Lingua Cards-language Flashcard System Presentation

Created by: Lidya Ghede

Date: 07/23/2025

Overview
What is Lingua Cards?

- Lingua Cards is a console-based flashcard app to help study language vocabulary through self-paced learning. It bui;d in C++ and use a text file to store flash cards persistently between sessions.

User can:

- Add new flash cards (word + translation)
- Review Flash cards by testing their memory of translations. (if it answer correctly, it updates their progress)
- View progress stats like total numbers, how many have been “ mastered “ (answer correctly 3+ times)

Why it works: based on the learning principle that repeated exposure spaced over time improves memory retention. The system encourages reviewing cards just before you are likely to forgot them.

Program Data & Key Feature

01

- Front: the words to remember
- Back: its translation the words
- Correct Count: how many user got it right
- Last review: last time it was reviewed.

02

- This code explain that the internal structure and logic of the Lingua Cards Flash cards program.
- I use a class to organize the logic and data of the flash cards system. This keeps the code clean an modular.

```
// Flashcard structure representing each flashcard's data
struct Flashcard {
    std::string front;      // The word/question
    std::string back;       // The translation/answer
    int correctCount;       // Number of times answered correctly
    time_t lastReview;      // Last review timestamp
};
```

```
// LinguaCards class: handles flashcard management and review
class LinguaCards {
private:
    std::vector<Flashcard> cards;   // List of flashcards
    const std::string filename = "flashcards.txt";    // File to store cards
```

```
    void loadCards() {
        std::ifstream file(filename);
        if (!file.is_open()) return;    // if file doesn't exist, skip

        std::string front, back;
        int count;
        time_t last;
    // Read each card's data: front, back, correctCount, lastReview
        while (std::getline(file, front) && std::getline(file, back) && file >> count >> last) {
            file.ignore(); // Clear newline
            cards.push_back({front, back, count, last});
        }
        file.close();
    }
// Saves current flashcards to file pon exit or update
    void saveCards() {
        std::ofstream file(filename);
        for (const auto& card : cards) {
            file << card.front << "\n"
                 << card.back << "\n"
                 << card.correctCount << " "
                 << card.lastReview << "\n";
        }
        file.close();
    }
```

Saving and Loading Flash cards.

- To keep the flashCards between program runs, we read and write them to a file.
- The loadCards: this function opens the file, read data line by line and loads flashcards into memory .
- The SaveCrads: the function write all flashcards data to the file so progress is saved.
- The AddCard:: method let user add new words and their translations. It add the card to memory, sets review count to 0 and current time, saves the new card to file.

```
// Add a new flashcard to the list
    void addCard(const std::string& front, const std::string& back) {
        cards.push_back({front, back, 0, time(nullptr)});
        saveCards();
    }
```

On this feature it choose cards that the user needs to review.

- It selects cards that: Are not mastered yer (less than 3 correct answer) or have not been reviewed in 24 hours.
- It also asks user to type the answer. If correct, increase Correct Count, updates times. If wrong, reduces Correct Count also update review times. After reviewing all cards, saves progress.

```
// Review flashcrads using spaced repetition logic
    void reviewCards() {
        if (cards.empty()) {
            std::cout << "No flashcards available. Add some first!\n";
            return;
        }

        std::vector<Flashcard> dueCards;
        time_t now = time(nullptr);
        // Select cards that are either not mastered or due after 1 day
        for (const auto& card : cards) {
            // Simple spaced repetition: review if not answered correctly 3 times or 1 day has passed
            if (card.correctCount < 3 || difftime(now, card.lastReview) >= 86400) {
                dueCards.push_back(card);
            }
        }

        if (dueCards.empty()) {
            std::cout << "No cards due for review!\n";
            return;
        }
       // Shuffle cards randomly for varied practice
        std::random_shuffle(dueCards.begin(), dueCards.end());
        // Review each due card
        for (auto& card : dueCards) {
            std::cout << "Front: " << card.front << "\nEnter answer: ";
            std::string answer;
            std::getline(std::cin, answer);
```

```
            if (answer == card.back) {
                std::cout << "Correct!\n";
                // Update the original card in the list
                for (auto& c : cards) {
                    if (c.front == card.front) {
                        c.correctCount++;
                        c.lastReview = time(nullptr);
                    }
                }
            } else {
                std::cout << "Incorrect. Answer: " << card.back << "\n";
                // Penalize and update last review time
                for (auto& c : cards) {
                    if (c.front == card.front) {
                        c.correctCount = std::max(0, c.correctCount - 1);
                        c.lastReview = time(nullptr);
                    }
                }
            }
        }
        saveCards();   // Save updates after reviewing
    }
```

Progress Statistics

This method shows the total and mastered cards.

