Message Publishing System Based on MFC

Yulin Yang

School of Computer and Software, Jincheng College of Sichuan University, Chengdu 611731, Sichuan, China

Abstract: ***Messages are a primary vehicle for quickly obtaining fresh external information in daily life. Their main difference from what is commonly called news is their conciseness; they can be regarded as a type of news and therefore share news's timeliness. With the acceleration of modern life, timely sharing of the latest messages has become an important need—rapid message sharing in companies, schools, and other settings is also crucial. This paper introduces a message publishing system designed and implemented with MFC, discusses how its main functions are realized via MFC, and explains how JSON is used as an intermediary for data exchange.***

Keywords: MFC; Message publishing system; JSON.

1. KEY TECHNOLOGIES USED IN THE SOFTWARE

Among object-oriented programming languages, C++ is designed based on the C language. Although it is more complex compared to other languages, it offers the flexibility to accomplish almost everything other languages can do. For larger-scale software systems, it does not rely more frequently on the standard library.

MFC is a class library provided by the Visual Studio programming environment. It encapsulates many Windows functions and features into controls for our use, allowing us to program with various controls in a dialog-based manner. In this type of program design, the main task is the proper use of these controls. For the same application, the quality of control usage determines the difficulty of development and the readability of the program. Therefore, before using any control, one should carefully consider whether it is suitable for the intended purpose. In natural language processing, Yang et al. (2025) [1] proposed a novel GAN-based extractive text summarization approach combining transductive and reinforcement learning, while Xie and Chen (2025) [2] developed CoreViz, a context-aware reasoning engine for business intelligence dashboards. For system diagnostics, Zhu (2025) [3] introduced TraceLM for temporal root-cause analysis using contextual embedding models, and Zhang (2025) [4] presented CrossPlatformStack to enhance service availability across Meta platforms. In creative applications, Hu (2025) [5] created GenPlayAds for procedural generation of interactive 3D advertisements. Computer vision research includes Zheng et al. (2025) [6]'s DiffMesh framework for video-based human mesh recovery and Peng et al. (2024) [7]'s work on domain adaptation for human pose estimation. Healthcare analytics has seen contributions from Zhang et al. (2025) [8] in biomechanical anomaly detection and Chen et al. (2024) [9]'s Bimcv-R dataset for medical image retrieval. Additional NLP research by Yu et al. (2025) [10] explored transformer-based text summarization. Economic applications include Bi and Lian (2025) [11]'s study on AI in digital finance exports and Pal et al. (2025) [12]'s AI credit risk assessment system. Finally, Chen et al. (2023) [13] advanced medical imaging with self-supervised neuron segmentation.

JSON is a data-exchange format that is concise, clear, and highly readable, making it suitable for exchanging small amounts of data. In this application, JSON is primarily used to unify the data format for sending and receiving across various interfaces.

2. MAIN FUNCTIONALITY DESCRIPTION

Upon launching the software, the user first arrives at the main interface, where they can choose to log in or register. The home page also provides buttons for browsing messages by category and for composing new messages. Before logging in, users can still browse messages in each category and read the comments beneath them, but they cannot use the comment feature, nor can they edit or publish any messages. In the message directory under each category, users can see the publication time of each message, and messages are listed in chronological order. After logging in, users can compose and publish messages, enter the comment section under any message's detail page to reply, and, when viewing their own message's detail page, delete or modify that message. Likewise, they can delete or modify their own replies. Logged-in users can also access a personal information page to update their details at any time.

3. ENVIRONMENT PREPARATION AND CONFIGURATION

First, the server can be deployed on a virtual machine to isolate it from the client; alternatively, you can host it on a cloud server according to your needs to simulate a real C/S architecture. The advantages of this architecture mainly include: (1) high accuracy, (2) strong security, (3) good interactivity, and (4) fast interaction speed, making it a commonly used design today [3]. When starting the environment setup, first download and install the WampServer on the server side. After installation, the first launch may fail because the port is occupied; simply change the port number. In addition, if you want to access the configured WampServer from another network, you will be told that you lack permission—this is because WampServer by default only allows access from the local host. To change this, open the Apache folder, edit the httpd.conf file, delete “Deny from all”, and change "Allow from 127.0.0.1" to "Allow from all" [4]. Next, create the required tables in the SQL database according to the different needs that arise during software design. Finally, follow an online tutorial to configure the C++ server; inside its configuration package there is usually a sql.json file. In sql.json you must write stored procedures based on the tables you created, their key fields, and the database operation functions. When everything is ready, the server side effectively uses three servers: a web server, a C++ server, and a database server. The web server receives requests from the client and communicates with the C++ server. The C server receives data and requirements from the web server, runs business code, and performs various operations on the SQL tables. The database server executes the previously prepared functions according to the requests from the C++ server, operates on the table fields, and returns the data. After the environment is configured, use the JSON Tools utility to test whether database operations succeed. JSON is used as the intermediary for data exchange; all operation data and commands are converted into JSON format, acting like a unified interface. Later, even if additional client software is added, it only needs to convert its operation data into JSON format. If no exception codes or other errors are returned after testing, the basic environment setup is considered complete.

