International Journal of Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2025.2.1

Volume 11, Issue 2
February - 2025

Study and Implementation Of API and Hashing Methods in Document Management System

A Rick Pernando Sinaga[1], Elliana Gautama[2]

[1]Department of Information System, Faculty of Information Technology, Perbanas Institute, Jakarta, Indonesia

[2]Department of Data Science, Faculty of Information Technology, Perbanas Institute, Jakarta, Indonesia

[2]Departement of School and Graduates Studies Asia e University, Kuala Lumpur, Malaysia

ABSTRACT

Document Processing Information System, better known as Document Management System, is an information system that utilizes information technology to assist and facilitate the processing of documents in an agency. Currently, the Document Management System is only used as a stand-alone information system and is only limited to managing documents for the system's needs. In its storage, the Document Management System still uses hard disk storage media or computer hard drives in the form of original documents. This makes it difficult for users to provide document backup data and has the risk of document damage and document access from outside the Document Management System. This study aims to connect Document Management System with other information systems that run outside DMS using API and hashing methods. The API method is used as a communication path or bridge so that DMS is a stand-alone information system and can support other information systems in document processing. This study also uses the hashing method to change the form of the original document into plaintext so that the document can be stored in the database storage media so that users can create document backup data, and documents can no longer be accessed from outside the Document Management System.

Key Words: Application Programming Interface (API), Document Management System (DMS), Hashing.

1. INTRODUCTION

A document management system (DMS) is a computer system that stores and accesses electronic files. In addition to tracking various versions of updates made by different users, this system is also used to share files with other users. DMS is often used to store asset and imaging documents, organize workflow systems and manage management records. As a document archiving system, registered users can only access documents on DMS and are not connected or integrated with support systems that run outside DMS. In addition, Document Management System (DMS) uses a document storage method on the hard disk (Hard drive) of the computer running the system. Storing documents on a computer's hard disk (Hard drive) has security risks, such as the absence of encryption or changes in the form of documents so that documents are very easy to identify and easy access to documents on the computer's hard disk (Hard drive) can lead to duplication and retrieval of documents without the knowledge of the authorized party. In addition to security risks, there is also a risk of document damage where storing documents on a computer's hard disk (Hard drive) makes it very difficult to create backup data. Implementing the API (Application Programming Interface) method and the Hashing method in the document processing information system (Document Management System) is expected to be a method for further system development in responding to new challenges and risks in the current DMS system.

Application Programming Interface (API) is an interface used to connect one application to another. The role of the API is as an intermediary that connects different applications from the same platform and across platforms. Application development that requires API consists of several elements, such as functions, protocols, and tools [1]. There are at least three API architectures: RCP (Remote Procedure Call), REST (Representational State Transfer), and SOAP (Simple Object Access Protocol).

Hashing is an arithmetic transformation of a character string into a value representing the original string. According to language, hash means to cut and then combine. In computer science, hash functions are often used to store database data. A hash function is a function that accepts input in the form of a string and then processes it according to the standard of the hash function, which will then produce an output in the form of a string with a fixed length, and the length does not depend on the initial size of the input string [2].

The cryptographic hash function is a mathematical function used in digesting messages. It takes a message as input and produces an output as a hash value [3]. The size of the hash output value depends on the algorithm used, such as 64 bits, 128 bits, or 256 bits. Hash is a one-way encryption, which means it does not have a function to restore the encrypted value [4].

One of the hashing algorithms that is developing is the base64 algorithm. Base64 transformation is one of the algorithms for encoding and decoding data into ASCII format, which is based on the number 64 or can be said to be one of the methods used to encode binary. Base64 transformation is widely used in the internet world as a plaintext data media, so this data format will be much easier to send compared to data formats in the form of binary. The base64 algorithm uses ASCII code and base64 index code to carry out the encryption and description process. This study developed a document management system (DMS) to connect and support other systems that run outside the DMS, using the API method as a communication tool between systems or applications. This study also uses the hashing method to change the format or form of a document containing binary into plaintext. It stores the results of document hashing in the database to minimize security risks and document damage and facilitate the provision of backup data or document backups on the Document Management System (DMS).

