CRUD Operation On WordPress Custom Post Type (CPT) From C# Over REST API

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Senior Professor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Computer Science.
Type of the Paper: Experimental Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.10408545
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Chakraborty, S. & Aithal, P. S. (2023). CRUD Operation On WordPress Custom Post Type (CPT) From C# Over REST API. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *7*(4), 323-331. DOI: https://doi.org/10.5281/zenodo.10408545

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJCSBE.2581.6942.0322

Paper Submission: 05/12/2023
Paper Publication: 20/12/2023

CRUD Operation On WordPress Custom Post Type (CPT) From C# Over REST API

Sudip Chakraborty [1] & P. S. Aithal [2]
[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2] Senior Professor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *WordPress is a popular content management system. Around 45% of the internet's websites are powered by WordPress. Its main feature is posts. The contents are stored as posts. WordPress uses a database to store posts and configurations. Its REST API interface is robust and feature-rich. We can efficiently save the data from external environments into the database over the REST API as a post. However, standard post types provide enormous amounts of data. Low-speed systems and real-time processing can create a bottleneck. The data flow can be optimized using custom post type (CPT). Here, we describe how to create a custom field using available plugins. We make a REST API client using C# language. From the client application, we execute CRUD operation. The complete project code is freely available on GitHub.*
Design/Methodology/Approach: *There are several ways to test WordPress code. Instead of testing in live WordPress website, the best way is to work in local website. It is not harmful anymore because code is deploying in local only. to install WordPress locally, we use the "**LOCAL**" application. Using this approach, we can eliminate the server hiring cost for the experiment. after that inside the local application, we install WordPress Website. Then create Custom route to navigate our code. For client side, we make a client application in C# with GUI for better operation.*
Findings/Result: *We ran the project for a long time with practical examples without any issues. This architecture can be used to store IoT data inside the cloud. For the POST data is stored in the "**wp_Posts**" table where the additional CPT field is kept inside the "wp**_postmeta**" table, a couple of extra transactions that might cause data flow delays.*
Originality/Value: *WordPress database generally stores external data using post type. Here, we are using an approach beyond general usage. Even C# is not used for this purpose. The other more high-level languages are available to do our task effectively. However, the researchers from the dot net framework domain who want to keep data in the WordPress database as a custom post type can get this as a practical reference.*
Paper Type: *Experimental-based Research.*
Keywords: WordPress custom post type, WordPress with dot net, WordPress with C# implementation. WordPress CPT demonstration. CPT with C#. CRUD operation on CPT.

1. INTRODUCTION :

The IoT market is booming. Sensors and sensors now surround us. For analysis, we need to upload sensor data to the cloud or any server where they are running around the clock. Now, several IT giant company has their IOT infrastructure. Generally, we deploy our project into that cloud. But it takes an amount of expanse per month. It is acceptable for the company where cost generally does not matter; quality of service provided by setup matters. But when we are research people, some of our research budgets don't permit IoT cloud subscriptions. So, in that scenario, we have an idea to implement cloud space. Almost every research organization has a website that runs inside the cloud server and is available around the clock. We can use the space to store our sensor data for research. Here is the complete experiment executed using a locally deployed WordPress website. It mimics on behalf of the online web. We install a plugin and create a custom route for request execution. Using the dot net framework and C# language, we created a client application to communicate with the WordPress server over REST API.

2. RELATED WORKS :

Leone, S. et al. describe a meta-plugin for bespoke data management in WordPress [1]. Wright-Porto, H., in his paper, demonstrates how to Set Up a Custom Domain [2]. Budd, A. provides a solution for designing a Blog in WordPress [3]. Fernandes, S. et al. give the procedure for WordPress Digital marketing [4]. Williams, B. et al. describe how to design and develop Professional WordPress [5]. Denis, A. et. Al. provides valuable tricks and techniques for WordPress in their book [6]. Fragulis, G. F. et. Al. demonstrates ODES, an online dynamic examination system based on a CMS WordPress plugin [7]. Silver, T. B., in his book, nicely describes how to design WordPress Theme Design for WordPress Blogs and Websites [8]. Lee, H. J. provided suggestions on WordPress Theme Filtering Service [9]. Murolo, A. et. Al. describes custom post types from digital mock-ups [10]. Slavin, L. et. Al. provides some guidance on Creating dynamic subject guides [11]. In their paper, Chakraborty S. et al. demonstrate GIT technology, which is very helpful in maintaining the research code [12]. Their article explains the debug procedure [13]. Another paper provides information on how to get simulated data from Modbus [14]. Their research paper shows GUI design using the MVVM approach [15].

3. OBJECTIVES :

We have a couple of objectives for this research work. One is to see how we can use our website database as our IOT data storage space to access global data. The second one is to access the WordPress custom data type outside the WordPress environment. The third is to see how we can access the WordPress form C# language or dot net environment. It might provide some practical reference information to the researcher who is spending time or going to learn how to integrate the WordPress database as data storage into their project.

