs.

Challenge 4: Group Generation Logic

Balancing optimal matching with practical constraints was complex. The algorithm first tries to match highly compatible students (score > 30), then fills groups to minimum viable size. This two-phase approach ensures no student is left unmatched while still prioritizing compatibility.

6. FUTURE ENHANCEMENTS

With additional development time, I would implement:

• Advanced Matching Algorithm: Implement machine learning to improve compatibility predictions based on successful group outcomes and feedback.

• Real-time Messaging System: Add in-app communication between group members using socket programming for coordinating study sessions.

• Web Interface: Develop a modern web application using React frontend and RESTful API backend to replace the console interface.

• Calendar Integration: Connect with Google Calendar or Outlook to automatically schedule group meetings and send reminders.

• Performance Analytics: Track group success metrics, study session attendance, and academic performance improvements to validate matching effectiveness.

• Database Migration: Replace JSON file storage with a proper database system (PostgreSQL or MongoDB) for better scalability and concurrent access.

• Mobile Application: Develop iOS and Android apps for on-the-go access and push notifications about group activities.

• Group Management Features: Allow students to request group changes, rate study sessions, and provide peer feedback for continuous improvement.

7. CONCLUSION

The Peer Study Matching System successfully demonstrates core C++ programming concepts including class design, object-oriented principles, dynamic memory management with vectors, file I/O operations through JSON persistence, and user interaction. The program addresses a real-world educational need by automating the complex task of forming effective study groups.

The implementation showcases proper software engineering practices with separation of concerns through the JSONHandler class, robust error handling and input validation, clear code organization and documentation, and scalable architecture for future enhancements. The compatibility algorithm provides intelligent matching that considers multiple factors to create balanced, effective study groups.

This project has strengthened my understanding of C++ programming fundamentals while creating a practical application with real potential for educational settings. The modular design and clean architecture provide a solid foundation for the planned future enhancements.