Quiz Application

31.07.2023

Team INNOV8
AASTU
Addis Ababa, Ethiopia

Contributors

	Name	ID number
1	Tamirat Dejene	ETS 1518/14
2	Tadiyos Dejene	ETS 1522/14
3	Tebarek Shemsu	ETS 1526/14
4	Yohannes Tigistu	ETS 1703/14

Class of 2026
Section: E

Submitted to: Mrs. Eleni T.
08.04.2023

Github repository link: https://github.com/tamirat-dejenie/Quiz-Application.git

Project Overview

The Quiz Application Project aims to develop a user-friendly and interactive quiz application using the fundamentals of C++ programming language. The application will provide a platform for users to test their knowledge in various subjects by answering multiple-choice questions.
The application will store a collection of questions, each consisting of a question statement, multiple-choice options, and a correct answer. The user will be prompted to select their answer from the given options, and their score will be calculated based on the number of correct answers. At the end of the quiz, the user will receive their final score and feedback on their performance.

Project Requirements and Dependencies

The project is developed using the MSVC compiler that supports C++14 or higher standards. The program utilizes standard C++ libraries for basic operations. It is developed using “Visual Studio 2022 (v143)” IDE. The project depends on external files to store data about the user and required files.

Data and File Dependencies

The project depends on some external files which it uses to store the data related to the specific user and the questions to be displayed. And thus the program should be granted the necessary file reading and writing permission on the system.

After navigating through the directory of theProject file (Quiz Application)

- The directory “Quiz Application\questionBank” - used to store the question files.
- The directory “Quiz Application\userInfo” - to store the user’s information.
- The directory “Quiz Application\newquestionBank” - to store user provided questions.

In the above directories the file is stored in the text file format for the program’s simplicity. And other directories are used to store the build and debug information used by the system (IDE).

Pseudocode and Algorithms

Class and Member Function Declaration

The program makes use of two classes to hold data related to the user and the questions. The following will be the declaration of header files for the class user and question respectively.

Class User

This class has data members: full_name to store user' full name, email to store user's email, user_name to store user's user name, typeof_User to store the user' type and char array of size five to store the password of the user's account. The member function of this class will be defined in the next subsection.

This specific class is defined in a header file called user.h to be included where it is required as follows.

```
//user.h
#ifndef USER_H
#define USER_H
#include <string>
class User {
private:
	std::string full_name, email, user_name, typeof_User;
	char password[5];
public:
	int showMenu();
	void signIn();
	void signUp();
	void saveSignUpInfo();
	bool checkLoginInfo(std::string, char[]);
	void scoreBoard();
	void resetScoreboard();
	void deleteAccount();

	std::string fullName();
	std::string getEmail();
	std::string getUsername();
	char* getPassword();
	char moreOption();

	bool checkPasswordValidity(char[]);
	void encryptPassword(char[]);
```

```
};
#endif
```

Class Question

This class is used to store data related to the questions and to perform functions. This function inherits the public member functions of the class User. The data member of this class declares a vector of string of questions to store the questions, and choice read from the file, vector of character of correctAns to read correct answer from the file and char vector of userAns to read answers from the user. Integer variable score is initialized to zero to store the user's score. And also an integer variable called numofQ to store the number of questions the user wants to take. And finally the double data type is used to store the time elapsed by the user while taking the test.

The following is the class declaration or prototype in the header file "question.h".

```
// question.h
#ifndef QUESTION_H
#define QUESTION_H
#include "user.h"
#include <string>
#include <vector>
using namespace std;

class Question : public User {
private:
        vector <string> question;
        vector <string> choice;
        vector <char> correctAns;
        vector <char> userAns;
        int score = 0, numofQ;
        double timeElapsed;
public:
        char chooseCategory();
        int difficultyLevel();
        void readQuestion(char, int);
        void displayQuestion();
        void saveScore(string, char, int);
        int getScore();
        void freeMemory();
        void evaluateScore();
        void addQuestion();
};
#endif
```