4. IMPLEMENTATION OF SPECIFIC FUNCTIONS

4.1 Password Encryption and JSON Data Exchange

For password encryption, MD5 can be used. MD5 employs a hash algorithm whose clever feature is irreversibility: even if someone obtains the encrypted data, they cannot derive the original password. In the system’s login module, passwords are encrypted with MD5 before being stored in the database, enhancing system security [5].

Since various operations on the data require converting it to JSON format first, writing your own conversion functions is cumbersome; you can use an existing JSON library. The data packaging code is as follows:

```
Json:: Value root;
Json:: Value item;
    item["ID_Card"] = id_card;
    item["RealName"] = realname;
    item["ID"] = id;
    item["Password"] = password;
    item["E_mail"] = e_mail;
    item["NickName"] = nickname;
    root["reqKey"] = "Register";
    root["input"] = item;
    std::string Itc = root.toStyledString(    );
    std::string itc = "jsons=" + Itc;
    itc = string_To_UTF8(itc);
```

After steps like the above, the data to be exchanged becomes JSON format, which is then sent to the corresponding server address via an HTTP tool:

```
CMyHttpTools myhttp;
autos = myhttp.OnOpenHttp(addr, out);
```

Finally, the server interacts with the database, which returns the requested data; the client then retrieves the requested data from the returned JSON using the following code:

```
get <1>(s) = UTF8_To_string (get <1>(s));
std:: string strValue = get <1>(s);
```

This achieves data exchange between the client and the database.

4.2 Login and Registration Module

When implementing this part with MFC, you can place Edit Control widgets on the login and registration interfaces to receive user input such as username and password, and then use the following code to retrieve the information entered in the control:

```
CString ID;
GetDlgItem (IDC_EDIT1) -> GetWindowText(ID);
```

After obtaining the requested information, it becomes apparent that the data type retrieved is CString, which is not very convenient for subsequent operations. Therefore, the CString data needs to be converted into the more manageable char * type; the conversion code is as follows:

```
char*id == new char[50];
strcpy(id, (char*)_bstr_t(ID));
```

This allows subsequent operations to proceed. If registration is required, the obtained user information must also be validated. This step is executed after the user clicks the Confirm Registration button. In MFC, you can add a response event to a Button Control, treating the button click as triggering a function. In fact, most other controls can also have response functions added; simply double-click the control in the MFC window interface to create and jump directly to its response function. After adding the registration button's response function, you can check the username or password to see whether they meet length and character requirements or whether any required fields are empty. This step can be implemented directly with C++ for the checks, or via regular expressions—the former is simpler, the latter more precise. Once the format is verified, encrypt the password with MD5, then package the validated registration information into JSON and send the operation to the server. Using functions in the SQL database, compare the registration information against existing user data; if no identical account exists, registration succeeds.

The design of the login function is similar to that of registration. First, an Edit Control is used to capture the user's input, after which the entered data are validated for any illegality. Finally, the provided information is checked against the existing records in the database; if both the username and password exist and belong to the same user, the login succeeds. During verification, note that the password stored in the user data is MD5-encrypted, so the password entered by the user must also be MD5-encrypted before comparison.

4.3 Message Catalog Module

Message categorization is very easy to implement: just add the corresponding category selection buttons on the software's home page and attach the appropriate response function to each button. The message directory interface is designed so that all categories share one interface; in the response function for each category's button, simply retrieve the corresponding messages from the database, pass them to the message interface, and set a flag to "tell" the message directory which category was selected. This achieves categorized browsing while eliminating a lot of redundant code.

In the message catalog interface, you can use the List Control, which is convenient for displaying and adding list items. Before displaying the news list retrieved from the database, the List Control is divided into multiple columns; the code for one of the columns is as follows:

```
m_newslist.InsertColumn(1,_T("News Title"), LVCFMT_LEFT, 100);
m_newslist.SetColumnWidth(1, wid * 3 / 5);
```

Column number 1 is set to three-fifths of the entire list by this line of code. Here, wid represents the total width of the list control, which you need to obtain in advance.