4. APPROACH AND METHODOLOGY :

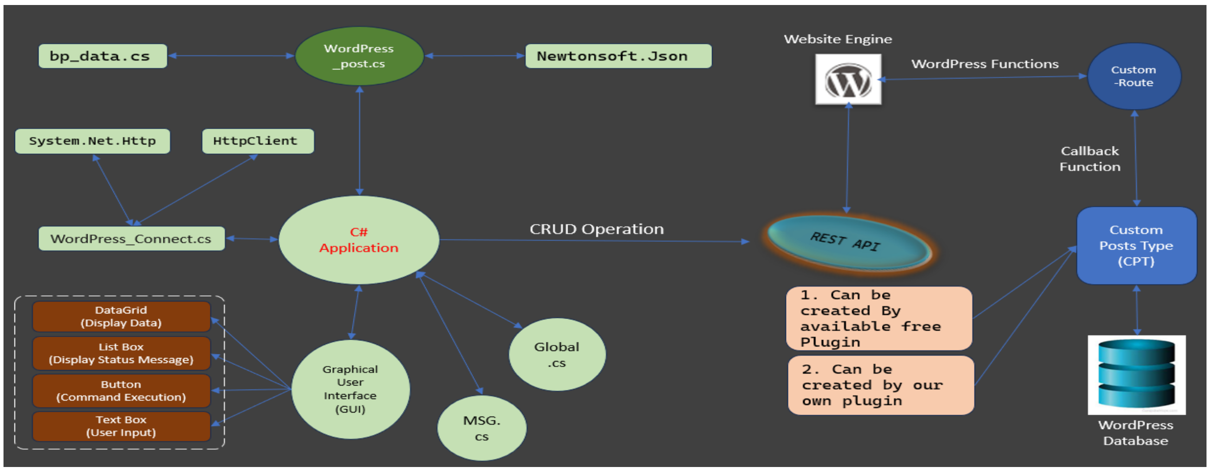
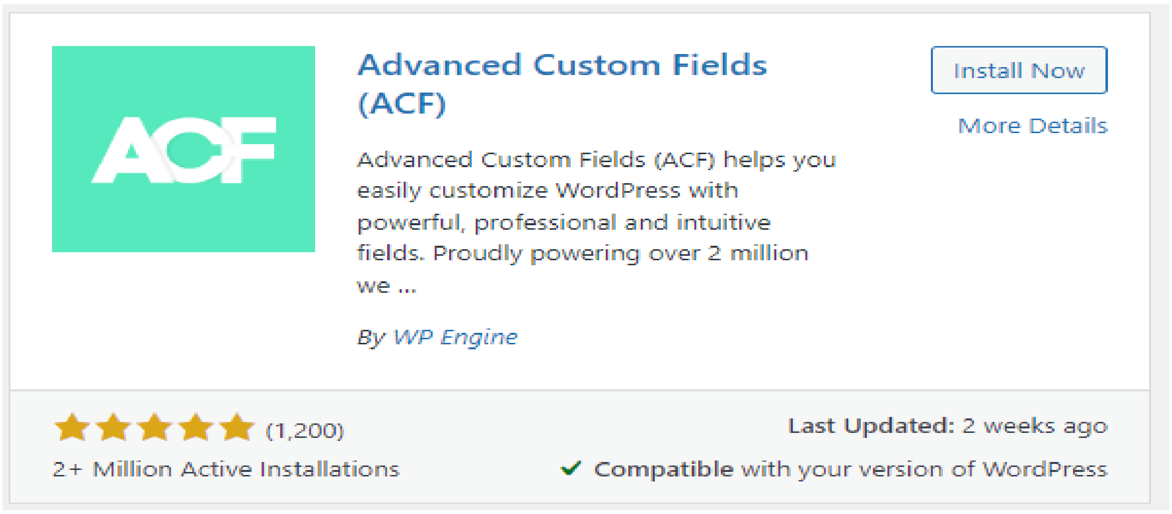


Fig. 1: Project block diagram

Figure 1 depicts the project block diagram. There are two-sided developments we need to do. One is the server side, and the other is the client-side development. We install the LOCAL application for the server side, and inside, we install WordPress websites. After that, we installed the “ACF” plugin to create additional fields. Then, inside the theme folder, we add a custom route. Inside the custom route, we add code to handle the custom post. When a client application sends a request to the server, it reaches the custom route callback function. Then, it parses the command on request, takes action, and responds to the client. If it is a read command, it sends the data as a response. Using the C# application, we create a GUI-based client application for easy user interaction. We use the HTTP client module for communication, including some modules like “**MSG.cs**” and “**Global. cs**,” which are used to help the main module or thread.