The following will be the pseudocode of the main function. It shows the program's flow of control.

```
1. Include all the necessary file and start of the main function:
   Declare User U and Question Q.
2. Prompt the user with the main menu options and switch based on the selected
option:
   switch (U.showMenu()):
      case 1:
         Call U.signIn();
         break;
      case 2:
         Call U.signUp();
         Call U.saveSignUpInfo();
         Call U.signIn();
         break;
3. Get the category choice from the user:
   Declare char variable category = Q.chooseCategory();
4. Check if the chosen category is not 'F' (i.e., it is a valid category):
   if (category != 'F'):
      Get the difficulty level from the user:
      Declare int variable difficultyL = Q.difficultyLevel();
      Read the question based on the chosen category and difficulty level:
      Call Q.readQuestion(category, difficultyL);
      Display the question to the user:
      Call Q.displayQuestion();
      Get the username of the current user:
      Declare string variable userFile = U.getUsername();
      Save the user's score for the chosen category and difficulty level:
      Call Q.saveScore(userFile, category, difficultyL);
      Evaluate the user's score:
      Call Q.evaluateScore();
      Free the memory used for the question:
      Call Q.freeMemory();
5. Prompt the user for more options and switch based on the selected option:
   switch (U.moreOption()):
      case 'A':
         Call Q.addQuestion();
         Display "Press any key to exit."
         Call system("pause>0");
         Call exit(1);
      case 'B':
         Call U.scoreBoard();
         break;
      case 'C':
         Call system("CLS");
         Go back to the main menu (goto H).
```

```
            break;
        case 'D':
            Call U.resetScoreboard();
            break;
        case 'E':
            Call U.deleteAccount();
            break;
        case 'F':
            Call system("CLS");
            Display "Thanks for using our app!\n Press any key to exit."
            Call system("pause>0");
            Call exit(1);
6. Pause the program to display the results:
   Call system("pause");
7. End of the main function and return 0.
```

The c++ code for the main function:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <iomanip>
#include "user.h"
#include "question.h"
using namespace std;

int main() {
H:      User U; Question Q;
        switch (U.showMenu()) {
        case 1:
                U.signIn();
                break;
        case 2:
                U.signUp();
                U.saveSignUpInfo();
                U.signIn();
                break;
        }
        char category = Q.chooseCategory();
        if (category != 'F') {
                int difficultyL = Q.difficultyLevel();
                Q.readQuestion(category, difficultyL);
                Q.displayQuestion();

                string userFile = U.getUsername();
                Q.saveScore(userFile, category, difficultyL);
```

```
            Q.evaluateScore();
            Q.freeMemory();
        }
        switch (U.moreOption()) {
        case 'A':
            Q.addQuestion();
            cout << "Press any key to exit.";
            system("pause>0");
            exit(1);
        case 'B':
            U.scoreBoard();
            break;
        case 'C':
            system("CLS");
            goto H;
            break;
        case 'D':
            U.resetScoreboard();
            break;
        case 'E':
            U.deleteAccount();
            break;
        case 'F':
            system("CLS");
            cout << "Thanks for using our app!\n Press any key to exit.";
            system("pause>0");
            exit(1);
        }
        system("pause");
        return 0;
}
```

Member Function Definition

Class User's member function definition

The **showMenu()** will display the option for the user and prompts the user to enter the value of the choice in integer value. After reading the input from the user it will check whether the input is valid and returns the value.

The code will be as follows:

```
int User::showMenu() {
```

```
	int choice;
	cout << "---------------------------------------" << endl;
	cout << "----- Welcome to Quiz Application -----" << endl;
	cout << "---------------------------------------" << endl;
	cout << "              1. Log in" << endl;
	cout << "              2. Sign up  "; cin >> choice;
	a:
	if (!(choice == 1 || choice == 2)) {
		cout << "Please enter a valid input: "; cin >> choice; goto a;
	}
	return choice;
}
```

The **signIn()** function will be used for signing in the user. By prompting the user to enter their username and password it will call the other member function to check the inputs given by the user. If the input is correct, the function will open the file related to the user in the file directory. From the file it will read the user data and store each data in the corresponding data members in the class 'User'. The user password will be encrypted and stored in the user data.

The implemented function will be as follows:

