Project Documentation

Title page:

Course name: Basics of Programming 2
Course ID: BMEVIIIAA03
Faculty Code: 5NAA8
Full Name: Maksatbek uulu Azim
Neptun code: HW9NCU
Lab Group ID: CS16D
Lab Instructor: Vaitkus Márton
Project title: Social-Media Management
Type of the Project: Final Project
Submission date: 18/05/2024

Introduction

The Social-Media Management is console-based application designed to stimulate the basic functionalities of the social-media platforms. The primary goal of the program is to allow users to create and manage posts, view posts from other users and manage data of the users. File management system is used to save and control the information throughout the program.

Program Interface

To interact with the Social Media Management, the user should set up the programming environment and execute the program either through the command line or using an Integrated Development Environment.

1. ### Setting up the environment

   Ensure that all the source files (**“Admin.cpp”, “MediaMain.cpp”, “Post.cpp”, “User.cpp”, “UserManager.cpp”)** and the header file (**“Media.h”)** are in the same directory.

   Ensure the data files (**“userdata.txt”, “users.txt”, “userpost.txt”)** are also in the same directory

2. ### Compiling the Program

   If you are using the command line than compile and run it using appropriate commands:

   e.g.

   g++ -o SocialMedia MediaMain.cpp Admin.cpp Post.cpp User.cpp UserManager.cpp

   ./SocialMedia

   If you are using the IDE, use the build or compile option to compile the program and run the program using run option.

3. Terminating the program

The program can be terminated by selecting the exit option from the menu within the program

Program execution

1. Initialization of the application

Upon starting the program, the user will be given three choices:

1. Register
2. Log in
3. Exit!

If the user already has used this application, then login option can be entered. However, if it is the first time using the program, 'Register' should be chosen. After, user is requested to enter the name, username, and password that will be use throughout the program.

2. Working with the application

After signing in, various choices will be available to the user depending on the status:

If user is **regular user**, the following options will be available:

1. Create a post
2. See the other users' posts
3. list my posts
4. Delete post
5. list all the people in the community
6. search by Username
7. My profile!
8. Display MENU
12. Exit!

If the user is **admin**, following additional options will be visible

9. Make an admin
10. Delete user
11. Delete others' post

User may choose the options by typing the corresponding numbers.

Main Commands:

a. Creating a post

The user can create a new post by selecting '1' from the menu option and entering the required details such as content of the post.

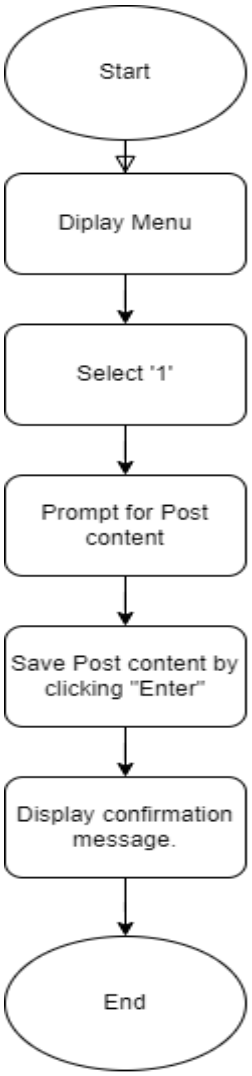
b. Viewing Posts

User can view own posts as well as the posts of the other users. When viewing the others' posts, it is possible to add a comment.

c. Managing the User data

Those functions are only accessible to 'Admins' of the application. The Admins can make other regular users 'admin' of the program and delete the users. They can also delete the other user's posts.

You can see the flow chart below as a demonstration of creating a post:



3. Exiting from the application

   You can exit from the application by choosing the ‘12’ from the Menu option.

Input and output

1. Inputs

   User inputs are taken via the command line for operations such as creating posts, entering user details, and selecting menu options

2. Outputs

   The program outputs data to the console, such as menu options, confirmation messages, and lists of posts or users.

