

**Gamify fyfyfyfyfyf gagagagfyfyfy~~~**

**Project Instructions**

- Project Due Dates:

-  **Phase-1 Due Date: Nov 15, 2024**

- Phase-2 Due Date: Dec 01, 2024

-Overview

- Key features include:

🥕Problems **read from a CSV/Text file**, allowing teachers to add new problem statements without code updates. Students can select different games and levels.

 🥕Support for **single-player and double-player games.**

🥕A **point system for multiple students** to track their progress.

Instructions:

- Clear code with:

🥕**Exception handling** to avoid crashes or incorrect data entries.

🥕Use a 3-file structure.

 🥕 Make assumptions only to improve the user experience without removing required features.

- Important Libraries:

1. Console Text Color:

`#include <windows.h>`, use `SetConsoleTextAttribute`.

2. Clear Console Screen:

- Windows: `system("CLS")`.

3. Add Delay:

- Windows: `#include <windows.h>`, `Sleep(milliseconds)`

4. Cross-Platform:

`#include<chrono>` `#include<thread>`std::this\_thread::sleep\_for(std::chrono::milliseconds(milliseconds))`.

5. UI Graphics:

- Windows: `#include <conio.h>`

---

**Phase-1 Requirements**

1. Add Teacher and Student Records

- Ability to add teachers and students.

- Assign multiple students to a teacher.

2. Create a Game: "Memory Game - Flip Cards for Matching"

- Rules:

- Set of 10 problems, 20 cards (10 cards with questions, 10 with answers).

- Randomly shuffled, numbered 1-20.

- Single-player game.

- Player selects a problem set and views flipped cards.

- Prompts for two card numbers, reveals selected cards.

- Matched pairs disappear; unmatched pairs flip back.

- The game ends when all pairs are matched.

- Scoring:

- 10 points for completion in <10 mins.

- 5 points for 10-20 mins.

- 2 points for >20 mins.

---

Weekly Work Distribution

**Day 1: Project Setup and Initial Planning**

**- Person A:**

- Set Up Directory and Classes : Create files for `main.cpp`, `Teacher.h/.cpp`, `Student.h/.cpp`, and `Game.h/.cpp`.

- Define Base Classes :

- Teacher class with attributes for teacher ID, name, and list of assigned students.

- Student class with attributes for student ID, name, and points.

- File Structure : Outline a basic CSV structure for teacher and student records.

**- Person B:**

- Menu Design : Create a `Menu` class to add teacher and student records.

- Function to Add Teachers : Begin coding a method to add teacher records.

- Game Planning : Outline the `MemoryGame` class structure, with attributes for card pairs and game methods.

**Day 2: Implement Teacher and Student Record System**

**- Person A:**

- Teacher Functions : Implement adding teachers to a CSV file, allowing easy registration of new teachers.

- Assign Students to Teacher : Add a method to assign multiple students to a teacher using student IDs.

- Reading Teacher Data : Start creating a method to read teacher data from CSV.

**- Person B:**

- Student Record Functions : Add methods to `Student.cpp` for adding and retrieving student records.

- Assign Students in Menu : Create a function in `Menu` for assigning students to teachers.

- Testing : Test adding records with sample data.

Day 3: Memory Game Design and Setup

**- Person A:**

- Base Structure in Game : Implement `MemoryGame` setup functions in `Game.cpp`.

- Load problem set from CSV for Math/Programming.

- Initialize and shuffle cards with questions and answers.

- Display Flipped Cards : Write a method to display cards by showing only card numbers.

**- Person B:**

- Game Start : Complete functions for the player to select a problem set and initialize the game.

- Shuffle and Position Cards : Ensure cards are shuffled and positions are mapped.

- Error Handling : Add exception handling for invalid game setup inputs.

**Day 4: Core Game Logic Implementation**

**- Person A:**

- Game Loop Logic : Write the main game loop in `MemoryGame`.

- Handle player card selection, card matching logic, and disappear cards when matched.

- Update the board as cards are flipped/matched.

**- Person B:**

- Scoring System : Track time using `std::chrono` and calculate points.

- Save Points : Update student record with points after each game.

**Day 5: Testing and Exception Handling**

**- Person A:**

- Game Loop Testing : Test different game scenarios, ensuring successful and unsuccessful matches work as expected.

- Final Check : Ensure all cards disappear after matching.

**- Person B:**

- Record System Testing : Test teacher and student record creation and error handling.

- Point System Testing : Verify scoring based on time.

- File Operations : Handle missing or corrupt files gracefully.

**Day 6: Code Review and Refinement**

**- Both :**

- Code Review : Verify naming conventions, indentation, modularity, and clear class structures.

- Function Comments : Refine comments at the start of each function.

- File Structure Check : Ensure proper 3-file structure with header, source, and main files.

**Day 7: Final Testing and Documentation**

**- Person A :**

- Full Testing : Conduct comprehensive testing of all functionalities and address any bugs.

- Bug Fixes : Document issues and improvements.

- **Person B :**

- Documentation : Create a README with setup instructions and usage guide.

- Submission Prep : Zip the project files and name the zip according to instructions (`23F-0005\_23F-0008\_Lab\_Project\_Phase\_1.zip`).



🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖 🥕💖