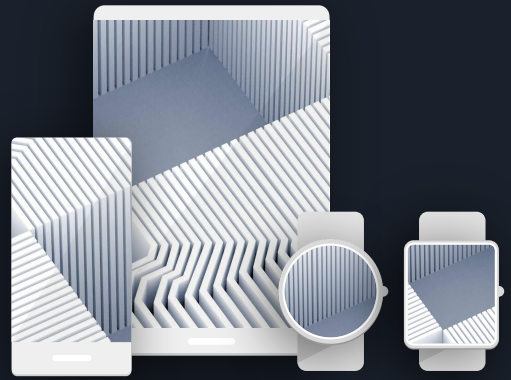
- A mastered card is one the user got correct 3 times.
- It helps user see progress and improvement.

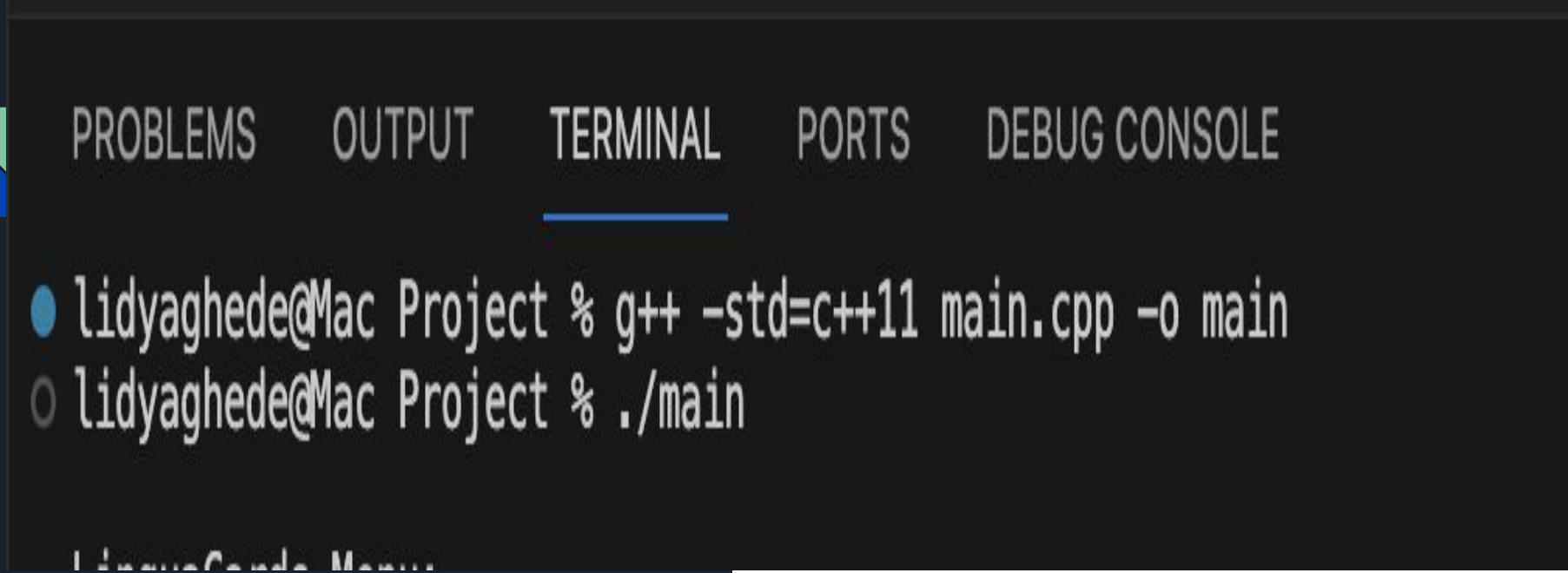
```
/ Display flashcard statistics to the user
    void showStats() {
        if (cards.empty()) {
            std::cout << "No cards available.\n";
            return;
        }

        int total = cards.size();
        int mastered = 0;
        // Count cards mastered (answer correctly 3+ times)
        for (const auto& card : cards) {
            if (card.correctCount >= 3) mastered++;
        }
        std::cout << "Total cards: " << total << "\n";
        std::cout << "Mastered cards: " << mastered << " (" << (mastered * 100 / total) << "%)\n";
    }
;
```

```
/ Main function: entry point for the application
nt main() {
    LinguaCards app;
    std::string choice;
    // Menu loop
    while (true) {
        std::cout << "\nLinguaCards Menu:\n"
                  << "1. Add flashcard\n"
                  << "2. Review flashcards\n"
                  << "3. Show stats\n"
                  << "4. Exit\n"
                  << "Choose an option: ";
        std::getline(std::cin, choice);

        if (choice == "1") {
            std::string front, back;
            std::cout << "Enter front (word): ";
            std::getline(std::cin, front);
            std::cout << "Enter back (translation): ";
            std::getline(std::cin, back);
            app.addCard(front, back);
            std::cout << "Card added!\n";
        } else if (choice == "2") {
            app.reviewCards();
        } else if (choice == "3") {
            app.showStats();
        } else if (choice == "4") {
            break; // Exist the program
        } else {
            std::cout << "Invalid option. Try again.\n";
        }
    }

    return 0;
```