2. METHOD

2.1 Document Management System

A document management system (DMS) is a process that can include storing, maintaining, and destroying documents. These documents form a product, such as information or reports that an organization or company can distribute for use by its members and customers. The main goal of a document management system is to help the effectiveness and efficiency of a company's document management, providing the organization's staff and customers with relevant, timely information at the lowest possible cost. Document Management System (DMS) is useful in making business processes effective and efficient. The main benefit is that users can find the information they need quickly, which can help the process become faster, better, and cheaper [5].

2.2 Application Programming Interface (API)

API is software that allows developers to integrate and allow two different applications to connect simultaneously [6]. REST API is one of the architectural designs contained in the API [7]. RESTful API works because the REST client will access data or resources on the REST server where the data or resources are distinguished by global ID or URTS (Universal Resource Identifiers) [8]. The data provided by the REST server can be text, JSON, or XML. The HTTP methods that are generally used in REST servers are as follows:

a. GET : functions to read data/resources from the REST server
b. POST : functions to create new data/resources on the REST server
c. PUT : functions to update data/resources from the REST server
d. DELETE : functions to delete data/resources from the REST server
e. OPTIONS : serves to get supported operations on resources from the REST server.

2.3 Hash

The hash function is a function that accepts input in the form of a string and then processes it according to the standard. The hash function will then produce an output in the form of a string with a fixed length; the length does not depend on the initial size of the input string [2]. The cryptographic hash function is a mathematical function used to digest messages, meaning that a message is needed as input and produces output as a hash value [3]. The size of the

DOI: 10.31695/IJASRE.2025.2.1

hash output value depends on the algorithm used, such as 64 bits, 128 bits, and 256 bits. Hash is one-way encryption. One-way means it does not have a function to return the encrypted value [4].

The hashing algorithm used in this study is Base64 [9]. According to R. Tinendung [10], base64 transformation is one of the algorithms for encoding and decoding data into ASCII format, which is based on the base 64 number or can be said to be one of the methods used to encode binary data. The characters produced in this base64 transformation consist of A..Z, a..z and 0..9, plus the symbols "+" and "/" and one character equal to (=) in the last two characters used for filling or in other words adjusting and completing binary data. The symbol characters that will be produced will depend on the running algorithm process [11].

Base64 transformation cryptography is widely used on the internet as a data format media for sending data because the results of base64 encoding are plaintext, so this data will be much easier to send compared to binary data formats. The base64 algorithm uses ASCII code and base64 index code to carry out the encryption and description process. The base64 index code needs to be modified to encrypt the website URL. The symbol "+" is altered to "-" and the symbol "/" to "_".

Table 1. Base64 Index Code (URL and Filename Safe)

Value	*Encoding*	*Value*	*Encoding*	*Value*	*Encoding*	*Value*	*Encoding*
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	-
15	P	31	f	47	v	63	_
						(*pad*)	=

Source: [12]

According to Aziz Pratama Nugraha and Erwin Gunadhi [12], the base64 encryption technique is simple. If there is a (string) of bytes to be encoded into the base64 algorithm, the stages are:

a. Divide the byte string into 3 bytes.
b. Combine 3 bytes to make 24 bits. Note that 1 byte = 8 bits, so 3 x 8 = 24 bits.
c. The 24 stored bits are combined andken down into 6 bits, producing 4 fractions.
d. Each fraction is converted into a decimal value, where the maximum 6-bit value is 63.
e. The last step is to make the decimal values into indexes to select a maximum of 64 indexes or 63 characters from the base64 compiler. And so on until the end of the bytes string that will be converted. If, in the encoding process, there is a remainder of the divisor, then add a pad character (=) to complete the remainder. Therefore, in base64, one or two characters (=) will sometimes appear.