5. EXPERIMENT :

Here, we will see how to add a field to the post and make the custom post type or CPT.



Advanced Custom Fields (ACF)

Install Now

More Details

Advanced Custom Fields (ACF) helps you easily customize WordPress with powerful, professional and intuitive fields. Proudly powering over 2 million we ...

By WP Engine

★★★★★ (1,200)

2+ Million Active Installations

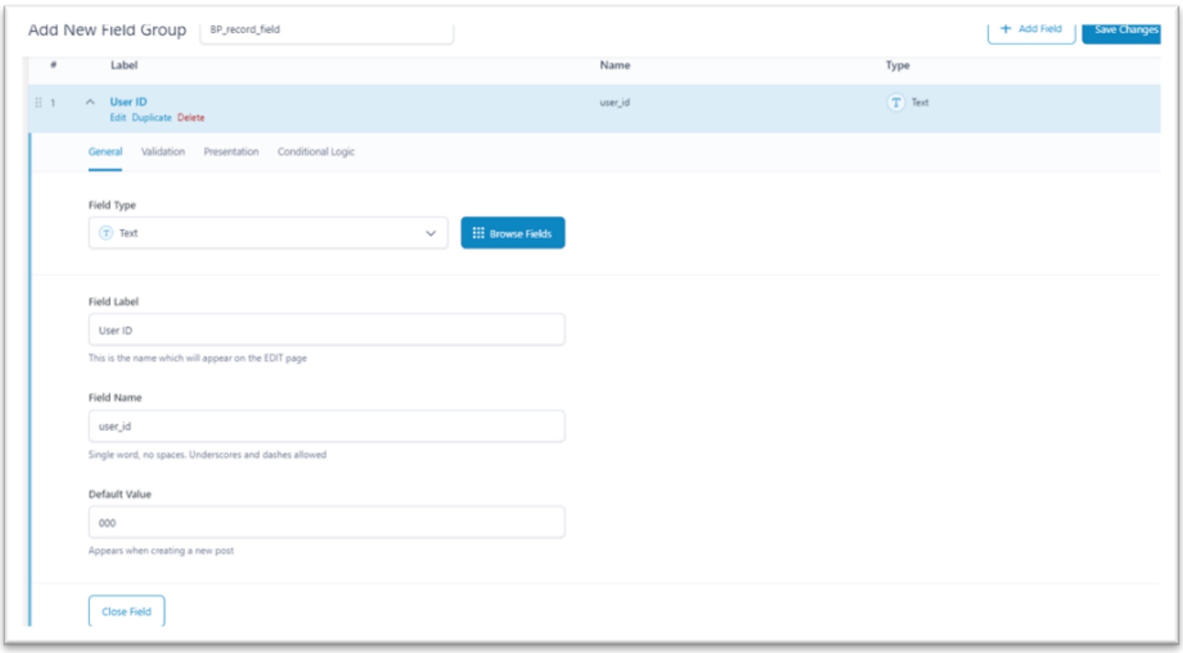
Last Updated: 2 weeks ago

✓ Compatible with your version of WordPress

Fig. 2: ACF plugins

5.1 Custom Post type using the available plugin:

1) Open local software. Click on “WP Admin.”
2) From the left side, click on “Plugins”.
3) Click on “**Add New Plugin**.” Type “advanced custom fields” in the search box and press enter.
4) We will find “**Advanced Custom Fields (ACF) by WP Engine.**”
5) We will get one plugin like in Figure 2. click on install Now.
6) After a couple of seconds, the installation is complete. Then click “**Activate**”.
7) On the left side of the WordPress admin panel, one “ACF” icon appeared after activation. Now click on the “ACF” icon.
8) Click on the “**+Add New**” button.
9) Add filed group title “BP_record_field.” Add field label, field name, and default value. The sample data is shown in Figure 3.



Add New Field Group BP_record_field

+ Add Field | Save Changes

#	Label	Name	Type
1	User ID Edit Duplicate Delete	user_id	Text

General | Validation | Presentation | Conditional Logic

Field Type: Text | Browse Fields

Field Label: User ID
This is the name which will appear on the EDIT page

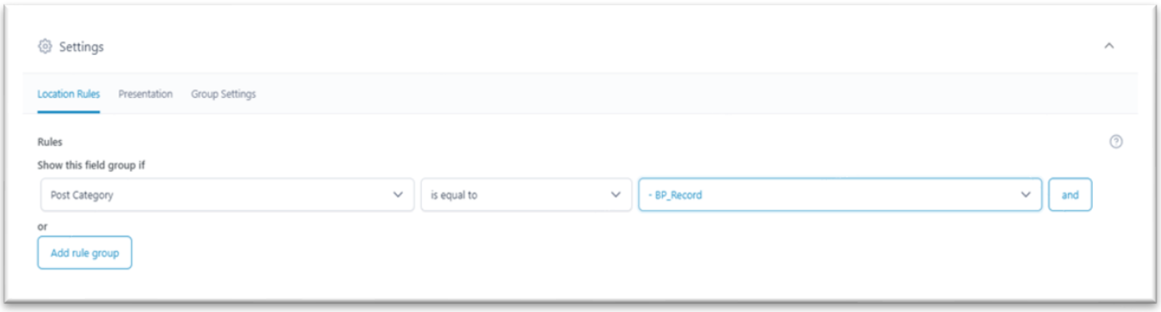
Field Name: user_id
Single word, no spaces. Underscores and dashes allowed