```
void User::signIn() {
	string userName; char securityKey[5];
re:	cout << "   User name: "; cin.ignore(); getline(cin, userName);
	cout << "   Password : "; cin.get(securityKey, 5);
	encryptPassword(securityKey);

	bool check = checkLoginInfo(userName, securityKey);
	if (!check) {
		cout << "Incorrect username or password. Retry\n";
		goto re;
	}
	else {
		ifstream inf;
		string fileN = "userInfo/" + userName + ".txt";
		inf.open(fileN);
		if (inf.is_open()) {
			inf.seekg(6);
			getline(inf, full_name);
			inf.seekg(7 + inf.tellg());
			getline(inf, email);
			inf.seekg(10 + inf.tellg());
			inf.get(password, 5);
```

```
                inf.seekg(8 + inf.tellg());
                getline(inf, typeof_User);
                inf.close();
        }
        user_name = userName;
        system("CLS");
        cout << "Logged in as ----  " << user_name << "  ----\n";
    }
}
```

The member function **checkLoginInfo()** checks the input from the user when it is called. By using the user name the function locates the user data. After reading the user's information stored, the data matches with the input, the function will return true.

The following will be the declaration of the function.

```
bool User::checkLoginInfo(string userN, char pass[]) {
    string fileName = "userInfo/" + userN + ".txt";
    ifstream readData;
    string name, Email, type;
    char password[5] = "";
    readData.open(fileName);

    if (readData.is_open()) {
        readData.seekg(6);
        getline(readData, name);
        readData.seekg(7 + readData.tellg());
        getline(readData, Email);
        readData.seekg(10 + readData.tellg());
        readData.get(password, 5);
        readData.seekg(8 + readData.tellg());
        getline(readData, type);
        readData.close();
    }
    bool passwordIsCorrect = true;
    for (int i = 0; i < 4; ++i) {
        if (password[i] != pass[i]) {
            passwordIsCorrect = false;
            break;
        }
    }
    return passwordIsCorrect;
}
```

The function **signUp()** prompts the user to enter all the data belonging to the user to be stored in the data member of the class. After reading the password, it will check the validity of the password using the function. If the function returns true, the password will be encrypted and saved in the data member.

The following is the implementation for the function:

```
void User::signUp() {
	string fn, ln;
	cout << "Well :):\n";
	cout << "  First name          : "; cin.ignore(); getline(cin, fn);
	cout << "  Last name           : "; getline(cin, ln);
	full_name = fn + " " + ln;
	cout << "  E-mail address      : "; getline(cin, email);
	cout << "  Create user name    : "; getline(cin, user_name);
	cout << "  Password (4 char)   : ";

p: cin.get(password, 5);

	if (checkPasswordValidity(password))
		encryptPassword(password);
	else {
		cout << "Please enter a valid password: "; goto p;
	}
}
```

The function **saveSignUpInfo()** will be functioning the saving process of the data stored in the data member of the class in a separate file for each user in the given directory. The function makes use of the file stream and appropriate file opening and closing system. If the saving process goes successfully it will proceed, if not it will notify the user that the operation was unsuccessful and exits from the program.

The following is the function's definition:

```
void User::saveSignUpInfo() {
	char choice;
	system("CLS");
	cout << "   ----- Sign-up as -----" << endl;
	cout << "          a. Student" << endl;
	cout << "          b. Teacher" << endl;
	cout << "          c. Other" << endl;
	cout << "*-*-*--> "; cin >> choice;

	system("CLS");
	string fileName = "userInfo/" + user_name + ".txt";
```

```
	ofstream newUser;
	newUser.open(fileName);

	if (newUser.is_open()) {
		newUser << "Name: " << full_name << endl;
		newUser << "Email: " << email << endl;
		newUser << "Password: " << password << endl;
		switch (choice) {
		case 'a':
			newUser << "Type: Student\n"; break;
		case 'b':
			newUser << "Type: Teacher\n"; break;
		case 'c':
			newUser << "Type: Other\n" << endl;
		}
		newUser.close();
		system("CLS");
		cout << "--------You have successfully created your account!--------"
<< endl;
		cout << "	a. Back to log in page\n"
			<< "	b. Exit -------------> "; cin >> choice;
		switch (choice) {
		case 'a':
			break;
		case 'b': {
			cout << "Thanks for registering.\n";
			system("pause");
			exit(1);
		}
		}
	}
	else {
		cout << "Operation Unsuccessful! Retry later.\n";
		system("pause");
		exit(1);
	}
}
```

The function **checkPasswordValidity(char pass[])** will accept an array of characters which is the encrypted array of password characters, it will check the user's password length and type with the character arrays passed to it. If the data meets the required quality the function will return true.

The function will be as follows:

```
bool User::checkPasswordValidity(char pass[]) {
	bool isCorrect = true;
	int i = 0;
	while (pass[i] != '\0')
		++i;
	if (i != 4)
		isCorrect = false;
	return isCorrect;
}
```

The function **encryptPassword(char pass[])** accepts an array of validated characters of the password and encrypts the password using the algorithm provided in the function and updates each character value with the encrypted one. The algorithm can be changed as required.

The following is the implementation of the function.

```
void User::encryptPassword(char pass[]) {
	int i = 0;
	while (pass[i] != '\0') {
		if (int(pass[i]) <= 91)
			pass[i] = char(int(pass[i]) + 10);
		else
			pass[i] = char(int(pass[i]) - 10);
		++i;
	}
}
```

Since the class's data members are private to the class meaning the data can only be accessed by the member function or only inside the class, we make use of the getters to access those data properly. These getters are defined as the public member functions of the class and they will return the required value when called. And the following will be the definitions of all the getters declared in the class.

```
string User::fullName() {
	return full_name;
}
```

```
string User::getEmail() {
	return email;
}
```

```
string User::getUsername() {
	return user_name;
}
```

```
char* User::getPassword() {
	return password;
}
```

The member function **resetScoreboard()** is used to reset the user's score list saved up in the user's file. The score is stored in the user's profile starting from the specific line in the text file used to save the user's information. The function declares a vector string called lines. After opening the user's file using the user's user name, it will go on every line of the code and saves the data in the vector of string. After reading the file the function will rewrite only the user profile (ignoring the scores) in the user's file. After the function is performed it will report the status and exits the program.

The function's implementation code will be as follows.

```
void User::resetScoreboard() {
	string filePath = "userInfo/" + user_name + ".txt";
	vector<string> lines;
	string line;

	ifstream infile;
	infile.open(filePath);
	if (infile.is_open()) {
		while (getline(infile, line)) {
			lines.push_back(line);
		}
		infile.close();
	}
	const int start_l = 5;
	const int end_l = lines.size();

	ofstream outFile(filePath, ios::out | ios::trunc);
	if (outFile.is_open()) {
		for (int i = 0; i < start_l - 1; ++i)
		{
			outFile << lines[i] << endl;
		}
		outFile.close();
	}
	system("CLS");
	cout << "Erasure successful!\n";
	cout << " Press any key to exit.";
	system("pause>0");
```

```
	exit(1);
}
```

The **moreOption()** function displays the additional functionality that can be performed by the program and returns the choice of the user by checking whether the input from the user is valid. If the input is invalid the user will be re-prompted to enter his/her choice.

The implementation:

```
char User::moreOption() {
	char moreChoice;
	cout << "Press any key to proceed."; system("pause>0");
	system("CLS");
	cout << "More options :)\n";
	cout << "  a. Add own question\n"
		<< "  b. Show my scoreboard\n"
		<< "  c. Back to homepage\n"
		<< "  d. Reset scoreboard\n"
		<< "  e. Delete Account\n"
		<< "  f. Exit --- >> ";  k: cin >> moreChoice;
	if (moreChoice == 'a' || moreChoice == 'b' || moreChoice == 'c' ||
moreChoice == 'd' || moreChoice == 'e' || moreChoice == 'f') {
		moreChoice = toupper(moreChoice);
		system("CLS");
		return moreChoice;
	}
	else {
		system("CLS");
		cout << "Please enter valid input: "; goto k;
	}

}
```

The function **scoreBoard()** will be used to display the scoreboard of the user by reading from the file of that user. It makes use of file streams to read the data and the scores are stored starting from line 5 in the user data and displays the score read from the file.

The implementation will be as follows:

```
void User::scoreBoard() {
	string username = getUsername();
	ifstream scoreOut;
	string score;
	string fileName = "userInfo/" + username + ".txt";
```