Main function - THE APP RUNNER

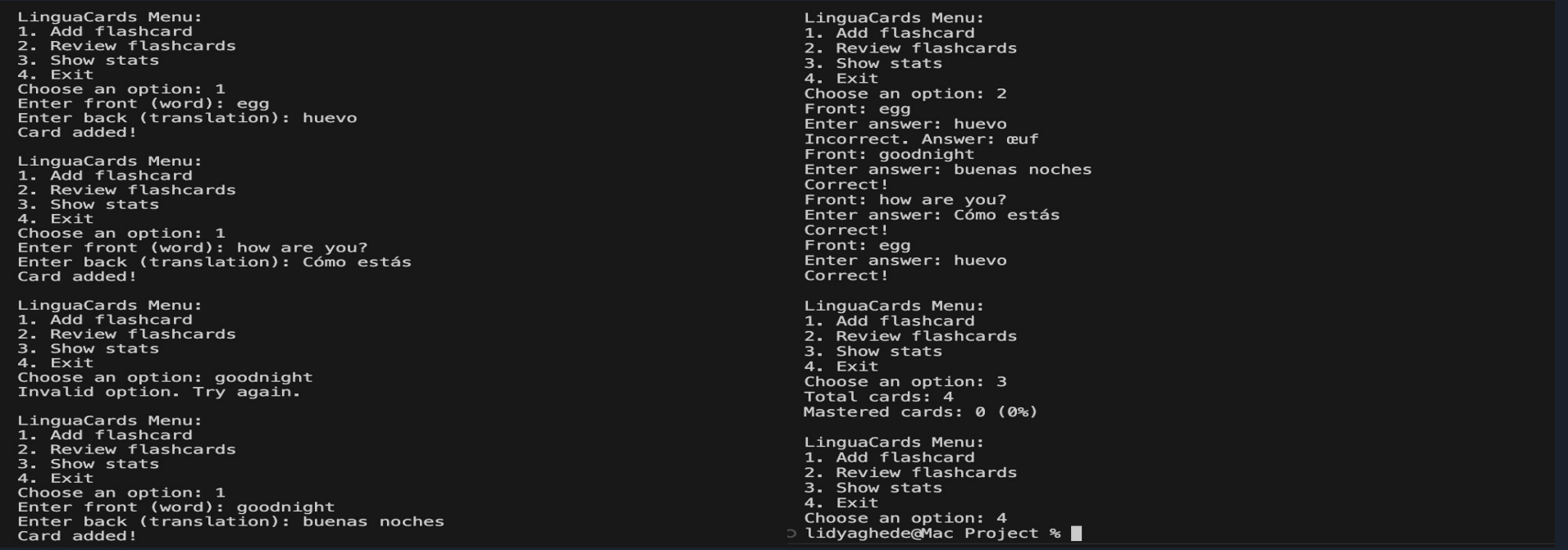
- It displays a menu to the user repeatedly until they choose to exit.
- Calls the corresponding method based on user input. It also keeps the interface clean and user - friendly.





