International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

Design and Development of a Headless Content Management System

Jigisha Kamal[1]

[1]*Department of Information Science and Engineering, R.V. College of Engineering, Bangalore, Karnataka, India*

Abstract - *Website development has evolved to become a prominent and ever-growing field in today's world. Having an effective web presence is critical for achieving sustainable business growth. However, small scale businesses with limited budgets cannot afford to hire professionals to develop their website. In such a situation, a Content Management System is a cost-effective solution as it allows non-developers to build websites without the requirement of any coding knowledge. While in a traditional CMS, the frontend and backend are coupled together, in a headless CMS, the backend is completely separated from the frontend thus, making it more flexible and scalable. The purpose of this research paper is to design and develop a headless content management system and evaluate its performance.*

***Key Words*: Web Development, Content Management System, Headless Content Management System, API, React, NodeJS, Docker**

1. INTRODUCTION

The internet has grown exponentially in the past two decades and with this, the number of businesses and users on the internet has increased tremendously. In today's day and age, most people prefer to use the internet for everything from research to business. Thus, it has become increasing important for businesses to enter the digital world.

A well-developed website can have an incredible effect in complementing a business's digital marketing efforts and can effectively drive-up sales. Thus, a lot of companies have increased their investment in web development so that they can enhance their businesses by meeting the needs of their digital consumers. However, the process of developing a website is a time-consuming task and requires professionals who have knowledge about various web development technologies. This can turn into an expensive affair for businesses that are just starting up [15]. This along with the rapid developments in the field make it difficult for even developers to keep updated with the latest technologies. This is where a Content Management System (CMS) comes into picture.

A CMS is essentially a software that enables the creation and storage of content in a digital format. This content is then used to create a website based on certain templates. It does not require the user to have any prior knowledge about web development and coding. There are two parts that make up a traditional CMS, i.e., a content management application (CMA) that enables a user to create the content and that will be displayed on the website and a content delivery application (CDA) that delivers the content to the website once the user creates or updates the same [8]. Additionally, there are three types of architecture that are commonly used. The first is the traditional or monolithic architecture, where the backend and frontend are coupled together into a single application. The most popular example of this is Wordpress [6]. A decoupled architecture comes next, in which the backend can either be integrated with the frontend or the content can be pushed to the frontend via API calls. But the data from the backend is continuously pushed somewhere even if there are no API calls being made. Lastly comes the headless architecture, where the backend is completely separated from the frontend or the presentation layer. The developed content is made accessible via REST API or GraphQL and can be displayed on any device possible.

Due to the differences in architecture, the performance of these CMSs also varies. This research aims to develop a headless content management system using Javascript, NodeJS, Postgres, Redis, React and Docker, while also providing an insight into the advantages and disadvantages of a headless CMS. Along with the headless CMS, an authentication service is also developed in order to provide a level of security to the CMS application.

2. RELATED WORKS

A comprehensive literature review was conducted with reference to numerous technical journals and writings published in standard periodicals. Most of the research done in this field involves assessing and comparing the performance of existing Content Management Systems like Wordpress, Drupal, Jumla, etc.

In 2018, a research study [1] was done which involved developing a hand coded and a Wordpress website to render the same content and analyze the load time for each. It was found that the load time for the hand coded website is larger than WordPress for image, text and video content, but more API calls are made in Wordpress.

Another research in 2018 was done [2] in order to analyze the impact of WordPress in website development. It was concluded that Wordpress powers 29% of all websites. It is backward compatible, provides a number of plug-ins and themes for developing websites.

The authors of [3] studied the use of open-source CMSs for creating and managing web content pertaining to all important Union government ministries in India. Their results showed that WordPress is the most prominent globally. In government ministries, Drupal is the most commonly used.

A unique research done in 2015 [4], involved using data-driven and mobile-driven technology which uses duplication and multiplication in cloud for creating a standard and user-friendly website. The authors developed a CMS where users can create and manage their website without pre-install software. It offered superior performance than traditional CMSs.

In a study conducted in 2014 [5], it was found that Joomla, Drupal and Wordpress are the top 3 CMSs amongst Comparing Alfresco, Typo3, Dotnetnuke, Drupal, Joomla, Wordpress and Plone, based on system requirements, performance, security, built-in features, support. However, there is no one CMS that provides all the best features and the choice of which one to use depends on the user requirements.

Lastly, a 2013 study [6] compared 13 Java based open-source CMSs available based on 29 features like admin control, auto code generation, multiple user support, etc, and concluded that Alfresco is the best available open-source CMS.

3. DESIGN OF THE PROPOSED SYSTEM

This section describes the design of the overall system. The system is broadly divided into 3 parts. These include the CMS service frontend, the CMS service backend and the Authentication service. In order to get a better understanding of the high level overview of the entire system, the overall architecture of the proposed system is shown in Fig-1.