Default Value: 000
Appears when creating a new post

Close Field

Fig. 3: Add New Field Group interface.

10) Using the same procedure, create another three fields:
11) Field Type: Text, Field Label=SYS, Field Name=SYS, default value=120
12) Field Type: Text, Field Label=DIA, Field Name=dia, default value=80
13) Field Type: Text, Field Label=PUL, Field Name=pul, default value=72
14) after entering the field, we need to set location rules. Figure 4 depicts example location rules.



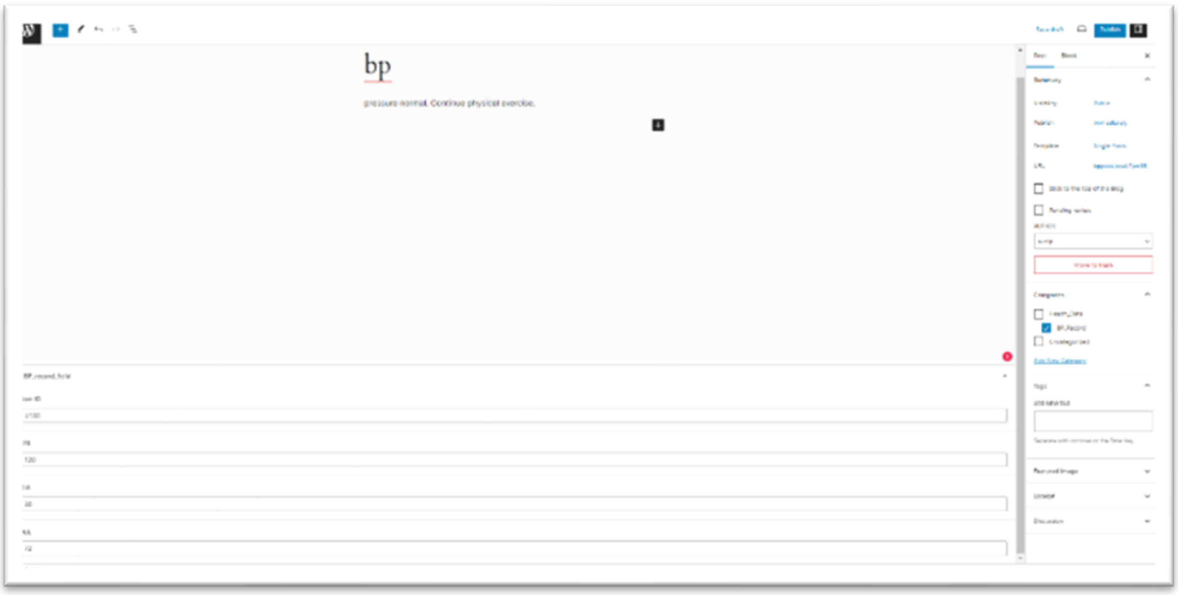
Rules
Show this field group if
Post Category | is equal to | - BP_Record | and
or
Add rule group

Fig. 4: Location Rules Configure Interface.

15) At the bottom, change the slide to "**show in REST API**." now, on the top side, click "save changes."