```
        scoreOut.open(fileName);
        int line = 1;
        if (scoreOut.is_open()) {
                while (!scoreOut.eof() && getline(scoreOut, score)) {
                        if (line > 5)
                                cout << score << endl;
                        ++line;
                }
                scoreOut.close();
        }
}
```

If the user wants to delete the account the function **deleteAccount()** is called and used to delete the file. Using the user name of the user the user's file will be located in the file directory and if the user confirms the file will be removed.

The function implementation will be as follows:

```
void User::deleteAccount() {
        char decision;
        cout << "Are you sure? You can't undo this. (y/n) "; cin >> decision;
        if (decision == 'y' || decision == 'Y') {
                string userFile = "userInfo/" + user_name + ".txt";
                if (remove(userFile.c_str()) != 0) {
                        perror("Error deleting the account! Retry Later.");
                        exit(1);
                }
                else {
                        cout << "Successfully deleted!\n";
                        cout << "Press any key to exit :) "; system("pause>0");
                        exit(1);
                }
        }
}
```

Class Question's member function definition

As we can understand from the class's declaration or prototype, this class inherits the public member functions of the class User. And the member function definition of this class will be as follows.

The **chooseCategory()** member function of this class is used to display the category of the question that the user may want to choose and prompts the user to enter the value of his/

her choice. Then the function will make sure that the user has entered the valid input and returns the input.

The function code is as follows.

```
char Question::chooseCategory() {
	char ch;
	cout << "--- Choose type of the questions ---" << endl;
	cout << "	a - Science related\n"
		<< "	b - General knowledge related\n"
		<< "	c - Technology related\n"
		<< "	d - Sport related\n"
		<< "	e - Do some random questions\n"
		<< "	f - More options   ---- >> "; cin >> ch;
k:	ch = toupper(ch);
    if (!(ch == 'A' || ch == 'B' || ch == 'C' || ch == 'D' || ch == 'E' ||
ch == 'F')) {
	system("CLS");
	cout << "Enter valid input: "; cin >> ch;
	goto k;
}
system("CLS");
return ch;
}
```

As we (the developers) tried to mention in the project proposal, one of the functionality of the program is to provide questions with different levels of difficulty. The function **difficultyLevel()** prompts the user to choose the difficulty level and after validating the input it will return the user's choice.

The implementation goes as follows:

```
int Question::difficultyLevel() {
	int d;
	cout << " --- Choose difficulty level ---\n";
	cout << "   1 - Easy\n"
		<< "   2 - Medium\n"
		<< "   3 - Difficult --- > "; cin >> d;
l:   if (!(d == 1 || d == 2 || d == 3)) {
	system("CLS");
	cout << "Enter valid input: "; cin >> d;
	goto l;
}
system("CLS");
return d;
```

```
}
```

The function **readQuestion(char, c int d)** takes two arguments which are the category choice and difficulty level choice of the user passed to it. By using these arguments the function will locate the desired file in the question bank(file) stored in the directory. The function declares input file stream, string and character variables. The input file stream is used to read the desired data from the located file to the desired variable. The string declared will be used as a temporary data holder and it is used to save data each time of the loop in the data member of the class.

The implementation of the class will be as follows.

```
void Question::readQuestion(char c, int d) {
	ifstream qFile;
	c = tolower(c);
	switch (c) {
	case 'a':
		if (d == 1) qFile.open("questionBank/sciEasy.txt");
		else if (d == 2) qFile.open("questionBank/sciMedium.txt");
		else qFile.open("questionBank/sciHard.txt");
		break;
	case 'b':
		if (d == 1) qFile.open("questionBank/genEasy.txt");
		else if (d == 2) qFile.open("questionBank/genMedium.txt");
		else qFile.open("questionBank/genHard.txt");
		break;
	case 'c':
		if (d == 1) qFile.open("questionBank/techEasy.txt");
		else if (d == 2) qFile.open("questionBank/techMedium.txt");
		else qFile.open("questionBank/techHard.txt");
		break;
	case 'd':
		if (d == 1) qFile.open("questionBank/spoEasy.txt");
		else if (d == 2) qFile.open("questionBank/spoMedium.txt");
		else qFile.open("questionBank/spoHard.txt"); break;
	case 'e': qFile.open("questionBank/randomQuest.txt");
		break;
	}
	string strInput; char charInput;
	int i = 0;
	if (qFile.is_open()) {
		while (!qFile.eof()) {
			getline(qFile, strInput); question.push_back(strInput);
			while (i < 4) {
				getline(qFile, strInput);
choice.push_back(strInput);
```