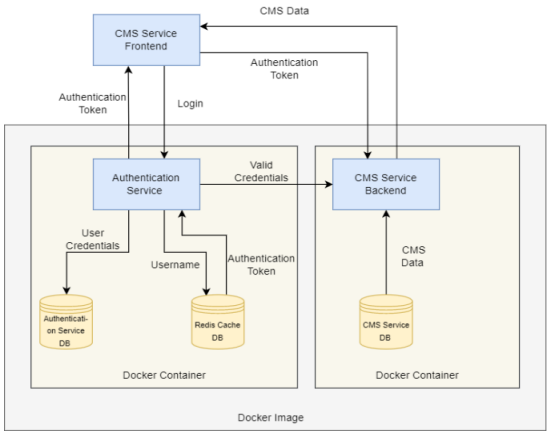

Fig-1: Architecture for the Design and Development of a Headless CMS

As seen in the above figure, there are two services, the CMS service and the Authentication service. The CMS service has a frontend and a backend. Once a user logs into their account, the user credentials are stored in the Authentication service database and the username is used to generate a unique authentication token for the user, which is stored in the Redis cache database. This authentication token is sent back to the CMS frontend and all requests to the CMS backend are authenticated using the token. If the token is valid, data is sent from the CMS service database to the frontend. The CMS service backend and the authentication service are dockerized into their respective Docker containers. A Docker image is created comprising of these two Docker containers. In order to run the application, only the docker image needs to be run, using docker-compose.

In order to understand the system from the user's perspective, an activity diagram was designed which can be seen in Fig-2.

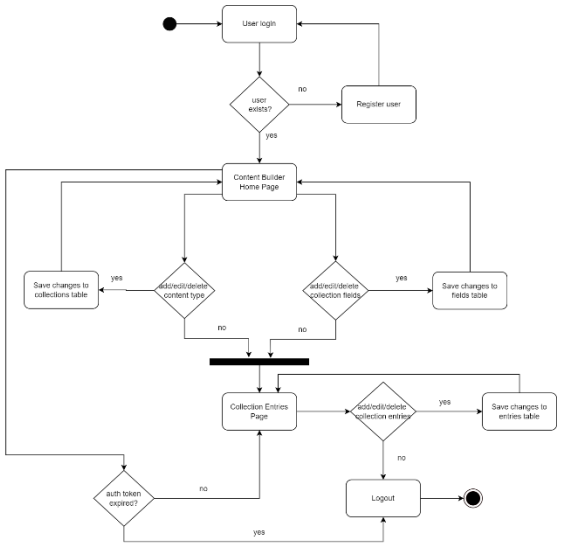

Fig-2: Activity Diagram for the Design and Development of a Headless CMS

As seen in the above figure, first, the user is prompted to login. If the user has an account, the credentials are accepted otherwise the user is directed to the registration page where they create a new account with which they can login to the CMS service. Once logged in, the user is directed to the Content Builder page where they can add, edit and delete content types and their respective fields. Any changes made are saved to the collection and fields tables of the CMS service database respectively. Since a unique authentication token is generated for each user when they login, there is a continuous check to see whether the authentication token has expired. If the token expires, the user is automatically logged out of their account and taken to the login page. Otherwise, the user can continue to make changes on the CMS pages. If the user doesn't wish to add, edit or delete the content type or it's fields, they can navigate to the Collection

Entries Page where they have the option to add, edit and delete entries for a particular content type. The changes made are saved to the entries table of the CMS service database and reflected on the frontend in real time. All the pages provide the user with a logout functionality, using which the user can logout of their account.

4. METHODOLOGY

This section describes the development process that has been followed to create the headless CMS. The development of the entire application has been divided into 4 tasks. The first is to develop an authentication service which provides functionalities to register and login a user and enables access control using an authentication token. Second comes the design and development of the headless CMS service that will enable content creation, content storage. The third task includes integrating the authentication service with the CMS service. Lastly, the entire application is dockerized.

4.1 Development of the Authentication Service

This involves developing the functionalities that will allow a user to register and login to their CMS account. An authentication token is generated every time the user logs in and this is unique for each registered user. This is stored as a key-value pair in Redis, a cache database. This token is used to maintain access to the CMS account and an expiration time is set after which the token becomes invalid and thus, the user is logged out of their account. The authentication service is accessed via API calls.

As a part of this service, the database schema for storing the user data is created and then the API endpoints for registration and login are created. The login is then integrated with Redis to generate unique authentication tokens for each user. For the login API, the credentials entered by the user would be sent as request body parameters in the API call. The login service would then make a query to check whether the credentials have a match in the database. If the credentials match, a success message is returned but if the username is correct but the password is incorrect a message stating the same is returned. Lastly, if neither the username and password match, a message requests the user to register is returned. Similarly, for the registration API, where a new user is created, the credentials and details entered by the user would be sent as request body parameters in the API call and a query to add the data to the Postgres database would be made.