5.2 Add Data to the Custom Post:

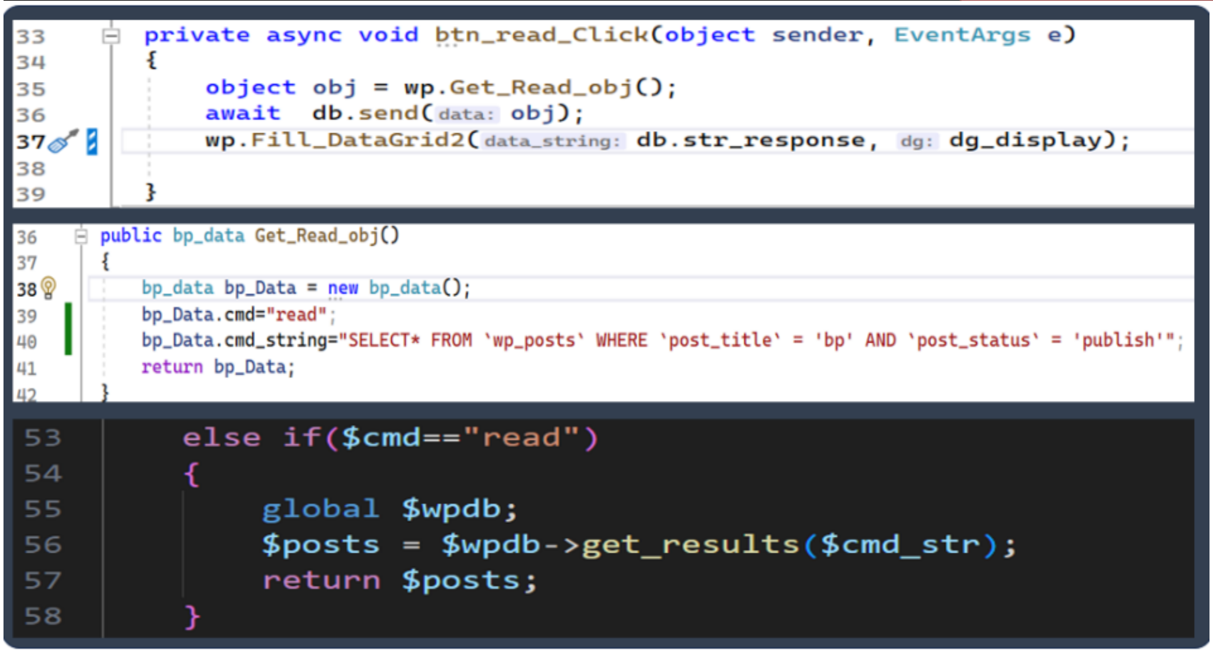
Now we created our place where we will store our data. In this place we can store data using two ways. one we can add or insert data from WordPress admin panel and other way is from WordPress external environment using REST API. There is one major drawback to add data from admin panel. Every time we need to login inside the WordPress panel using login credentials. The best way is to add, modify and delete from the client application because using the GUI element , user can easily add, modify read and delete data from the application. Now we will see how we can insert data to the database from WordPress admin panel. Then we will do same task from GUI based application. At first Click "**Add New Post**" from the admin panel under the post. Add the title "bp." From the right side, under the category, select "**Health_Data**" and "**BP_Record**.". now, fill in your User ID, SYS, DIA, and PUL, and click on the "publish" button. Figure 5 depicts sample data added through the WordPress admin panel.



bp

pressure normal. Continue physical exercise.

Fig. 5: Example Data entry inside the WordPress Admin Panel



```
33      private async void btn_read_Click(object sender, EventArgs e)
34      {
35          object obj = wp.Get_Read_obj();
36          await  db.send(data: obj);
37          wp.Fill_DataGrid2(data_string: db.str_response, dg: dg_display);
38
39      }
```

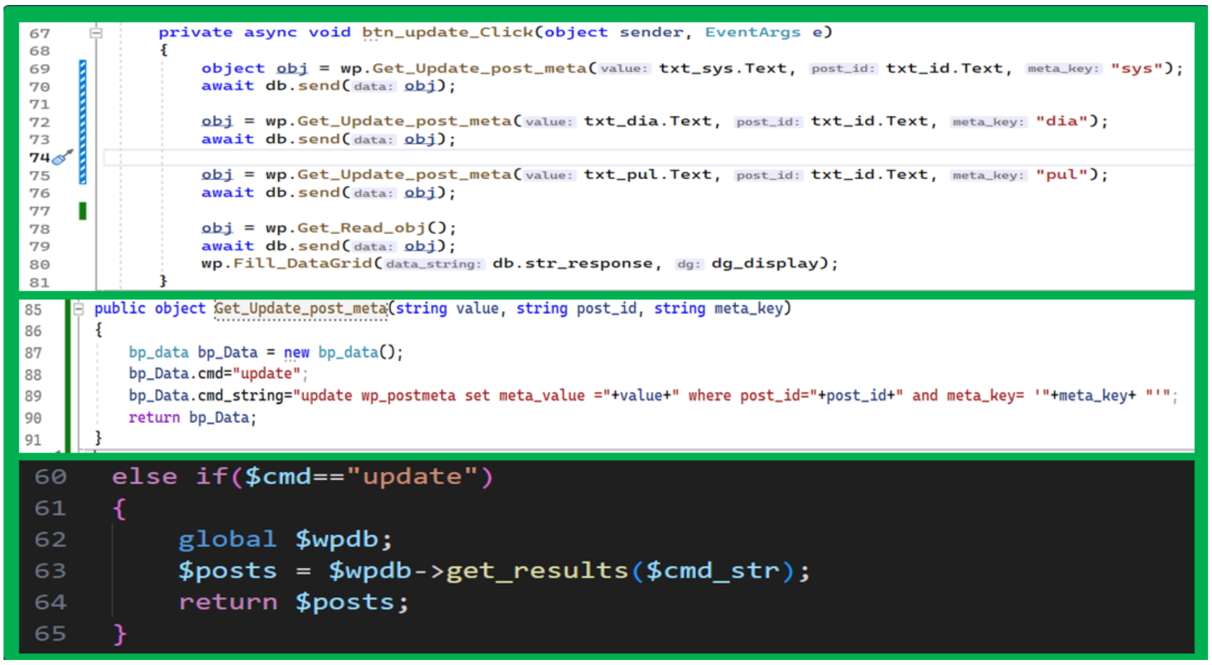
```
36  public bp_data Get_Read_obj()
37  {
38      bp_data bp_Data = new bp_data();
39      bp_Data.cmd="read";
40      bp_Data.cmd_string="SELECT* FROM `wp_posts` WHERE `post_title` = 'bp' AND `post_status` = 'publish'";
41      return bp_Data;
42  }
```

```
53      else if($cmd=="read")
54      {
55          global $wpdb;
56          $posts = $wpdb->get_results($cmd_str);
57          return $posts;
58      }
```