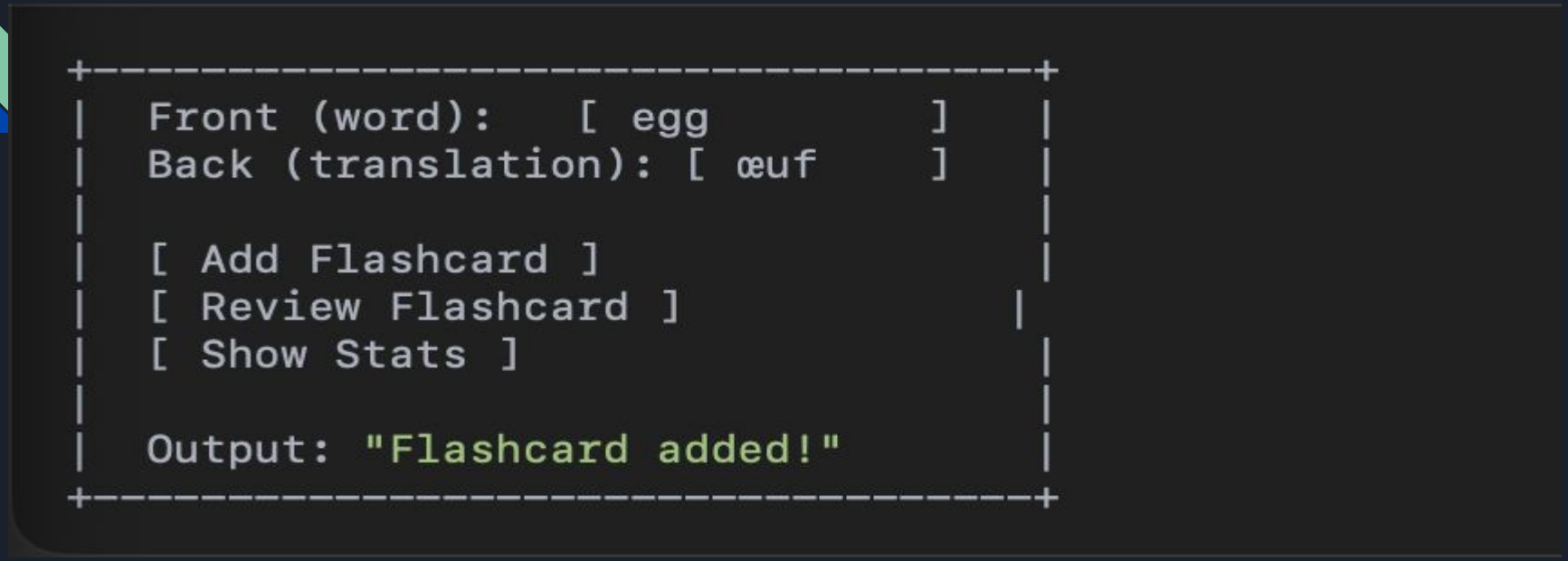
I used g++ -std=c++11 main.cpp -o main to compile my program with modern C++ features and give it a custom name. It ensures my flashcard app works correctly with newer language tools.

This is the GNU C++ compiler it compiles your C++ source code .cpp into a program your computer can run.



The program starts by showing a menu with 4 option:Add FlashCard, Review FlashCards, Show sats and Exit. if the user choose option 1(Add FlashCards) and words like e.g. “egg”-> “ huevo” or “how are you” -> “ cómo estás” or “ goodnight” -> “ buenas noches”. When an invalid input like “goodnight” was entered instead of number, the program correctly handled it by saying “ invalid option. Try again.” the user choose option 2 (Review flashcards) and practice by typing the translation the program will marked the answer as correct or incorrect based on the user input. Finally by choosing option 3 the user saw stats like: “Total cards: 4”, “Master cards: 0(0%). Exiting the program using option 4 ends the session.

Graphical User Interface



This project can be enhanced with a Graphical User Interface (GUI) using the Qt framework for C++. The GUI would replace the text-based menu with input fields and buttons to make the program more user-friendly.
The GUI would include: A text box to enter the flashcard front (e.g., the English word), A text box to enter the back (e.g., the translation), A button to add a flashcard, A button to review flashcards, A button to show statistics, A display area showing system messages (e.g., “Correct!”, or “Flashcard added!”)

Conclusion

In this project, i develop Lingua Cards, a flash cards based language learning tools written in C++. it allows users to:

- Add new vocabulary words and translations, Review FlashCards, view performance statistics.
- I used modular programming techniques with a custom FlashCards struct and a Lingua Cards class to organize the code. The program was compiled using G++ -std=c++11 to ensure compatibility with modern c++ features.
- The key skills applied: file handling, object oriented programming, and input validation and menu systems.
- I learned how to design a simple, interactive console application that mimics real-world education tools.
- This project helped reinforce concepts in C++ and improved my ability to build user focused program.

Thank you!