4.2 Development of the CMS Service

This constitutes the core of the research and involves developing the frontend and backend for the Content Management System. This service should enable a user to create a new content type and dynamically update it by adding fields to that content type. Additionally, the user should be able to create data for a given content type in the form of entries and there should be functionality that gets all the data for a given content type. Furthermore, the user should have the option to edit and delete the content type and its respective fields and entries.

In order to achieve this, first the CMS backend is developed which includes designing the database schema and developing the APIs that will handle the creation, editing and deleting of a content type, it's fields and its entries. It was decided to store a particular collection's fields and entries in the JSON format as a single field (field name and field type) and entry (entry data and field) would be easily accessible. There are 2 main pages as a part of the application. These are the Content Builder page and the Collection Entries page.

The Content Builder page that enables the user to view, add, edit or delete different content types and their respective fields. The user can use the user interface to add a content type and this is saved in the CMS service database which then returns the list of all the content types (including the newly added content type) and this is displayed to the user. The user can then add fields to a newly created or an existing content type. As a part fields, the user is required the enter the name of the field and the data type it will accept. For example, 'Person' can be a content type and 'Name' can be a field that is added to it which takes the data type 'String'. Furthermore, the user can edit or delete fields within a content type as well as edit and delete the entire content type. The changes made are reflected in the CMS service database as well as on the user interface.

The Collection Entries page enables the user to add, edit entries for a particular content type. The user is displayed a form where they are prompted to provide an entry for each of the defined fields for that collection type, The data is stored as a JSON containing the entry value and the field name. For example, for the collection type, 'Person', a collection entry, 'Jane Doe' would correspond to the collection field, 'Name'. The user can also edit and delete entries for a collection type. All the changes made are reflected in the CMS service database as well as on the user interface.

The HTTP request for data is made to the backend via the Axios library which takes the API endpoint as a parameter and returns the data fetched from the database.

4.3 Integration of the Authentication Service with the CMS Service

The CMS service has a register and login page that is integrated with the authentication service developed. This is done so that all API calls from the CMS that access the database service are authorized via a unique authentication token for each user once they are logged in to their account. The user is automatically logged out once the authentication token expires. This provides an additional layer of security.

4.4 Dockerization of the Application

It is essential to ensure that there are no deployment or runtime issues that arise due to the change in the system that's deploying the application. This is the reason why Docker is used in order to perform containerization of the Authentication service and the CMS service.

A docker file is created for the Authentication service and the CMS service which describes all the commands a user would call on the command line to assemble an image. Since there are two different services, Docker Compose is used as it helps in defining and running multiple Docker containers as a single service. Once this is done, the entire application is present within a Docker image and can be run with a single command.

5. RESULTS

This section contains information about the results that were obtained while carrying out this research. It also discusses the results of the testing that was carried out on the system.

A headless CMS was successfully built that enables content creation and storage. The system was tested extensively using the Jest testing library. Unit testing, integration testing and system testing were carried out to ensure that each of the components carry out their required functionality as well as to check that the flow of data between various components and to ensure that a particular component is not impacted by the presence or absence of another component. A code coverage of greater than 90% was achieved for the system.

The designed system is user friendly and allows the creation of various content types, their fields and their respective entries. This data can then be effectively utilized to create a website via API calls. Since a headless CMS completely separated the backend from the frontend, there is no CDA that is associated with the application and the data that has been created can be accessed via API calls. The developed application supports the same.

As a part of the research, a headless content management system was developed. This differs from a traditional CMS as it completely separates the backend or the content creation and storage part from the frontend or the presentation layer. Such an architecture enables multichannel content delivery as the data is accessed via API calls and thus, can be modified as required before being displayed onto different devices. This also enables content reuse as the frontend is designed independently. This is not possible in case of a traditional architecture as the frontend and backend are coupled together and there is a structured presentation layer that gets its data from the content database in the backend. Unlike a traditional CMS, with a headless CMS, there are no limitations on the frontend technologies that will used to deliver the content and the technology best suited for the desired application can be utilized. Thought a headless CMS performs better in terms of faster page load time and provides more customization options, it has a few drawbacks. It is more expensive to develop and does not provide pre-built templates to display content like most traditional CMSs.

6. CONCLUSION

This research demonstrates the design and development of a headless CMS. The designed system is effective in providing capabilities for content creation and storage however, it does not include workflow management to control permissions for different levels of users and to manage various roles and approvals. The introduction of this can help in making the developed system ready for commercial use as workflow management forms an important part for any business use case. Furthermore, the introduction of drag-and-drop features can enable marketers to model their own content based on certain templates. This could be useful for designing some custom components. Lastly, a feature that enables integration of data feed from a third party like a S3 bucket can reduce the manual work to be done by the user as they will not have to manually add the collections entries.