3. Inputs from files

“userpost.txt”: The data about the posts of the users of the application is located here. Upon starting the program, the data is uploaded from the file to program itself.

“users.txt”: The passwords of the users are located in this file for the safety. This file also is checked by the program when someone is login in to the application to check the validity of the user.

“userdata.txt”: Other information about the users are located in this file such as age, job etc. Upon starting the program, the data in this file is also uploaded to program.

4. Outputs to files

“userpost.txt”: Every new post is uploaded to this file. The format of the file is following:

Username: AZM
-Post: 1
-Content: hey I am great to be here.
---Comments:
----comment 1: Super!!
-----comment 2: perfect!!
-----comment 3: Tractor!
postend

“users.txt”: When new user is registered, the username and password are uploaded here. The format is following:

Users	*Password*
AZM	*Azim4545*
Azam	*8888*

“userdata.txt”: The data about users such as age, job and name are outputted here. The format is following:

User 1
--Admin: 1
--Name: Maksatbek uulu Azim
--Username: AZM
--Age: 20
--Occupation: Programmer

Program structure

1. Modules and Classes

The Social Media Management is structured into several key modules, each responsible for handling different aspects of the system’s functionality. There are 4 classes which play different roles withing the program: **User class, Admin class (**Inherited from User**),** **Post class** and **UserManager class.** The classes defined and implemented in separate modules. Each module will be explained below

a. Main program

MeidaMain.cpp: This file contains the main function, which serves as the entry point for the program. It controls the overall execution flow by initializing necessary components, displaying the main menu, and handling user inputs to navigate through different functionalities. The main function ensures that the program operates cohesively by coordinating interactions between various modules.

b. User management

User.cpp: This file defines the **User class**, which encapsulates user-related functionalities such as creating new users, authenticating users, and managing user profiles. The class includes methods for setting and getting user details and updating user information.

Admin.cpp: This file defines the **Admin class**, which provides administrative functionalities. The **Admin class** is inherited from the **User class** and consists additional privileges such as deleting the others' posts, deleting users and making regular users admin of the group with the help of the polymorphism. This class ensures that administrative tasks are performed efficiently and securely.

UserManager.cpp: This file consists of the functions of the **UserManager class**, which is responsible for managing the data across the platform. This class updates the information about the users when it is necessary and saves it files used within the program such **'users.txt', 'userdata.txt'** and '**userpost.txt'**

c. Declarations

Media.h: All the declarations of the classes as well as the declarations of their methods and attributes are located in this file. Moreover, the libraries are also included at the beginning of this file.

2. Data structures and libraries

The Social Media Management utilizes a variety of data structures and libraries to manage and organize data efficiently. Below is a detailed explanation of the data structures and libraries used in the program:

a. Data structures

Dynamically allocated array of pointer to user objects: This is the main data used across the program. It is located in the UserManager class as the attribute of it:

User ** udata;

Constructors and destructors are used to add user to this array and delete it without memory leakages.

std::string: This data structure is used extensively for handling text data such as usernames, passwords, and post content.

std::vector: Used to store collections of posts and comments. It provides dynamic array functionality, allowing for efficient addition and removal of elements:

b. Libraries

Libraries which are included at top of the 'Media.h' file and their purposes:
#include <iostream>: For the standard input and output operations, such printing to the console and reading user input.

#include <vector>: Provides support for the **std::vector** class, enabling the use of dynamic arrays to store collections of objects like comments and posts.

#include <string>: Provides support for the **std::string** class, which is used for handling text data throughout the program.

#include <fstream>: Provides support for file stream operations, enabling the program to read from and write to files. This is crucial for persisting user data and posts.

#include <sstream>: Provides support for string stream operations, which are useful for parsing and formatting strings. It is used for **std::istringstream** within the program, to extract the data elements from the line of the text.

#include <iomanip>: Provides support for input/output manipulators, allowing for more precise control over the formatting of output, such as setting the width of fields.