Fig. 6: Example of read function

5.3 Read the custom post from the client application:

Now we read it and display it inside the data grid. Figure 6 depicts the read function. In the figure, the first part is the button handler code. Line 35 to 37 code is added to a " READ " button. Pressing the read button will get a read JSON object string. In the second part of the code, the section shows how the function prepares the JSON string object. Then, it is sent to the WordPress server. The server reads the command and parses it. It matches with the available command. The third part of the code section describes the server processing part. Once the command matches the read command, the receive command string, an SQL command string, is sent to the WordPress engine using the “get_results()” function. Whenever the server returns, it directly sends back to the client as a command response. One thing to remember: from “wp_posts,” we only process the post ID. In the custom post scenario, the additional fields are available in “wp_postmeta.” So, we need to send a request to read the additional field from “wp_postmeta” table for every row.



```
    private async void btn_update_Click(object sender, EventArgs e)
    {
        object obj = wp.Get_Update_post_meta(value: txt_sys.Text, post_id: txt_id.Text, meta_key: "sys");
        await db.send(data: obj);

        obj = wp.Get_Update_post_meta(value: txt_dia.Text, post_id: txt_id.Text, meta_key: "dia");
        await db.send(data: obj);

        obj = wp.Get_Update_post_meta(value: txt_pul.Text, post_id: txt_id.Text, meta_key: "pul");
        await db.send(data: obj);

        obj = wp.Get_Read_obj();
        await db.send(data: obj);
        wp.Fill_DataGrid(data_string: db.str_response, dg: dg_display);
    }
```

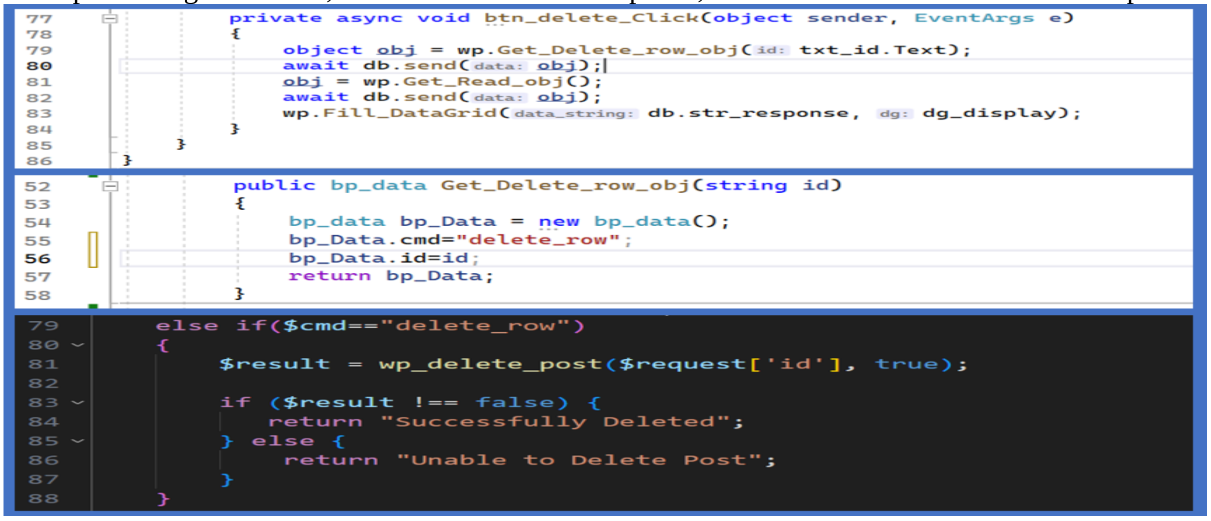
```
public object Get_Update_post_meta(string value, string post_id, string meta_key)
{
    bp_data bp_Data = new bp_data();
    bp_Data.cmd="update";
    bp_Data.cmd_string="update wp_postmeta set meta_value ="+value+" where post_id="+post_id+" and meta_key= '"+meta_key+ "'";
    return bp_Data;
}
```

```
else if($cmd=="update")
{
    global $wpdb;
    $posts = $wpdb->get_results($cmd_str);
    return $posts;
}
```

