

**Transparent Government Display System:
Conceptual Implementation of CRUD approach**

Demain, Jaime Luis M.
Feria, Louis Andrei M.
Lara, Juan Paulo C.
Mendoza, Nathaniel B.

Technological Institute of the Philippines
Quezon City

November 2025

Table of Contents

Table of Contents.....	2
Introduction.....	3
The Project.....	4
Objectives.....	4
Flowchart of the System.....	5
Pseudocode.....	8
Data Dictionary.....	9
Code.....	14
Results and Discussion.....	17
Conclusion.....	22
References.....	23
Appendices.....	26

Introduction

The lack of transparency in government has become a major challenge in today's society as it obscures evidence which indicates the truth. These issues in the government lead to the misuse of funds, corruption, and the failure to meet project and budget goals. This makes the situation unbelievable for taxpayers, as they find it difficult to understand where the money is being spent, resulting in government corruption and mismanagement. Furthermore, the lack of transparency conflicts with public distrust. People become more concerned because government spending may lack the necessary oversight, especially when funds are transferred outside of regular budget cycles, especially when there are no consistent or open financial records (Farazmand et al., 2022). In addition to that, the credibility of the published projects is one thing that most people find suspicious.

Difficulties in tracking the usage of taxes, a lack of consistent records, and if it were to be based on hearings that there's barely any evidence presented at all, which leads to an inability to see the records. Due to this general problem that the whole country is facing, these specific issues are likely to be agreed upon by most citizens. It is the visibility of track records of funds on where it is being used that will bring transparency for the citizens, since this kind of problem is quite important not only for the taxpayers/citizens but also for government officials, as well, since not only it can be used to present the usage logs but also to present as evidence for future uses. Additionally, research highlights that the failure of transparencies often arise from "transparency anti-pattern" which makes something look transparent on the outside but actually hides and blocks information on the inside that makes it questionable as it consists of incomplete and delayed documentation and information (Zuijderwijk et al., 2025). On the other hand, the study of (Abhishek et al., 2022) propose technological solutions, like blockchain-based fund tracking systems, which could make financial records tamper-proof and traceable, this suggests that it should be transparent and traceable in addition to approach of Republic Act No. 6713 and Executive Order No. 2 2016 in the Philippines.

Through the approach of Republic Act No. 6713 and Executive Order No. 2 2016, an accessible system for Filipino citizens containing publicly disclosed documents and records can allow for government transparency and public trust. This system will only contain information about the usage logs of the funds on where it is being used so that it will not violate any laws. With this feature, the public will be able to keep updated to the actions that are being taken with the funds, ensuring that the taxes they are paying are going for the greater good.

The Project

This system will present the information of different projects that have been issued. In this way, we can address its transparency problem to create public trust by detailing where the funds are being allocated and the specific projects they support with the usage of tax. Through this system, all the information about this project will be presented to the citizens for them to track and observe the government's use. Therefore, they would not be worried about corruption. With the use of a register and log-in feature, the public will be able to see the dashboard containing the projects that were made in the past month to now, and they can also see the old files that have been deleted but archived due to being outdated. This can also be used against someone who disobeyed the law. Alongside the dashboard, they can observe and see details like: Release statement of the project, the total budget data and how much was already spent on the project. This Dashboard is available for all citizens to see, even the government officials.

For managing the system itself, like implementing new information and announcements, there's a separate log-in feature where dedicated personnel can log-on to. They can add, edit, and delete all the information inside the system. Logging in with a verified account will give you privileges to edit all the information inside the system, making updates with certain information and adding more if new projects are released or there have been changes to the project for the public to see.

Objectives

The main objective for this project is to display the funds usage for citizens to see, so that the people are aware of where it is being used. Specifically, the project aims to:

1. Develop a management system that:
 - a. Integrates a log-in feature to access information.
 - I. For local user log-in (For the public with view feature only)
 - II. For administrative log-in (For dedicated users to edit, delete, update, etc.)
 - b. Read user credentials and privileges
 - c. Displays and creates all project information such as, budget used, etc.
 - d. Deletes outdated projects BUT stores it in the archive.
2. Test and evaluate the system's accuracy.

Flowchart of the System

This shows the visual representation of process flow and decision points of the code.

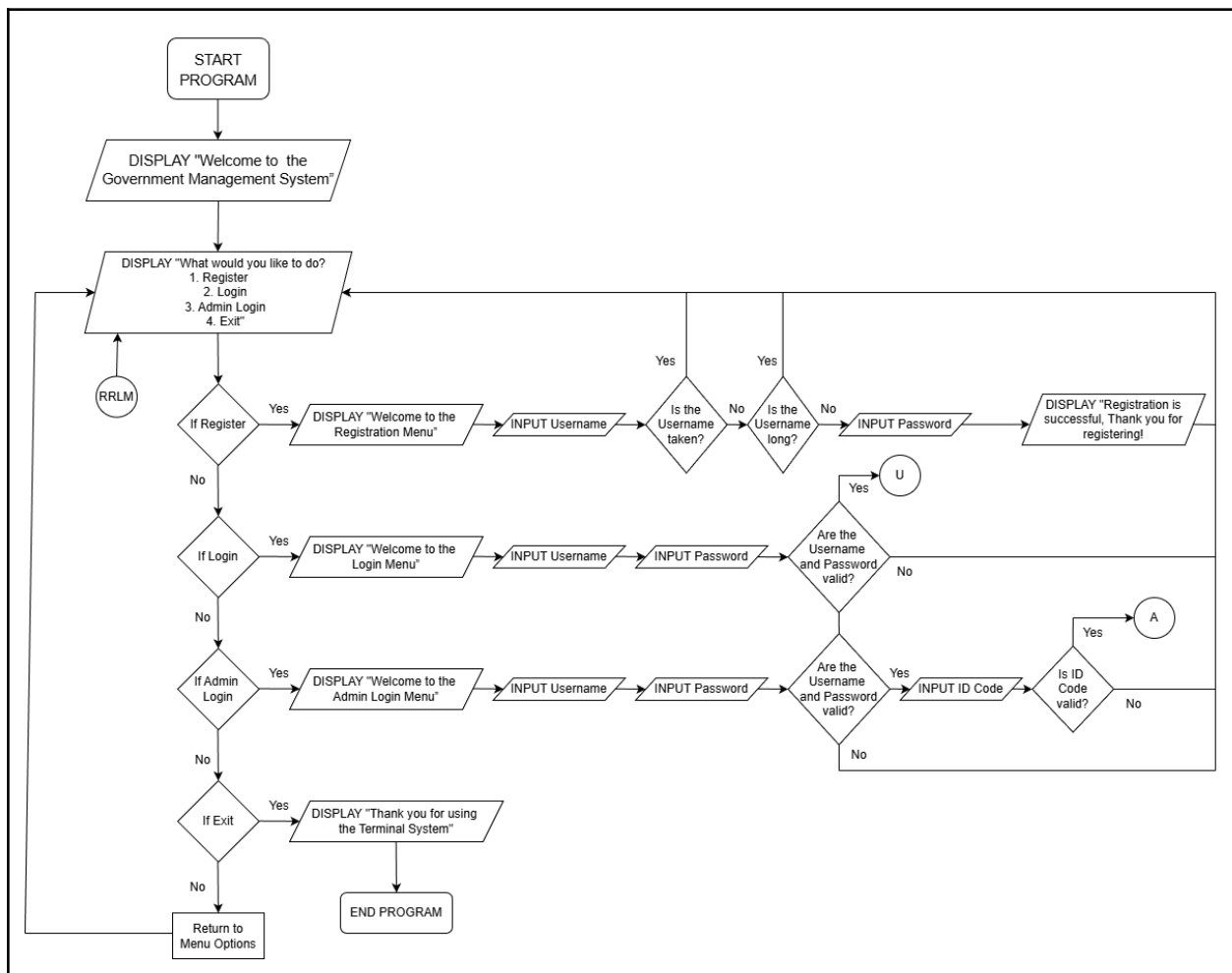


Figure 1: Flowchart Part 1

Figure 1 Illustrates flowchart for Register/Login Menu and its process

Abbreviations:

A - Admin

U - User

RRLM - Return to Register/Login Menu

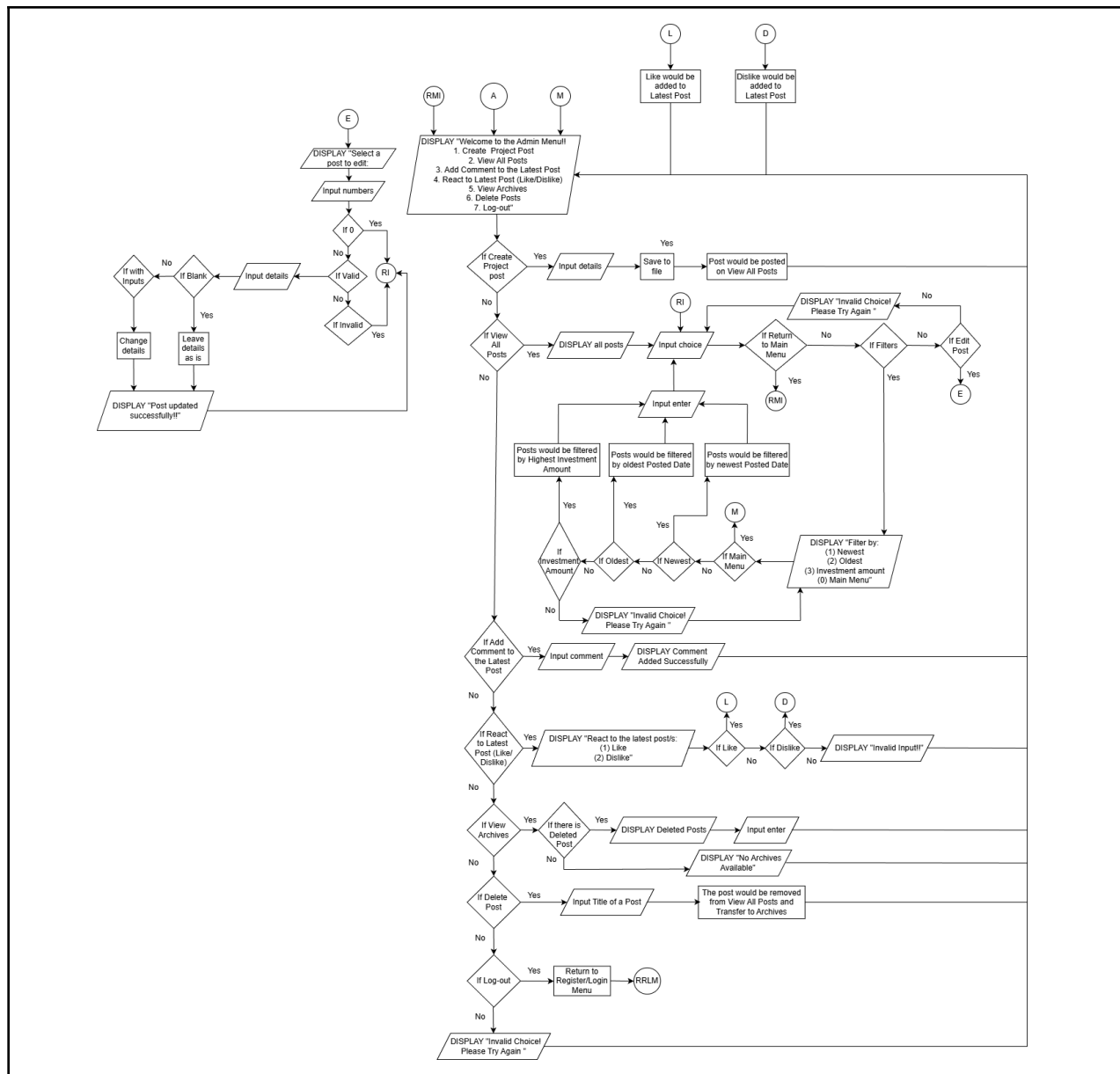


Figure 2: Flowchart Part 2

Figure 2 Illustrates flowchart for Admin Menu

Abbreviations:

A - Admin

E - Edit

L - Like

D - Dislike

M - Menu

RI - Return to Input

RMI - Return to Menu from Input

RRLM - Return to Register/Login Menu

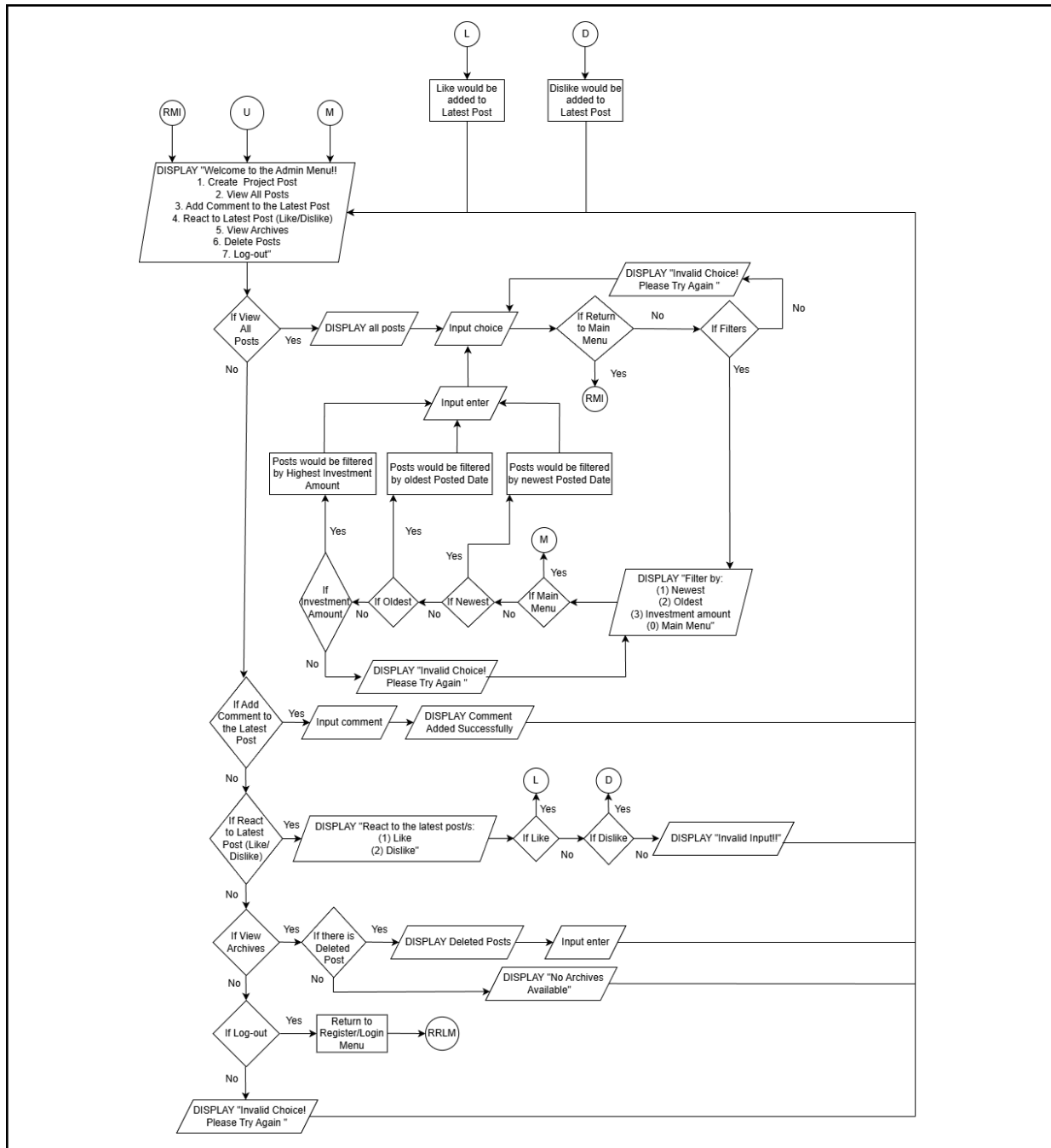


Figure 3: Flowchart Part 3

Figure 2 Illustrates flowchart for User Menu

Abbreviations:

U - User

L - Like

D - Dislike

M - Menu

RMI - Return to Menu from Input

RRLM - Return to Register/Login Menu

Pseudocode

This represents the logical outline of the program in a simplified language.

```
START PROGRAM
  DISPLAY "Welcome to the Government Management System"

  REPEAT
    DISPLAY "What would you like to do?"
    DISPLAY "(1) Register"
    DISPLAY "(2) Login"
    DISPLAY "(3) Admin Login"
    DISPLAY "(4) Exit"
    PROMPT user for choice

    IF choice = 1 THEN
      DISPLAY THE REGISTRATION MENU
    ELSE IF choice = 2 THEN
      DISPLAY THE LOGIN MENU
      IF login is successful THEN
        DISPLAY THE LOCALMENU
      END IF // END THE IF STATEMENT
    END IF
    ELSE IF choice = 3 THEN
      DISPLAY THE ADMIN LOGIN MENU
      IF admin login is successful THEN
        INPUT AN ID CODE // ID FOR VERIFY
        IF code is correct THEN
          DISPLAY THE ADMIN MENU
        END IF
      END IF
    ELSE IF choice = 4 THEN
      DISPLAY "Thank you for using our system!!"
      EXIT LOOP
    ELSE
      DISPLAY "Invalid Main Menu choice!"
      WAIT for 2 seconds
    END IF
  UNTIL choice = 4
END PROGRAM
```

Figure 4: Pseudo code of the system

Figure 4 presents the Pseudocode of the code and its logic behind the actual code. Presented here is the starting point of the system, it greets the user with a message of welcome, and then prompted to the menus where you can select different features like register to the system, login to the system or as an admin where you can edit, delete, and view the information.

Data Dictionary

This table contains the descriptions of data names used in the system according to their size in file and memory, data type used for the initialized data names and brief information about the data names in context of its function in the system. Each command that interacts with the data, such as reading and writing the contents from memory and local storage in some way or another is recorded in the table. Data names are taken from the system folder containing the (.cpp) source files.

This represents the structured reference in which defines its attributes, format, and relationships between data elements within the system to ensure clarity and proper usage. This organizes and acts as a guide for users to ensure consistency and accurateness on how data is defined and used.

Table 1: Data Dictionary

Data Name	Size	Data Type	Description
1. text	Varies from user input or read from txt file (up to 256 bytes)	string	Contains text data to display with type effect animation
2. speed	4 bytes	int	Delay (in ms) between each character from txtAnimation()
3. username	Varies from user input or read from txt file (up to 256 bytes)	string	Username entered by the user in LoginSys.cpp

4. password	varies (up to 256 bytes)	string	Password entered by the user
5. storedUser	varies (up to 256 bytes)	string	Username data read from credentials.txt
6. storedPass	varies (up to 256 bytes)	string	Password data read from credentials.txt
7. file	varies on file (1KB)	ifstream, ofstream	Contains the filename for read and write in login verification
8. IDcode	4 bytes	int	Admin code entered by the user
9. storedCode	4 bytes	int	Admin code read from credentials.txt
10. code	varies on file (1KB)	ifstream	Reads IDcode.txt to verify admin code
11. choice	4 bytes	int	User input choice in login menu
12. DATA_FILE	varies (up to 256 bytes)	string	File name containing current government posts (government_posts.txt) in mainMenu.cpp
13. ARCHIVE_FILE	varies (up to 256 bytes)	string	File name containing archived government posts (archive_posts.txt)
14. Achoice	4 bytes	int	User input in Admin Main Menu
15. Lchoice	4 bytes	int	User input in regular Main Menu
16. choice	4 bytes	int	Contains menu and filter selections
17. name	varies (up to 256 bytes)	string	Name of the government project post

18. status	varies (up to 256 bytes)	string	Current project status
19. feedback	varies (up to 256 bytes)	string	Official remarks on the project
20. startDate	varies (up to 256 bytes)	string	Project starting date
21. finishDate	varies (up to 256 bytes)	string	Project finish date
22. taxes	8 bytes	double	Amount of taxes allocated in the project
23. investment	8 bytes	double	Amount of funds used in the project
24. file	varies on file (1KB)	ifstream, ofstream	Contains filename to read and write
25. inFile	varies on file (1KB)	ifstream, ofstream	Reads data from txt file
26. outFile	varies on file (1KB)	ifstream, ofstream	Writes data from memory to file
27. archive	varies on file (1KB)	ifstream, ofstream	Reads and writes archived_posts.txt
28. readFile	varies on file (1KB)	ifstream, ofstream	Reads the txt data files
29. tempFile	varies on file (1KB)	ifstream, ofstream	Temporarily assigns the txt file for read and write
30. line	varies (up to 256 bytes)	string	Reads lines from data files
31. filter	4 bytes	int	Contains filter choice when sorting posts
32. Post	varies (up to maximum system memory)	struct	Structure containing project details
33. posts	24 bytes	vector<Post>	Dynamic list of Post structures loaded from data file

34. current	varies on file (1KB)	Post	Temporarily holds post data when loading txt file
35. block	varies (up to 256 bytes)	string	Aggregated text block representing a full post entry
36. content	varies (up to 256 bytes)	string	Contains the entire data from the file into a variable
37. comment	varies (up to 256 bytes)	string	User input containing comments to latest post
38. likes	4 bytes	int	Amount of likes of latest post
39. dislikes	4 bytes	int	Amount of dislikes of latest post
40. likePos	8 bytes	size_t	Position of "Likes: " field in the post file
41. dislikePos	8 bytes	size_t	Position of "Dislikes: " field in the post file
42. likeEnd	8 bytes	size_t	Ending position of "Likes: " line in the post file
43. dislikeEnd	8 bytes	size_t	Ending position of "Dislikes: " line in the post file
44. postBlock	varies (up to 256 bytes)	string	Stores text of a post when deleting or archiving
45. currentTitle	varies (up to 256 bytes)	string	Stores project text when deleting or archiving
46. found	1 byte	bool	Indicates when target post is available for deletion
47. title	varies (up to 256 bytes)	string	Stores project title entered for deletion or

			archival
48. now	8 bytes	time_t	Stores current system date and time
49. ltm	8 bytes	tm*	Pointer to local time structure for time format
50. buffer	40 bytes	char[]	Character array to store formatted timestamp
51. postedDate	varies (up to 256 bytes)	string	Project date creation

Code

In the figure below are the actual code snippets used for the system. Each part is separated into subsections and linked together using header files.

```
#include "LogInSys.h"
using namespace std;

int main(int argc, char** argv) {
    loginSys();
    return 0;
}
```

Figure 4. The Main cpp file

The figure shown above presents the main C++ file of the system. This file contains the function “loginSys()”, which serves as the core function of the system. Within this core function, several sub functions are called to ensure the functionality and flow of the system.

```
#ifndef DESIGN_H
#define DESIGN_H

#include <string>
using namespace std;

void ms250();
void ms500();
void s1();
void s2();
void Loading();
void txtAnimation(string text, int speed);

#endif
```

Figure 5. An Header file named “Design.h”