2.5 Research Stages

In order for this research to be directed and in accordance with the expected goals, the research methodology uses a research framework. The research flowchart can be seen in Figure 1:

DOI: 10.31695/IJASRE.2025.2.1

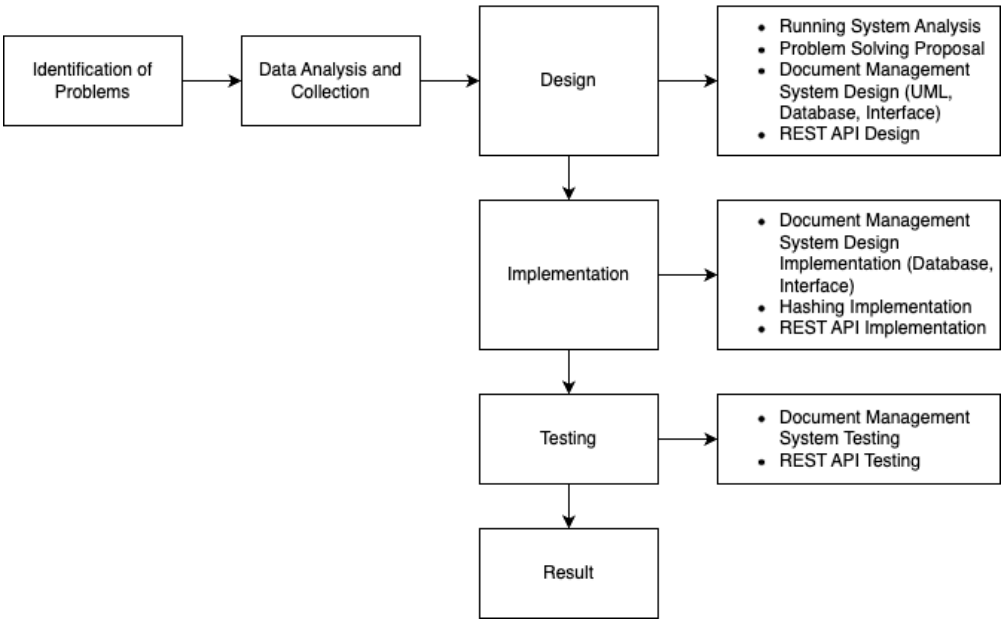

Figure 1. Research Flowchart

3. RESULTS

3.1 Results of System Analysis that is Currently Running

After analyzing the current system (Figure 2), several deficiencies were found in the system, including:

a. Document Management System (DMS) that is not yet connected or integrated and supports systems that run outside the DMS system.
b. Documents in the Document Management System (DMS) are still stored on the computer's hard disk (Hard Drive) storage media in their original document format or form and have not been encrypted.
c. Documents on the Document Management System (DMS) are still stored on the computer's hard disk (hard drive), and storage media do not have backup data or document backups.
d. Documents in the Document Management System (DMS) are still stored on the computer's hard disk (hard drive), and storage media can be accessed outside the DMS without any validation process.

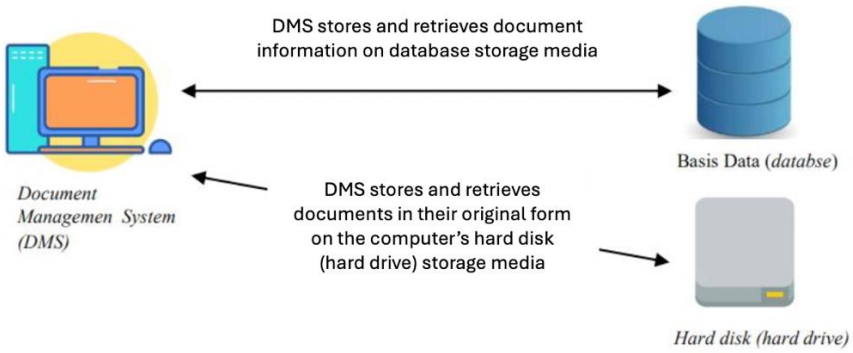

Figure 2. Current System Flow

Based on the results of the analysis of the current system, solutions that can overcome these problems include:

a. Building a document management system using the application programming interface (API) method can be a communication bridge between the document management system (DMS) and the information system running outside the DMS.
b. Change the form or encrypt documents stored in the Document Management System (DMS) using the hashing method with the base64 algorithm.