Fig. 7: Example of update function

5.4 Update CPT data:

We can update the custom post-type data. First, we need to read the data from the WordPress server. Then, the fetched data will populate the data grid. Now, we must select data from DataGrid by clicking on the row depicted in figure 9. The input text box will update from the clicked row data. We can directly change the data inside the text field. Once the change is completed, Click the Update button. Figure 7 depicts the update code. The first part is the button-handled code. Clicking the "UPDATE" button prepares the JSON string object, providing the update parameter. Then, it sends it to the server. Then, similarly, we update for every field. Once the update is completed, the function again reads the data and displays it inside the data grid so that we can confirm the data is updated. The third part of the figure depicts the server-side PHP code. Once the command string is sent to the WordPress engine, it starts processing. After that, the server returns the response, whether it is successful or fails to update.



```
        private async void btn_delete_Click(object sender, EventArgs e)
        {
            object obj = wp.Get_Delete_row_obj(id: txt_id.Text);
            await db.send(data: obj);
            obj = wp.Get_Read_obj();
            await db.send(data: obj);
            wp.Fill_DataGrid(data_string: db.str_response, dg: dg_display);
        }
    }
}
```

```
        public bp_data Get_Delete_row_obj(string id)
        {
            bp_data bp_Data = new bp_data();
            bp_Data.cmd="delete_row";
            bp_Data.id=id;
            return bp_Data;
        }
```

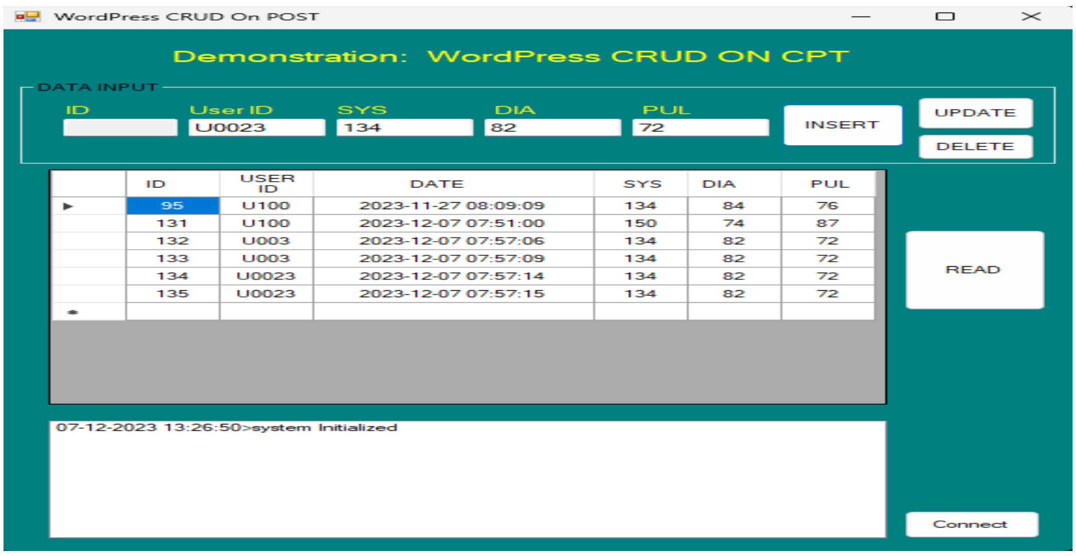
```
    else if($cmd=="delete_row")
    {
        $result = wp_delete_post($request['id'], true);

        if ($result !== false) {
            return "Successfully Deleted";
        } else {
            return "Unable to Delete Post";
        }
    }
```

Fig. 8: Example of delete function

5.5 Delete Custom Post Type:

Deleting a post is not too complex a task. To delete any post, you need to know the post ID. At first, we need to add a couple of codes inside the button handler. Figure 8 depicts the deleted postcode. depicts the deleted code for the CPT. The first part is the button handler function. Before we delete, we need to select the row to delete. Just click the row in the data grid. Then we press the delete. At first, it will create a JSON object providing a post ID. Then, we send the object to the server. Once deleted, we again read the post. We observe the data grid. The deleted post is unavailable inside the data grid, meaning the row or the post has been deleted. The result can be executed from the client application.