#include <limits>: Provides support for querying the properties of arithmetic types, such as the maximum and minimum values that a type can hold. This is useful for handling edge cases and ensuring robust input validation.

#include <exception>: Provides support for exception handling, enabling the program to gracefully manage and respond to runtime errors

Testing and verification

The testing of the program was done in three steps: Unit testing, Integration testing, and User testing.

1. ### Unit testing:

   Each module and function of the Social Media Management was tested separately to ensure that they work correctly and in comprehensive way.

   a. **User Module Testing**
      Tested user creation, deletion, and update functionalities.
      Verified that user data is correctly stored and retrieved from **"userdata.txt"** and **"users.txt"**

   b. **Post Module Testing**
      Ensured that posts can be created, viewed, and deleted.
      Verified that post data is correctly stored and retrieved from **"userpost.txt"**

   c. **Admin Module Testing**
      Tested administrative functions such as viewing all users and posts.

Verified that admin-specific operations do not interfere with regular user functionalities

d. **Manager Module Testing**
Ensured that all the files are updated correctly within the program.

2. Integration testing

The integration of the modules and overall system was tested to verify seamless interaction between classes.

a. **User (Admin) and Post integration.**
Verified that users can create posts and are correctly associated with the user.

b. **User Manager and Admin integration.**
Tested admin functionalities and their interaction with Manager functionalities.
Verified that Admin can manage the whole program by deleting posts and users of the program.

3. User testing

The program was tested by the volunteers to make sure that it is comprehensive and understandable to a regular user.

a. **Usability Testing**
Volunteers was navigated to perform tasks throughout the program.
Afterwards, feedbacks were considered to enhance a ease of use, menu structure and clarity of the instructions.

b. **Functional Testing**
Volunteers have entered invalid data by which program was verified for the durability when unexpected orders are given.
Tested that all menu options and functionalities work as intended.

Improvements and Extensions

Potential improvements:

1. **More developed comments:**
The comments can be seen when viewing the posts. However, one will not able to see who wrote it. So, one improvement is adding this feature into the program.
2. **Enhanced security:**
The security of the program is not advanced. One can access the passwords of the users by only getting access to the file **"users.txt".** Due to this, program can be further improved by using hashed passwords or another method to secure the data about users.

3. **Using GUI:**
   This program is basic console-based program. For the better experience and convenience, the application can be extended to include a graphical user interface (GUI).

Difficulties encountered

1. **Managing File I/O Operations Efficiently**
   Efficiently managing file input and output operations was challenging. The system relies on text files to store user data, posts, and other relevant information. Ensuring that these files are read from and written to correctly was crucial for the program's performance and reliability.

2. **New programming language and OOP.**
   Since C++ is new programming language for me, it was difficult to learn about the language's features and create a project at the same time. Additionally, building a program based on the OOP principles was also significantly challenging, since I am also new in this.

Conclusion

The Social Media Management successfully stimulates a basic social-media platform, providing essential functionalities such as user account management, post creation. The project also demonstrates the principles of the OOP, concepts of the file I/O operations in a good way. However, the improvements and extensions, which were indicated in the corresponding chapter, can also be applied for more robust and advanced application.
Overall, this project was significant step for understanding the principles of the OOP and features of the C++ programming language in more advanced level.

References

[1] Stack Overflow, "How to handle wrong data type input?," Stack Overflow, Apr. 27, 2012. [Online]. Available: https://stackoverflow.com/questions/10349857/how-to-handle-wrong-data-type-input [Accessed: May 10, 2024].

[2] Brad Koch, "istringstream: how to do this?," Stack Overflow, Feb. 25, 2010. [Online]. Available: https://stackoverflow.com/questions/2323929/istringstream-how-to-do-this. [Accessed: May 3, 2024].

[3] GeeksforGeeks, "File Handling in C++," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/file-handling-c-classes/. [Accessed: May 3, 2024].