The figure shown above presents the header file that contains several functions designed to make the system more interactive. By utilizing libraries such as <chrono> and <thread>, these functions enable features like output delays and text animations, enhancing the overall user experience in the console interface. The main code snippets used in these functions is the combination of thread and chrono libraries. This ensures that, when the system runs and the function is called, the display of output is delayed before proceeding to the next line or code.

```

#ifndef LOGINSYS_H
#define LOGINSYS_H

#include <string>
using namespace std;

int LoginSys();
void registration();
bool checkUser(string user);
bool login();
bool adminLogin();
bool codeInter();
#endif

```

Figure 6. An Header file named "LogInSys.h"

The figure shown above presents the header file where the "loginSys()" function is declared. It also contains other functions that handle key features of the login system, such as user registration, administrator login, and regular user login. Additionally, this header file includes the verification features of the system, such as checking the user's entered username and validating the administrator's credentials through a code. This set of functions is responsible for managing user access and ensuring that the users can enter and interact with the system's contents. Furthermore, these functions utilize the <fstream> library to access external text files, allowing the system to write, read, and store user credentials.

```

#ifndef MAINMENU_H
#define MAINMENU_H

#include <string>
#include <vector>
using namespace std;
// ===== FUNCTION PROTOTYPES =====
void adminMenus();
void LocalMenu();
void createPost();
void editPost();
void addComment();
void addReaction();
void deletePosts();
string getTimeStamp();

```

```

struct Post {
    string name;
    string postedDate;
    string startDate;
    string finishDate;
    string status;
    double taxes;
    double investment;
    string feedback;
    string content;
};
vector<Post> LoadPosts();
// ===== FILES USED =====
extern string DATA_FILE;
extern string ARCHIVE_FILE;

#endif

```

Figure 7. An header file named “mainMenu.h”

```

#ifndef VIEWING_H
#define VIEWING_H

void view();
void viewPosts();
void filters();
void viewArchive();
void viewToDel();

#endif

```

Figure 8. An header file named “Viewing.h”

Finally, the figure shown above represents the main functionality of the entire system. These two header files are responsible for displaying, creating, editing, and deleting posts within the system, as well as additional features such as adding comments and reacting to posted information. The availability of these functions depends on the user’s login path, whether as an administrator or a regular user. The headers also include string data for accessing external text files, enabling the system to store, read, delete, and write information. Overall, these figures represent the core functions of the system, handling both information display and content management.

Results and Discussion

Presented in this section are the results and discussions based on the system's performance. This section aims to determine whether the objectives of the project have been achieved.

```
Welcome to the Government Management System!!|
```

```
What would you like to do?
```

```
(1) Register
```

```
(2) Login
```

```
(3) Admin Log-in
```

```
(4) Exit
```

```
Please input a number: |
```

```
(4) Exit
```

```
Please input a number: 5
```

```
Invalid input, Select between 1 and 4!|
```

```
(4) Exit
```

```
Please input a number: a
```

```
Invalid input, Please enter a number!|
```

At the start of the system, a greeting message is displayed to welcome the user. The user is then prompted to choose whether to register, log in, or log in as an administrator. With this Additionally, the system also includes an error checking feature that detects if the user enters an invalid input, such as letters instead of numbers or a number that is outside the given range. This is implemented throughout the entire system to ensure smooth and reliable functionality.

```
Welcome to the Registration Menu!!
```

```
Please enter your username: Group6Acc
```

```
Please enter your password: Group6Acc|
```

```
Registration is successful, Thank you for registering!
```

In the registration interface, the system prompts the user to create a username and password that will be used for logging into the system. The registered account can be used to access either logging in as an administrator or as a regular user.

```
Welcome to the Registration Menu!!  
Please enter your username: Group6Account  
Username too long! Please use up to 12 characters only.  
|
```

```
Welcome to the Registration Menu!!  
Please enter your username: Group6Acc  
Username has already been taken!!  
|
```

Additionally, certain limitations are applied when entering a username, it must be unique and can contain a maximum of 12 characters. These constraints are implemented to prevent any potential system errors and ensure smooth functionality of the system.

```
Thank you for using our system!!  
-----  
Process exited after 8.954 seconds with return value 0
```

An appreciation message is then displayed if the user chose to exit the system.

```
Welcome to the Log-in Menu!!  
Please enter your username: Group6Accs  
Enter your password: Group6Accs|
```

```
Invalid username or password! Please try again.|
```

```
Welcome to the Main Menu!!  
(1) View All Posts  
(2) Add Comment to Latest Post  
(3) React to Latest Post (Like/Dislike)  
(4) View Archives  
(5) Log-out  
Enter choice: |
```

In the log in interface, the system prompts the user to enter their registered username and password. If an invalid account is entered, the system notifies the user and returns to the main interface. However, if the username and password is correct, the system proceeds to the main menu for regular users. This menu allows the user to view all posted projects, add comments and reactions to the latest post, and access archived projects.

The information below is only an example to show the result of the system.

```
All Existing Projects:

Project: Bing Bing
Posted Date: 2025-11-03 22:14:04
Start Date: 1234-12-12
Finish Date: 1234-12-12
Status: Ongoing
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: qwerty
Likes: 0
Dislikes: 1
Comments:
Comment: Goods
-----

(0) Return to Main Menu   (1) Filters
Enter your choice: |
```

In the “View all posts” section, the system displays all existing projects along with detailed information for each project. This includes the project name, posting date, project start date, expected completion date, project status, applicable taxes, invested amount, official project description, current reactions (limited to like and dislike), and the latest comments related to the project. Additionally, there’s also a filtering feature where users can filter by newest to oldest, oldest to newest, and invested amounts.

```
Enter your ID Code: 20250001|
```

```
Welcome to the Admin Main Menu!!
(1) Create Project Post
(2) View All Posts
(3) Add Comment to Latest Post
(4) React to Latest Post (Like/Dislike)
(5) View Archives
(6) Delete Posts
(7) Log-out
Enter choice: |
```

In the admin log in interface, the process is similar to that of a regular user but includes an additional ID code used to verify the administrator’s credential. These ID codes are exclusively assigned to administrators. This verification process is implemented to maintain system organization and prevent unauthorized tampering with information. Once the credentials are verified, the system directs the administrator to the admin menu, where they can create and delete project posts.

```
Create a new project post!
Project Name: Bridge Construction
Taxes Used (P): 123456
Amount Invested (P): 123456
Status (Ongoing/Completed): Ongoing
Official Start Date (YYYY-MM-DD): 1234/12/12
Official Finish Date (YYYY-MM-DD): 1234/12/12
Official Feedback: Manggahan brigde
Project post saved successfully|
```

```
All Existing Projects:
Project: Bing Bing
Posted Date: 2025-11-03 22:14:04
Start Date: 1234-12-12
Finish Date: 1234-12-12
Status: Ongoing
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: qwerty
Likes: 0
Dislikes: 1
Comments:
Comment: Goods
-----
```

```
Project: Bridge Construction
Posted Date: 2025-11-08 20:46:57
Start Date: 1234/12/12
Finish Date: 1234/12/12
Status: Ongoing
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: Manggahan brigde
Likes: 0
Dislikes: 0
Comments:
-----
```

```
(0) Return to Main Menu   (1) Filters   (2) Edit post
Enter your choice: |
```

```
Select a post to edit:
1. Bing Bing (Ongoing)
2. Bridge Construction (Ongoing)

Enter post number to edit(Enter 0 to cancel): |
```

```
Editing post: Bridge Construction
Leave blank to keep existing value.

New Project Name (Bridge Construction):
New Taxes Used (current: 123456):
New Investment (current: 123456):
New Status (Ongoing): Completed
New Start Date (1234/12/12):
New Finish Date (1234/12/12):
New Feedback (Manggahan brigde):

Post updated successfully!!
|
```

```

All Existing Projects:
Project: Bing Bing
Posted Date: 2025-11-03 22:14:04
Start Date: 1234-12-12
Finish Date: 1234-12-12
Status: Ongoing
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: qwerty
Likes: 0
Dislikes: 0
Comments:
-----
Project: Bridge Construction
Posted Date: 2025-11-08 20:46:57
Start Date: 1234/12/12
Finish Date: 1234/12/12
Status: Completed
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: Manggahan brigde
Likes: 0
Dislikes: 0
Comments:
-----
(0) Return to Main Menu   (1) Filters   (2) Edit post
Enter your choice: |

```

In the “Create Project Post” section, the system will prompt the administrator to enter the required project information. Once all the necessary details are provided, the project is added to the list of existing projects. Additionally, administrators have the privilege to freely edit posts whenever modifications or updates are made to the project.

```

Input the EXACT Project Title to Delete and Transfer to the Archives
Bridge Construction
Post archived successfully!

