Code Documentation & Final Report

Group 4 (Brooklyn(fm6391), Sarthak(bv9292), Triet(re2653))

Final Report:

main.cpp

The main entry point of the program.

Contains the main function which runs the program by taking user input and calling relevant functions.

Provides user interface and controls the flow of the program.

login.h

Contains the Login class which provides user authentication functionality.

Has functions for creating new user account, logging in and logging out.

Utilizes C++ standard library functions for file handling and password hashing.

journal.h

Contains the JournalDetails class which provides functionality to create and manage journal entries.

Has functions for creating new journal entry, saving and loading entries to and from a JSON file, removing a journal entry and printing all entries.

Uses the nlohmann/json library for working with JSON data.

Overall, the three files work together to provide a simple journaling application with user authentication functionality. The program allows users to create an account and log in to write and manage their journal entries. Entries can be saved to a JSON file and loaded back in for later use.

main.cpp

The provided C++ code appears to be an implementation of a console-based login system with journal functionality. Here are some comments on the code:

The program is designed as a menu-based console application, where the user is prompted with different options based on their login status. If the user is not logged in, they are given three options to choose from, which are to log in, create an account, or retrieve a forgotten password. If the user is logged in, they are given additional options, including printing their journal entries, creating a new journal entry, removing a journal entry, and signing out.

The LoginClass and JournalDetails classes are defined in separate header files and are imported in the main.cpp file using #include statements. This is a good practice as it separates the code into different modules, making it more modular and easier to maintain.

The main() function defines several variables, including an integer to store the user's input, an instance of the LoginClass, an instance of the JournalDetails class, and two strings to store the user's login credentials.

The main() function also defines a boolean variable called "is_logged_in" to keep track of whether the user is currently logged in or not. This variable is used to determine which options to show to the user in the menu.

Inside the while loop, the program prompts the user to enter a menu option and then uses a switch statement to handle the user's input. If the user is not logged in and selects the "login" option, the program prompts the user to enter their username and password and then calls the LoginClass's user_login() method to check if the credentials are valid. If the login is successful, the program sets the "is_logged_in" variable to true and loads the user's journal entries from a JSON file.

If the user selects the "create account" option, the program prompts the user to enter a username and password and then calls the LoginClass's create_account() method to create a new account.

If the user is logged in and selects the "print" option, the program calls the JournalDetails class's print_journal() method to display the user's journal entries.

If the user is logged in and selects the "add new journal" option, the program calls the JournalDetails class's journal_entry() method to prompt the user to enter a new journal entry. The program then calls the JournalDetails class's create_new_journal() method to create a new journal entry for the user.

If the user is logged in and selects the "remove journal" option, the program calls the JournalDetails class's print_journal() method to display the user's journal entries, prompts the user to enter the ID of the journal entry they want to remove, and then calls the JournalDetails class's remove_entry() method to remove the selected entry.

If the user selects the "sign out" option, the program calls the JournalDetails class's save_as_json() method to save the user's journal entries to a JSON file and then sets the "is_logged_in" variable to false.

If the user enters an invalid menu option, the program displays an error message and terminates.

Overall, the provided code appears to be a good implementation of a console-based login system with journal functionality. However, without the implementation of the LoginClass and JournalDetails classes, it is impossible to determine the correctness of the program.

login.h

This C++ code defines a class called LoginClass that allows users to create accounts, login and stores user data in a JSON file. The class has private member variables for the user's username, password, a vector of user data pairs, and a nlohmann::json object for storing user data in a JSON format.

The class has public member functions that allow users to login, create accounts, save user data as a JSON file, and perform encryption and decryption of user passwords. The user_login function takes the username and password entered by the user and checks if they match the username and password of any user in the users vector. The create_account function creates a new user account and adds it to the users vector. The save_as_json function saves the data stored in the users vector as a JSON file.

The encrypt and decrypt functions perform password encryption and decryption, respectively. The encryption function adds a key of 2 to the ASCII value of each character in the password, while the decryption function subtracts a key of 2 from the ASCII value of each character in the password.

Overall, the code provides basic functionality for user account creation and login with password protection using simple encryption and decryption techniques. However, it should be noted that the encryption used here is not very strong and could be easily broken by an attacker. A more secure approach to password protection should be used in a real-world application.

journal.h

This is a C++ code that defines a JournalDetails class that provides functionality for creating and managing a journal.

The code includes necessary header files, including <ctime> for time-related functions, <tuple> for tuple-related functions, <iostream> for input/output functions, <fstream> for file I/O functions, <sstream> for string stream functions, <filesystem> for filesystem functions, <vector> for vector-related functions, <utility> for utility functions, <string> for string-related functions, and nlohmann/json.hpp for JSON parsing.

The JournalDetails class has several member variables, including an integer journal_id for keeping track of the journal's ID, a string current_time for keeping track of the current time, a string journal_data for storing the journal data, a vector journal for storing the entries in the journal as tuples of (username, timestamp, (ID, entry)), and a JSON object journal_json_data for storing the journal data as JSON.

The class provides several member functions, including journal_entry() for prompting the user to enter new journal data, create_new_journal() for creating a new journal entry with the given username and journal data, print_journal() for printing all journal entries to the console, save_as_json() for saving the journal data as a JSON file, load_json_entries() for loading the journal data from a JSON file, and remove_entry() for removing a journal entry by ID.