Next, insert the obtained message list into the list:

m_newslist.SetItemText(m, 1, Ntitle);

Finally, refresh the interface to see the message list displayed in the List Control.

4.4 Message Detail Page and Comment Module

In the message directory list interface, you can use a CPoint variable to get the mouse position and add a click event to open the message detail page. In the detail page, multiple Edit Control controls can be set to read-only to display the publisher's information and the message content. Read-only controls do not allow user input like normal edit boxes; after setting the read-only attribute, the edit box can only be used to display data.

In the message detail page interface, set a Button Control whose response function opens the comment interface. In the comment interface, you can use a group of Edit Control controls N to display comment information obtained from the database. If the last few comments on a page are empty, you can use the ShowWindow() function to hide the comment boxes, making the interface cleaner. The specific usage is as follows:

CButton m_udr1;
m_udr1.ShowWindow(FALSE);

This statement makes the control associated with variable m_udr1 invisible.

Another feature here is determining whether the message or comment was posted by the logged-in user. If so, the user is allowed to modify or delete the message and comments. Initially, the Disabled property of the modify and delete buttons is set to false, so the buttons are unusable at first. When entering the news, compare the logged-in user's ID with the message publisher's ID. If they match, use the EnableWindow() function to enable the button. The specific code is as follows:

m_updatenew.EnableWindow(TRUE);

This allows the logged-in user to modify or delete their own messages and comments via these buttons.

4.5 Posting Messages and Modifying Personal Information Module

The message posting feature is relatively simple in the system. First, set Edit Control controls to get the title and content of the message entered by the user. Then, set the send button's response function to package this data into JSON format and send it to the server. Finally, the server stores the data in the corresponding table in the database. Since the requirement mentions recording the posting time of each message, the SQL side will use a function to get the current time and store it together with the received data.

Modifying personal information is similar: package all modified information into JSON data to update it. The tricky part is what to do if some data is left blank because the user doesn't want to change it. The solution is to set an initial value for the sent data, using the logged-in user's current information as the default.

5. CONCLUSION

This paper describes how to use C++ and MFC to implement the various functions of a message publishing platform, focusing on the use of several common MFC controls and the roles of their properties. It also briefly outlines the entire design process of the C/S framework and its essential components, as well as how and why the JSON data format is employed for data exchange.

REFERENCES

[1] Yang, Jing, et al. "A generative adversarial network-based extractive text summarization using transductive and reinforcement learning." IEEE Access (2025).

[2] Xie, Minhui, and Shujian Chen. "CoreViz: Context-Aware Reasoning and Visualization Engine for Business Intelligence Dashboards." Authorea Preprints (2025).
[3] Zhu, Bingxin. "TraceLM: Temporal Root-Cause Analysis with Contextual Embedding Language Models." (2025).
[4] Zhang, Yuhan. "CrossPlatformStack: Enabling High Availability and Safe Deployment for Products Across Meta Services." (2025).
[5] Hu, Xiao. "GenPlayAds: Procedural Playable 3D Ad Creation via Generative Model." (2025).
[6] Zheng, Ce, et al. "Diffmesh: A motion-aware diffusion framework for human mesh recovery from videos." 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). IEEE, 2025.
[7] Peng, Qucheng, et al. "Exploiting Aggregation and Segregation of Representations for Domain Adaptive Human Pose Estimation." arXiv preprint arXiv:2412.20538 (2024).
[8] Zhang, Shengyuan, et al. "Research on machine learning-based anomaly detection techniques in biomechanical big data environments." Molecular & Cellular Biomechanics 22.3 (2025): 669-669.
[9] Chen, Yinda, et al. "Bimcv-r: A landmark dataset for 3d ct text-image retrieval." International Conference on Medical Image Computing and Computer-Assisted Intervention. Cham: Springer Nature Switzerland, 2024.
[10] Yu, Z., Sun, N., Wu, S., & Wang, Y. (2025, March). Research on Automatic Text Summarization Using Transformer and Pointer-Generator Networks. In 2025 4th International Symposium on Computer Applications and Information Technology (ISCAIT) (pp. 1601-1604). IEEE.
[11] Bi, Shuochen, and Yufan Lian. "Research on the Export Trade Path Mechanism of Digital Finance and High-tech Industries under AI Technology." (2025).
[12] Pal, P. et al. 2025. AI-Based Credit Risk Assessment and Intelligent Matching Mechanism in Supply Chain Finance. Journal of Theory and Practice in Economics and Management. 2, 3 (May 2025), 1–9.
[13] Chen, Yinda, et al. "Self-supervised neuron segmentation with multi-agent reinforcement learning." arXiv preprint arXiv:2310.04148 (2023).