```

```

Archived Posts:

Project: Bridge Construction
Posted Date: 2025-11-08 20:46:57
Start Date: 1234/12/12
Finish Date: 1234/12/12
Status: Completed
Taxes Used: P123456.00
Invested: P123456.00
Official Feedback: Manggahan brigde
Likes: 0
Dislikes: 0
Comments:
-----
Archived On: 2025-11-08 21:32:59
-----
Press enter to return to the menu...|

```

In the “Delete Post” section, the system first displays all existing projects. The user is then prompted to enter the exact project to be deleted. Once confirmed, the system removes the project from the “View all posts” section and archives it.

Conclusion

The development of the *Transparent Government Display System* successfully achieved its main objective to promote transparency by allowing citizens to view how government funds are utilized. The system effectively integrates essential features such as user and administrator login, project creation, updating, deletion, and archiving of outdated records. Through these functions, the project demonstrates how technology can strengthen accountability and public trust by providing accessible and organized information about government projects and expenditures.

All of the objectives that are established in this project were met, such as:

The **log-in system** ensures secure access and distinguishes between public users and administrative users with different privileges.

The **data management functions** (create, read, update, delete, and archive) allow proper record handling and ensure data integrity.

The **archiving feature** retains deleted project records for reference and transparency.

The **testing and evaluation phase** confirmed the system's reliability and accuracy in displaying project data.

Though there's still some features that may be added in the future. Through the making of this project, the students had some recommendation to fully maximize its potential, specifically they should at least:

1. Consist a Graphic User Interface system for better user experience and beginner friendly..
2. The Security System among accounts has vulnerabilities.
3. Consists of a larger data base as it is only a prototype, and more secure.
4. More user interaction and tabs for deeper information regarding the project. (Example : Lists people that are connected to the project)
5. Stayed Login feature
6. Saved Account for login feature

References

Justification Reference:

- *An act establishing a code of conduct and Ethical Standards for public officials and employees, to uphold the time-honored principles of public office being a public trust, granting incentives and rewards for exemplary service, enumerating prohibited acts and transactions and providing penalties for violations thereof and for other purposes. (1989). In Republic Act NO. 6713 [Legal document]. https://www.ombudsman.gov.ph/docs/republicacts/Republic_Act_No_6713.pdf*
- *Duterte, R. R. & Philippines. (2016). Executive Order No. 02, s. 2016. In MALACAÑANG PALACE. https://www.pdic.gov.ph/files/foi/Annex_A-EO_2_FOI.pdf*
- *Ombudsman of the Philippines. (1989). Republic Act No. 6713: Code of conduct and ethical standards for public officials and employees. https://www.ombudsman.gov.ph/docs/republicacts/Republic_Act_No_6713.pdf*
- *The 1987 Constitution of the Republic of the Philippines, Article XI: Accountability of Public Officers. (n.d.). Office of the Ombudsman (Philippines). https://www.ombudsman.gov.ph/docs/republicacts/Article_XI_1987_Philippine_Constitution.pdf*

Other research reference:

- *Abhishek, A., Rao, S., & Sinha, R. (2022). Towards devising a fund management system using blockchain. arXiv preprint arXiv:2211.03613. <https://arxiv.org/abs/2211.03613>*
- *Farazmand, A., Kim, S., & Rafiq, A. (2022). Corruption, lack of transparency and the misuse of public funds in times of crisis. *Public Organization Review*, 22(4), 901–915. <https://doi.org/10.1007/s11115-022-00651-8>*
- *Transparency International. (2024). Corruption Perceptions Index 2024 – Philippines. Retrieved from <https://www.transparency.org/en/countries/philippines>*

- Zijderwijk, A., Al Mushayt, O., & Janssen, M. (2025). *TAPAS: A pattern-based approach to assessing government transparency*. *arXiv preprint arXiv:2505.16413*.
<https://arxiv.org/abs/2505.16413>

Reference for code used:

- C++ file operation snippets. (n.d.). Gist. <https://gist.github.com/Gibgezr/0f403ee93a661bb01bbd>
- Filch, P. (2014, April 4). *Using exclamation marks "!" in c*. Stack Overflow.
<https://stackoverflow.com/questions/22855483/using-exclamation-marks-in-c>
- GeeksforGeeks. (2025, July 23). *Difference Between int and size_t in C++*. GeeksforGeeks.
https://www.geeksforgeeks.org/cpp/difference-between-int-and-size_t-in-cpp/
- GeeksforGeeks. (2025, October 28). *File handling in C++*. GeeksforGeeks.
<https://www.geeksforgeeks.org/cpp/file-handling-c-classes/>
- GeeksforGeeks. (2025a, May 21). *size_t data type in C*. GeeksforGeeks.
https://www.geeksforgeeks.org/c/size_t-data-type-c-language/
- GeeksforGeeks. (2025b, July 23). *std::string::replace() in C++*. GeeksforGeeks.
<https://www.geeksforgeeks.org/cpp/std-string-replace-in-cpp/>
- GeeksforGeeks. (2025b, September 19). *substr() in C++*. GeeksforGeeks.
<https://www.geeksforgeeks.org/cpp/substring-in-cpp/>
- GeeksforGeeks. (2024, December 27). *Vector push_back() in C++ STL*. GeeksforGeeks.
<https://www.geeksforgeeks.org/cpp/vector-push-back-cpp-stl/>
- Katie. (2013, May 3). *Current date and time as string*. Stack Overflow.
<https://stackoverflow.com/questions/16357999/current-date-and-time-as-string>
- *Understanding 'ios::trunc' Mode | File Streams | StudyPlan.dev*. (2023, June 22).
<https://www.studyplan.dev/pro-cpp/file-streams/q/understanding-ios-trunc>

- User. (n.d.). *What is the difference among ios::app, out, and trunc in c++?* Stack Overflow.
[https://stackoverflow.com/questions/48085781/what-is-the-difference-among-iosapp-out-and-trunc-i
n-c](https://stackoverflow.com/questions/48085781/what-is-the-difference-among-iosapp-out-and-trunc-in-c)
- W3Schools.com. (n.d.). https://www.w3schools.com/cpp/cpp_exceptions.asp
- W3Schools.com. (n.d.). https://www.w3schools.com/cpp/cpp_vectors.asp

Appendices

Appendix A: "Design.cpp"

```
#include "Design.h"

#include <iostream>
#include <string>
#include <cstdlib>
#include <chrono>
#include <thread>
using namespace std;

// Aesthetic Purposes =====
void ms250(){ // Delay for 250 millisecond before continuing
    this_thread::sleep_for(chrono::milliseconds(250));
}
void ms500(){ // Delay for 500 millisecond before continuing
    this_thread::sleep_for(chrono::milliseconds(500));
}
void s1(){ // Delay for 1 second before continuing
    this_thread::sleep_for(chrono::seconds(1));
}
void s2(){ // Delay for 2 second before continuing
    this_thread::sleep_for(chrono::seconds(2));
}
void loading(){ // Loading animation
    for(int a = 0; a < 2; a++){
        cout << "Loading"; ms250(); cout << "."; ms250(); cout <<
        "."; ms250();
        cout << "."; ms250(); system("cls"); ms250();
    }
}
void txtAnimation(string text, int speed){ // Typing effect on texts
    for(char c : text){
        cout << c << flush;
        this_thread::sleep_for(chrono::milliseconds(speed));
    }
}
```