ID	User ID	SYS	DIA	PUL
	U0023	134	82	72

ID	USER ID	DATE	SYS	DIA	PUL
95	U100	2023-11-27 08:09:09	134	84	76
131	U100	2023-12-07 07:51:00	150	74	87
132	U003	2023-12-07 07:57:06	134	82	72
133	U003	2023-12-07 07:57:09	134	82	72
134	U0023	2023-12-07 07:57:14	134	82	72
135	U0023	2023-12-07 07:57:15	134	82	72

```
07-12-2023 13:26:50>system Initialized
```

Fig. 9: The Client Graphical User Interface (GUI)

6. RECOMMENDATIONS :

- The project code is available to download from: https://github.com/sudipchakraborty/CRUD-Operation-On-WordPress-Custom-Post-Type-From-C-sharp-Over-REST-API.git
- The project code is kept as simple as possible. If we implement it in the production environment, we need to add exception handling.
- Using the project, we can explore several projects. We have a couple of project ideas we added [12][13][14][15].

7. CONCLUSION :

The researcher needs to store various research or sensor data in the cloud. Most of the cloud storage is not free. So, the WordPress website database can be used as a cloud server. WordPress is a popular content management system. WordPress uses a database to store posts and configurations. In this research work, we observe how to implement WordPress custom post-type to store our research data. Through the step-by-step procedures using practical examples, we execute the task. Without spending extra money, we can easily keep our sensor data in the website database directly, which is already running around the clock.

REFERENCES :

[1] Leone, S., de Spindler, A., & Norrie, M. C. (2012). A meta-plugin for bespoke data management in WordPress. In Web Information Systems Engineering-WISE 2012: 13th International Conference, Paphos, Cyprus, November 28-30, 2012. Proceedings 13 (pp. 580-593). Springer Berlin Heidelberg. Google Scholar↗

[2] Wright-Porto, H. (2011). Setting Up a Custom Domain. In Creative Blogging: Your First Steps to a Successful Blog (pp. 143-162). Berkeley, CA: Apress. Google Scholar↗

[3] Budd, A., Collison, S., Davis, C. J., Heilemann, M., Oxton, J., Powers, D., ... & Heilemann, M. (2006). WordPress. *Blog Design Solutions*, 171-213. Google Scholar↗

[4] Fernandes, S., & Vidyasagar, A. (2015). Digital marketing and WordPress. *Indian Journal of Science and Technology*, *8*, 61. Google Scholar↗

[5] Williams, B., Damstra, D., & Stern, H. (2015). *Professional WordPress: design and development*. John Wiley & Sons. Google Scholar↗

[6] Denis, A., & Magazine, S. (2013). *Useful Tricks and Techniques for WordPress*. Smashing Magazine. Google Scholar↗

[7] Fragulis, G. F., Lazaridis, L., Papatsimouli, M., & Skordas, I. A. (2018, September). ODES is an online dynamic examination system based on a CMS WordPress plugin. In *2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_CECNSM)* (pp. 1-8). IEEE. Google Scholar↗

[8] Silver, T. B. (2009). *WordPress 2.8 Theme Design: Create Flexible, Powerful, and Professional Themes for Your WordPress Blogs and Websites*. Packt Publishing Ltd. Google Scholar↗

[9] Lee, H. J. (2017). Development of Web UX Pattern System for Wordpress Service and Design Suggestion for Wordpress Theme Filtering Service. *International Journal of Contents*, *13*(1), 09-21. Google Scholar↗

[10] Murolo, A., & Norrie, M. C. (2015). Deriving custom post types from digital mockups. In *Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings 15* (pp. 71-80). *Springer International Publishing.* Google Scholar↗

[11] Slavin, L., & Dodson, J. (2011). Creating dynamic subject guides. *Library Technology Reports*, *47*(3), 34-37. Google Scholar↗

[12] Chakraborty, S., & Aithal, P. S., (2022). A Practical Approach To GIT Using Bitbucket, GitHub, and SourceTree. *International Journal of Applied Engineering and Management Letters (IJAEML), 6*(2), 254-263. Google Scholar↗

[13] Chakraborty, S., & Aithal, P. S. (2023). Industrial Automation Debug Message Display Over Modbus RTU Using C#. *International Journal of Management, Technology, and Social Sciences (IJMTS), 8*(2), 305-313. Google Scholar↗

[14] Chakraborty, S., & Aithal, P. S. (2023). Modbus Data Provider for Automation Researcher Using C#. *International Journal of Case Studies in Business, IT, and Education (IJCSBE), 7*(3), 1-7. Google Scholar↗

[15] Chakraborty, S., & Aithal, P. S., (2023). MVVM Demonstration Using C# WPF. *International Journal of Applied Engineering and Management Letters (IJAEML), 7*(1), 1-14. Google Scholar↗

****