Overall, it can be seen that a headless content management system is more flexible and performs better in terms of faster page load time. Additionally, it is a better option in the long run as the decoupling of the frontend and the backend ensures that the entire system doesn't have to be redesigned when any changes need to be made. It is also more secure as there is no direct access to the content that is used to create the website. Hence, the use of a headless CMS can have a great impact in developing a good website for a business as its excellent performance in terms of load time can greatly improve a website's ranking.

REFERENCES

[1] A. Kumar, A. Kumar, H. Hashmi and S. A. Khan, "WordPress: A Multi-Functional Content Management System," 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), 2021, pp. 158-161, doi: 10.1109/SMART52563.2021.9675311.

[2] G. Maragatham, S. N. A. Balaji, K. SaiKarthikeyan, V. Gokulakrishnan and M. Siddharth, "A Study on Performance Analysis for Different Wordpress and Hand Code Webpages," 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2018, pp. 191-204, doi: 10.1109/ICSSIT.2018.8748564.

[3] A. Mirdha, A. Jain and K. Shah, "Comparative analysis of open source content management systems," 2014 IEEE International Conference on Computational Intelligence and Computing Research, 2014, pp. 1-4, doi: 10.1109/ICCIC.2014.7238337.

[4] C. C. Hoong and M. A. Ameeden, "Intuitive Content Management System," 2015 International Conference on Computer, Communications, and Control Technology (I4CT), 2015, pp. 541-543, doi: 10.1109/I4CT.2015.7219637.

[5] M. Nath and A. Arora, "Content management system: Comparative case study," 2010 IEEE International Conference on Software Engineering and Service Sciences, 2010, pp. 624-627, doi: 10.1109/ICSESS.2010.5552271.

[6] Cabot, Jordi. (2018). WordPress: A Content Management System to Democratize Publishing. IEEE Software. 35. 89-92. 10.1109/MS.2018.2141016.

[7] A. Kumar, A. Kumar, H. Hashmi and S. A. Khan, "WordPress: A Multi-Functional Content Management System," 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), 2021, pp. 158-161, doi: 10.1109/SMART52563.2021.9675311.

[8] N. A. Khan and H. Ahangar, "Use of Open Content Management Systems in Government Sector," 2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), 2018, pp. 183-187, doi: 10.1109/ETTLIS.2018.8485191.

[9] Yermolenko, Andrei & Golchevskiy, Yuriy. (2021). Developing Web Content Management Systems – from the Past to the Future. SHS Web of Conferences. 110. 05007. 10.1051/shsconf/202111005007.

[10] H. Liduo and C. Yan, "Design and implementation of Web Content Management System by J2EE-based three-tier architecture: Applying in maritime and shipping business," 2010 2nd IEEE International Conference on Information Management and Engineering, 2010, pp. 513-517, doi: 10.1109/ICIME.2010.5477670.

[11] C. Esperança and A. Pereira, "Content management system for e-Government portals," 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), 2016, pp. 1-6, doi: 10.1109/CISTI.2016.7521369.

[12] D. Michelinakis, "Open Source Content Management Systems: An Argumentative Approach", Award MSc Electron. Bus. Manag, pp. 113, 2004.

[13] U. Naik and D. Shivalingaiah, "Open source software for content management system", Proc. 7th Int. CALIBER Pondicherry Univ. Pondicherry, pp. 225-239, 2009.

[14] X. Cao and W. Yu, "Using Content Management System Joomla! to Build a Website for Research Institute Needs," 2010 International Conference on Management and Service Science, 2010, pp. 1-3, doi: 10.1109/ICMSS.2010.5577465.

[15] P. Kiatruangkrai, P. Phusayangkul, S. Viniyakul, N. Prompoon and P. Kanongchaiyos, "Design and Development of Real-Time Communication Content Management System for E-Commerce," 2010 Second International Symposium on Data, Privacy, and E-Commerce, 2010, pp. 111-116, doi: 10.1109/ISDPE.2010.24.

[16] Chang Liu, Kirk P. Arnett, "Exploring the factors associated with Web site success in the context of electronic commerce", Information & Management, Volume 38, Issue 1, 2000, Pages 23-33, ISSN 0378-7206, doi: 10.1016/S0378-7206(00)00049-5.

[17] Martinez-Caro, Jose-Manuel, Antonio-Jose Aledo-Hernandez, Antonio Guillen-Perez, Ramon Sanchez-Iborra, and Maria-Dolores Cano. 2018. "A Comparative Study of Web Content Management Systems" Information 9, no. 2: 27. https://doi.org/10.3390/info9020027