Appendix B: "LogInSys.cpp"

```
#include "LogInSys.h"
#include "Design.h"
#include "mainMenu.h"

#include <iostream>
#include <fstream>
#include <string>
#include <limits>
#include <cstdlib>
using namespace std;

void registration(){ // Registration Definition
    const int maxLength = 12; string username, password;
    cout << "Welcome to the Registration Menu!!" << endl; s1();
    cout << "Please enter your username: "; cin >> username;
    if(username.length() > maxLength){
        cout << "Username too long! Please use up to 12 characters
only." << endl; s2(); return;
    }
    if(checkUser(username)){ // Calls the function to check if the
username is already taken
        cout << "Username has already been taken!!" << endl; s2();
return;
    }
    cout << "Please enter your password: "; cin >> password;
    ofstream file("credentials.txt", ios::app);
    if(file.is_open()){
        file << username << ' ' << password << endl;
system("cls"); loading();
        txtAnimation("Registration is successful, Thank you for
registering!", 10);
        cout << endl; file.close(); s2();
    }
}

bool checkUser(string user){ // Checking Username Definition
    string storedUser, storedPass;
    ifstream file("credentials.txt");
```

```

        if(file.is_open()){
            while(file >> storedUser >> storedPass){
                if(storedUser == user){
                    file.close(); return true;
                }
            }
            file.close();
        }
        return false;
    }
}

bool login(){ // Login Definition
    string username, password, storedUser, storedPass;
    cout << "Welcome to the Log-in Menu!!" << endl; ms500();
    cout << "Please enter your username: "; cin >> username;
    cout << "Enter your password: "; cin >> password;
    ifstream file("credentials.txt");
    if(file.is_open()){
        while(file >> storedUser >> storedPass){
            if(storedUser == username && storedPass ==
password){
                system("cls"); loading(); txtAnimation("Log-in
successful!", 10);
                file.close(); s2(); return true;
            }
        }
        file.close(); system("cls"); loading();
        txtAnimation("Invalid username or password! Please try
again.", 10); s2();
        return false;
    }
}

bool adminLogin(){ // Admin Log-in Definition
    string username, password, storedUser, storedPass;
    cout << "Welcome to the Admin Log-in Menu!!" << endl; ms500();
    cout << "Please enter your username: "; cin >> username;
    cout << "Enter your password: "; cin >> password;
    ifstream file("credentials.txt");
    if(file.is_open()){
        while(file >> storedUser >> storedPass){

```

```

        if(storedUser == username && storedPass ==
password){
            system("cls"); loading(); txtAnimation("Account
identified!", 10);
            file.close(); s2(); return true;
        }
    }
    file.close(); system("cls"); loading();
    txtAnimation("Invauid username or password! Please try
again.", 10); s2();
    return false;
}
}
bool codeInter(){
    int IDcode, storedCode; cout << "Enter your ID Code: ";
    cin >> IDcode; ifstream code("IDcode.txt");
    if(code.is_open()){
        while(code >> storedCode){
            if(storedCode == IDcode){
                system("cls"); loading();
                txtAnimation("Admin code identified, Admin
log-in successful!", 10);
                code.close(); s2(); return true;
            }
        }
        code.close(); system("cls"); loading();
        txtAnimation("Invalid ID code, Log-in terminated!", 10);
s2();
        return false;
    }
}
int loginSys(){ // Menu definition (Loops)
    int choice;
    txtAnimation("Welcome to the Government Management System!!",
10); s2();
    while(choice != 4){
        system("cls"); cout << "What would you like to do?" <<
endl;
        cout << "(1) Register\n(2) Login\n(3) Admin Log-in\n(4)

```

```

Exit" << endl;
    cout << "Please input a number: ";
    if(!(cin >> choice)){ // User Input Symbols or Numbers
        cin.clear(); // Clears input if error
        cin.ignore(10000, '\n'); // Ignores Invalid
        txtAnimation("Invalid input, Please enter a
number!", 10); s2(); continue;
    }
    if(choice == 1){
        system("cls"); loading(); registration();
    }
    else if(choice == 2){
        system("cls"); loading();
        if(login()){
            system("cls"); localMenu();
        }
    }
    else if(choice == 3){
        system("cls"); loading();
        if(adminLogin()){
            system("cls");
            if(codeInter()){
                system("cls"); loading(); adminMenus();
            }
        }
    }
    else if(choice == 4){
        system("cls"); txtAnimation("Thank you for using our
system!!", 10); s2();
    }
    else{
        txtAnimation("Invalid input, Select between 1 and
4!", 10); s2();
    }
}
return 0;
}

```

Appendix C: "mainMenu.cpp"

```
#include "Design.h"
#include "mainMenu.h"
#include "Viewing.h"

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <ctime>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <sstream>
#include <limits>
#include <vector>
using namespace std;
// ===== FILE USED =====
string DATA_FILE = "government_posts.txt";
string ARCHIVE_FILE = "archived_posts.txt";
// ===== ADMIN MENU =====
void adminMenus() {
    int Achoice;
    while(Achoice != 7){
        system("cls"); cout << "Welcome to the Admin Main Menu!!\n";
        cout << "(1) Create Project Post\n(2) View All Posts\n";
        cout << "(3) Add Comment to Latest Post\n";
        cout << "(4) React to Latest Post (Like/Dislike)\n";
        cout << "(5) View Archives\n(6) Delete Posts\n(7) Log-out\n";
        cout << "Enter choice: ";
        if(!(cin >> Achoice)){
            cin.clear(); cin.ignore(10000, '\n');
            txtAnimation("Invalid input, Please enter a number!", 10);
s2(); continue;
        }
        if(Achoice == 1){
            system("cls"); loading(); createPost();
        }
    }
}
```

```

        else if(Achoice == 2){
            system("cls"); loading(); viewPosts();
        }
        else if(Achoice == 3){
            system("cls"); loading(); addComment();
        }
        else if(Achoice == 4){
            system("cls"); loading(); addReaction();
        }
        else if(Achoice == 5){
            system("cls"); loading(); viewArchive();
        }
        else if(Achoice == 6){
            system("cls"); loading(); deletePosts();
        }
        else if(Achoice == 7){
            txtAnimation("Logging out...", 10); s2();
        }
        else{
            txtAnimation("Invalid input! Select a number between
1 and 7.", 10); s2();
        }
    }
}
// ===== LOCAL MENU =====
void localMenu(){
    int Lchoice;
    while(Lchoice != 5){
        system("cls"); cout << "Welcome to the Main Menu!!\n";
        cout << "(1) View All Posts\n(2) Add Comment to Latest
Post\n";
        cout << "(3) React to Latest Post (Like/Dislike)\n";
        cout << "(4) View Archives\n(5) Log-out\n"; cout << "Enter
choice: ";
        if(!(cin >> Lchoice)){
            cin.clear(); cin.ignore(10000, '\n');
            txtAnimation("Invalid input, Please enter a number!", 10);
s2(); continue;
        }
    }
}

```

```

        if(Lchoice == 1){
            system("cls"); loading(); view();
        }
        else if(Lchoice == 2){
            system("cls"); loading(); addComment();
        }
        else if(Lchoice == 3){
            system("cls"); loading(); addReaction();
        }
        else if(Lchoice == 4){
            system("cls"); loading(); viewArchive();
        }
        else if(Lchoice == 5){
            txtAnimation("Logging out...", 10); s2();
        }
        else{
            txtAnimation("Invalid input! Select a number between
1 and 5.", 10); s2();
        }
    }
}

// ===== CREATE POST =====
void createPost() {
    string name, status, feedback, startDate, finishDate;
    double taxes, investment; cout << "Create a new project post!\n";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Project Name: "; getline(cin, name);
    cout << "Taxes Used (P): "; cin >> taxes;
    cout << "Amount Invested (P): "; cin >> investment;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Status (Ongoing/Completed): "; getline(cin, status);
    cout << "Official Start Date (YYYY-MM-DD): "; getline(cin,
startDate);
    cout << "Official Finish Date (YYYY-MM-DD): "; getline(cin,
finishDate);
    cout << "Official Feedback: "; getline(cin, feedback);
    ofstream file(DATA_FILE, ios::app);
    if (!file) {
        txtAnimation("There's an error in opening the file!!", 10);
    }
}