```
                        ++i;
                }
                i = 0;
                qFile.seekg(8 + qFile.tellg());
                qFile.get(charInput); correctAns.push_back(charInput);
getline(qFile, strInput);
                getline(qFile, strInput);
        }
        qFile.close();
    }
    else
        exit(1);
}
```

The **displayQuestion()** is used to display the question read by the above function. This function prompts the user to enter the number of questions they want to take and displays the desired number of questions using the for loop. This function also makes use of a special function for the timer. After displaying each question the user is prompted to enter their answer and the user answer will be stored and compared with the correct answer read from the file and tells the user the status. At the end the timing will be stopped and saved.

The following display is the functions algorithm.

```
void Question::displayQuestion() {
    int numofchoice = 0, totalChoice = 0;
    char userInput;
    cout << "  How many questions do you want to enjoy( < 10): "; cin >> numofQ;
    system("CLS");
    cout << "Let's go! The timer has started :)\n\n";
    auto start = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < numofQ; ++i) {
        cout << i + 1 << ". " << question[i] << endl;
        while (numofchoice < 4) {
            cout << choice[totalChoice] << endl;
            ++totalChoice, ++numofchoice;
        }
        numofchoice = 0;

        cout << "--- "; cin >> userInput;
userAns.push_back(toupper(userInput));
        if (userAns[i] == correctAns[i]) {
            cout << "Correct! \n";
            ++score;
        }
        else
```

```
                cout << "Incorrect! Correct ans: " << correctAns[i] << endl;
            system("pause");
            system("CLS");
        }
        auto end = std::chrono::high_resolution_clock::now();
        chrono::duration<double> duration = end - start;
        timeElapsed = duration.count();
}
```

The following function's getter member function is used to return the score saved in the private data member of the class.

```
int Question::getScore() {
        return score;
}
```

The **saveScore(string, char , int)** is used to save the arguments that are passed to it. The first argument is used to pass the user's user name, the second is the question category and the third one is the difficulty level of the question. By opening the user's file using the user name the function is used to append the score of the user on the user's file.

The function's implementation will be as follows.

```
void Question::saveScore(string user, char questionCategory, int difficultyLevel) {
        ofstream userf;
        user = "userInfo/" + user + ".txt";
        userf.open(user, ios::app);
        if (userf.is_open()) {
            userf << "\nCat - " << questionCategory
                << " : D-level - " << difficultyLevel
                << " : Score - " << score << "/" << numofQ
                << " : Time - " << timeElapsed << " sec";
            userf.close();
        }
}
```

For the efficient allocation of memory the function **freeMemory()** is used to clear memory occupied by the vector data members of the class. By using the special method (.clear) of the vector it will clear the memory space occupied by these variables.

```
void Question::freeMemory() {
        question.clear();
        choice.clear();
        userAns.clear();
        correctAns.clear();
}
```

The function **evauluateScore()** is used to inform users the time elapsed, and out of hundred score.

The following displays the implementation of the function.

```
void Question::evaluateScore() {
	cout << "Time elapsed: " << timeElapsed << " seconds\n";
	double percentageScore = ((double)score / numofQ) * 100;
	if (percentageScore >= 75)
		cout << "Excellent! You scored " << percentageScore << "% ";
	else if (percentageScore >= 50)
		cout << "Good! You scored " << percentageScore << "% ";
	else if (percentageScore >= 25)
		cout << "Try hard! You scored " << percentageScore << "% ";
	else
		cout << "Poor! You scored: " << percentageScore << "% ";
	cout << " or " << score << "/" << numofQ << endl;
}
```

The function **addQuestion()** is used to add questions and this function will be released in the future updates.

```
void Question::addQuestion() {
	system("CLS");
	cout << "Subscribe for this service: only $20.00/year\n";
}
```