```

```

        return;
    }
    file << fixed << setprecision(2);
    file << "Project: " << name << "\n";
    file << "Posted Date: " << getTimestamp() << "\n";
    file << "Start Date: " << startDate << "\n";
    file << "Finish Date: " << finishDate << "\n";
    file << "Status: " << status << "\n";
    file << "Taxes Used: P" << taxes << "\n";
    file << "Invested: P" << investment << "\n";
    file << "Official Feedback: " << feedback << "\n";
    file << "Likes: 0\n"; file << "Dislikes: 0\n";
    file << "Comments:\n";
    file << "-----\n\n";
    file.close(); txtAnimation("Project post saved successfully!",
10); s2();
}
// ==== READS INFORMATION'S POST FOR FILTERING AND EDIT POST ====
vector<Post> loadPosts() {
    ifstream file(DATA_FILE); vector<Post> posts;
    if (!file) {
        return posts;
    }
    string line, block; Post current;
    while (getline(file, line)) {
        block += line + "\n";
        if (line.find("Project: ") != string::npos)
            current.name = line.substr(9);
        else if (line.find("Posted Date: ") != string::npos)
            current.postedDate = line.substr(13);
        else if (line.find("Start Date: ") != string::npos)
            current.startDate = line.substr(12);
        else if (line.find("Finish Date: ") != string::npos)
            current.finishDate = line.substr(13);
        else if (line.find("Status: ") != string::npos)
            current.status = line.substr(8);
        else if (line.find("Taxes Used: P") != string::npos)
            current.taxes = stod(line.substr(13));
        else if (line.find("Invested: P") != string::npos)

```

```

        current.investment = stod(line.substr(11));
        else if (line.find("Official Feedback: ") != string::npos)
            current.feedback = line.substr(19);
        else if (line.find("-----") !=
string::npos) {
            current.content = block; posts.push_back(current);
block.clear();
        }
    }
    file.close(); s2(); return posts;
}
// ===== EDIT POST =====
void editPost() {
    vector<Post> posts = loadPosts(); system("cls"); loading();
    if (posts.empty()) {
        txtAnimation("No posts found!", 10); cout << "\n"; return;
    }
    // Display posts with numbers
    cout << "Select a post to edit: \n";
    for (size_t i = 0; i < posts.size(); ++i) {
        cout << i + 1 << ". " << posts[i].name << " (" <<
posts[i].status << ")\n";
    }
    int choice; cout << "\nEnter post number to edit(Enter 0 to
cancel): ";
    cin >> choice; cin.ignore();
    if (choice == 0 || choice > posts.size()) return;
    Post &p = posts[choice - 1];
    // Show current info
    system("cls");
    cout << "Editing post: " << p.name << "\n";
    cout << "Leave blank to keep existing value.\n\n"; string input;
    cout << "New Project Name (" << p.name << "): "; getline(cin,
input);
    if (!input.empty()) p.name = input;
    cout << "New Taxes Used (current: " << p.taxes << "): ";
getline(cin, input);
    if (!input.empty()) p.taxes = stod(input);
    cout << "New Investment (current: " << p.investment << "): ";

```

```

getline(cin, input);
    if (!input.empty()) p.investment = stod(input);
    cout << "New Status (" << p.status << "): "; getline(cin, input);
    if (!input.empty()) p.status = input;
    cout << "New Start Date (" << p.startDate << "): "; getline(cin,
input);
    if (!input.empty()) p.startDate = input;
    cout << "New Finish Date (" << p.finishDate << "): ";
getline(cin, input);
    if (!input.empty()) p.finishDate = input;
    cout << "New Feedback (" << p.feedback << "): "; getline(cin,
input);
    if (!input.empty()) p.feedback = input;
    // Rewrite file
    ofstream file(DATA_FILE, ios::trunc);
    for (const auto &post : posts) {
        file << fixed << setprecision(2);
        file << "Project: " << post.name << "\n";
        file << "Posted Date: " << post.postedDate << "\n";
        file << "Start Date: " << post.startDate << "\n";
        file << "Finish Date: " << post.finishDate << "\n";
        file << "Status: " << post.status << "\n";
        file << "Taxes Used: P" << post.taxes << "\n";
        file << "Invested: P" << post.investment << "\n";
        file << "Official Feedback: " << post.feedback << "\n";
        file << "Likes: 0\n"; file << "Dislikes: 0\n";
        file << "Comments:\n"; file <<
"-----\n";
    }
    file.close();
    cout << "\n"; txtAnimation("Post updated successfully!!", 10);
cout << "\n"; s2();
}
// ===== ADD COMMENT TO LATEST POST =====
void addComment() {
    ifstream inFile(DATA_FILE);
    if (!inFile) {
        cout << "No posts available.\n"; return;
    }
}

```

```

        string content((istreambuf_iterator<char>(inFile)),
istreambuf_iterator<char>());
        inFile.close(); size_t pos =
content.rfind("-----");
        if (pos == string::npos){
            txtAnimation("No posts found!", 10); cout << "\n"; return;
        }
        string comment;
        cout << "Enter your comment: ";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
getline(cin, comment);
        content.insert(pos, "Comment: " + comment + "\n");
        ofstream outFile(DATA_FILE);
        outFile << content;
        outFile.close();
        cout << "\n"; txtAnimation("Comment added successfully", 10);
cout << "\n"; s2();
    }
// ===== ADD REACTION =====
void addReaction() {
    // Open the post file to read
    ifstream inFile(DATA_FILE);
    if (!inFile) {
        txtAnimation("No posts found!", 10); cout << "\n"; return;
    }
    // Read the entire file content into a string
    string content((istreambuf_iterator<char>(inFile)),
istreambuf_iterator<char>());
    inFile.close();
    // Find the last occurrence of "Likes:" and "Dislikes:"
    size_t likePos = content.rfind("Likes:");
    size_t dislikePos = content.rfind("Dislikes:");
    // If not found, stop
    if (likePos == string::npos || dislikePos == string::npos) {
        txtAnimation("No posts with reaction found!", 10); cout <<
"\n"; return;
    }
    // Get current like and dislike counts
    int likes = 0, dislikes = 0;

```

```

        try {
            likes = stoi(content.substr(likePos + 6)); // extract number
after "Likes:"
            dislikes = stoi(content.substr(dislikePos + 9)); // extract
number after "Dislikes:"
        } catch (...) {
            txtAnimation("Error reading like/dislike counts!", 10); cout
<< "\n"; return;
        }
        // Ask user for reaction
        int choice; cout << "React to the latest post/s:\n";
        cout << "(1) Like\n(2) Dislike\nEnter choice: "; cin >> choice;
        if (choice == 1) likes++; else if (choice == 2) dislikes++;
        else {
            txtAnimation("Invalid input!", 10); cout << "\n"; return;
        }
        // Replace the lines in the content string with updated counts
        size_t likeEnd = content.find('\n', likePos);
        size_t dislikeEnd = content.find('\n', dislikePos);
        content.replace(likePos, likeEnd - likePos, "Likes: " +
to_string(likes));
        content.replace(dislikePos, dislikeEnd - dislikePos, "Dislikes: "
+ to_string(dislikes));
        // Save updated content back to the file
        ofstream outFile(DATA_FILE);
        if (!outFile) {
            txtAnimation("There's an error saving reactions!", 10); cout
<< "\n"; return;
        }
        outFile << content; outFile.close();
        txtAnimation("Reaction recorded successfully!!", 10); cout <<
"\n"; s2();
    }
    // ===== DELETE ALL POSTS =====
    void deletePosts(){
        txtAnimation("Here are the Active Current Projects:", 10); cout
<< "\n";
        viewToDel(); string title, line; bool found = false;
        cout << "\nInput the EXACT Project Title to Delete and Transfer

```

```

to the Archives\n";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    getline(cin, title); ifstream readfile(DATA_FILE); // READS POST
    ofstream tempFile("TemporaryPosts.txt"); // TEMPORARY SAVE FILE
    ofstream archive(ARCHIVE_FILE, ios::app); // STORES DELETED POSTS
    if (!readfile) {
        txtAnimation("No posts found", 10); cout << "\n"; return;
    }
    string postBlock = ""; string currentTitle = "";
    while (getline(readfile, line)) {
        // Detect start of a new post
        if (line.find("Project: ") != string::npos) {
            // If we already have a previous post, check if it's the
one to archive
            if (!postBlock.empty()) {
                if (currentTitle == title) {
                    found = true; archive << postBlock;
                    archive << "Archived On: " << getTimestamp() <<
endl;

                    archive <<
"-----" << endl;
                }
                else {
                    tempFile << postBlock;
                }
            }
            // Start capturing the new post
            currentTitle = line.substr(line.find(":") + 2); postBlock
= line + "\n";
        }
        else {
            postBlock += line + "\n";
        }
    }
    // Handle the last post in the file
    if (!postBlock.empty()) {
        if (currentTitle == title) {
            found = true; archive << postBlock;
            archive << "Archived On: " << getTimestamp() << endl;

```

```

        archive << "-----" <<
endl;
    }
    else {
        tempFile << postBlock;
    }
}
readFile.close(); tempFile.close(); archive.close();
remove("government_posts.txt"); rename("TemporaryPosts.txt",
"government_posts.txt");
if(found){
    txtAnimation("Post archived successfully!", 10); cout << "\n";
s2();
}
else{
    txtAnimation("Post not found. Make sure your title matches
exactly!!", 10);
    cout << "\n"; s2();
}
}
// ===== TIME STAMPS =====
string getTimeStamp(){
    time_t now = time(0);
    tm *ltm = localtime(&now);
    char buffer[40];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", ltm);
    return string(buffer);
}

```

Appendix D: "Viewing.cpp"

```
#include "Viewing.h"
#include "mainMenu.h"
#include "Design.h"

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <sstream>
#include <limits>

// ===== VIEW POST(FOR LOCAL USERS) =====
void view() {
    ifstream file(DATA_FILE);
    if (!file.is_open()) {
        txtAnimation("No post file found.", 10); cout << "\n";
        return;
        txtAnimation("Press enter to return to the menu...", 10);
    }
    cin.get();
    file.seekg(0, ios::end); // Move file pointer to the end to check
    if it's empty
    if (file.tellg() == 0) {
        txtAnimation("No post found.", 10); cout << "\n";
        txtAnimation("Press enter to return to the menu...", 10);
    }
    cin.get();
    file.close(); return;
    }
    file.seekg(0, ios::beg); // Reset pointer to the beginning
    before reading
    txtAnimation("All Existing Projects:", 10);
    s1(); cout << "\n\n"; string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
}
```

```

    }
    file.close();
    int filter; //Filter out by date or investment amount
    do {
        cout << "(0) Return to Main Menu    (1) Filters\n";
        cout << "Enter your choice: "; cin >> filter; cin.ignore();
        switch (filter) {
            case 0: return;
            case 1: filters(); break;
            default: txtAnimation("Invalid choice. Please try
again!", 10); cout << "\n";
        }
    } while (filter != 1);
}
// ===== VIEW POSTS(FOR ADMINS) =====
void viewPosts() {
    ifstream file(DATA_FILE);
    if (!file.is_open()) {
        txtAnimation("No post file found.", 10); cout << "\n";
        return;
        txtAnimation("Press enter to return to the menu...", 10);
        cin.get();
    }
    file.seekg(0, ios::end); // Move file pointer to the end to check
if it's empty
    if (file.tellg() == 0) {
        txtAnimation("No post found.", 10); cout << "\n";
        txtAnimation("Press enter to return to the menu...", 10);
        cin.get();
        file.close(); return;
    }
    file.seekg(0, ios::beg); // Reset pointer to the beginning
before reading
    txtAnimation("All Existing Projects:", 10);
    s1(); cout << "\n"; string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
    file.close();
}

```

```

    int filter; //Filter out by date or investment amount
    do {
        cout << "(0) Return to Main Menu    (1) Filters    (2) Edit
post\n";
        cout << "Enter your choice: "; cin >> filter; cin.ignore();
        switch (filter) {
            case 0: return;
            case 1: filters(); break;
            case 2: editPost(); break;
            default: txtAnimation("Invalid choice. Please try
again!", 10); cout << "\n";
        }
    } while (filter != 1);
}
// ===== FILTERING MENU FUNCTION =====
void filters() {
    int choice; vector<Post> posts = loadPosts();
    while (choice != 0){
        system("cls");
        cout << "Filter by:\n";
        cout << "(1) Newest\n(2) Oldest\n";
        cout << "(3) Investment amount\n(4) Main Menu\n";
        cout << "Enter choice: "; cin >> choice; cin.ignore();
        //Filtering process
        switch(choice){
            case 1:
                sort(posts.begin(), posts.end(), [](const Post
&a, const Post &b) {
                    return a.postedDate > b.postedDate;
                }); break;
            case 2:
                sort(posts.begin(), posts.end(), [](const Post
&a, const Post &b) {
                    return a.postedDate < b.postedDate;
                }); break;
            case 3:
                sort(posts.begin(), posts.end(), [](const Post
&a, const Post &b) {
                    return a.investment > b.investment;

```

```

        }); break;
        case 4:
            return; break;
        default:
            cout << "\n"; txtAnimation("Invalid choice!",
10); cout << "\n";
            txtAnimation("Press enter to continue...", 10);
cin.get(); continue;
    }
    //Filtered Posts output
    system("cls"); cout << "Filtered results:\n\n";
    for (const auto &p : posts) {
        cout << p.content << endl;
    }
    txtAnimation("Press enter to return to filters...", 10);
cin.get();
    }
}
// ===== VIEW ARCHIVES =====
void viewArchive() {
    ifstream file(ARCHIVE_FILE);
    if (!file.is_open()) {
        txtAnimation("No archives available!", 10); cout << "\n";
return;
        txtAnimation("Press enter to return to the menu...", 10);
cin.get();
    }
    file.seekg(0, ios::end);
    if (file.tellg() == 0) {
        txtAnimation("No post found.", 10); cout << "\n";
        txtAnimation("Press enter to return to the menu...", 10);
cin.get();
        file.close(); return;
    }
    file.seekg(0, ios::beg); txtAnimation("Archived Posts:", 10);
s1();
    cout << "\n\n"; string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
}

```

```

    }
    file.close(); txtAnimation("Press enter to return to the
menu...", 10);
    cin.ignore(numeric_limits<streamsize>::max(), '\n'); cin.get();
}
// ===== VIEW POSTS FOR DELETION =====
void viewToDel(){
    ifstream file(DATA_FILE);
    if (!file.is_open()) {
        txtAnimation("No post file found.", 10); cout << "\n";
return;
        txtAnimation("Press enter to return to the menu...", 10);
cin.get();
    }
    file.seekg(0, ios::end); // Move file pointer to the end to check
if it's empty
    if (file.tellg() == 0) {
        txtAnimation("No post found.", 10); cout << "\n";
        txtAnimation("Press enter to return to the menu...", 10);
cin.get();
        file.close(); return;
    }
    file.seekg(0, ios::beg); // Reset pointer to the beginning
before reading
    txtAnimation("All Existing Projects:", 10);
    s1(); cout << "\n\n"; string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
    file.close();
}
}

```