DOI: 10.31695/IJASRE.2025.2.1

c. Database storage media (database) is also used as a document storage medium, making it easy to provide backup data or document backup.
d. Database storage media can be used as document storage media in the form of changed or encrypted documents so that they cannot be accessed from outside the Document Management System (DMS) without a prior validation process.

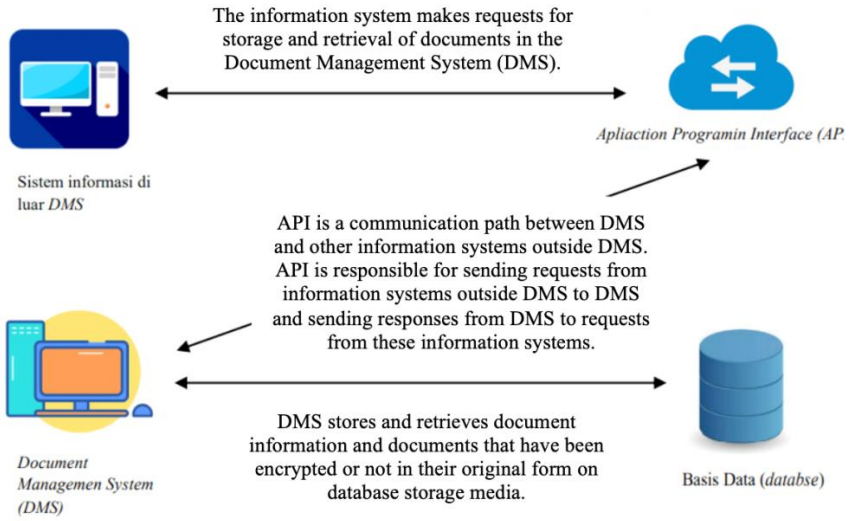

Figure 3. Proposed System Flow

3.2 System Design

The proposed Document Management System (DMS) Use Case Diagram will describe several system interactions with actors involved in the proposed system. The Proposed Use Case diagram is as follows in Figure 4.

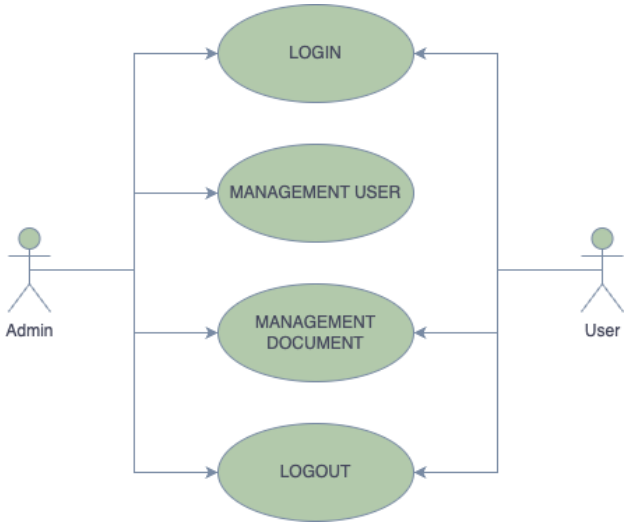

Figure 4. Use Case Diagram Document Management System (DMS)

3.3 Database Design

The database design describes the needs related to the database in building a web application. In the design, tables that already contain the field name needed for storing data that will be used in the system will be displayed. Using Entity Relationship Diagram (ERD) will show the relationship of a table with another table so that the system can process the data optimally. Database design using entity relationship diagram can be seen in the following Figure 5:

DOI: 10.31695/IJASRE.2025.2.1

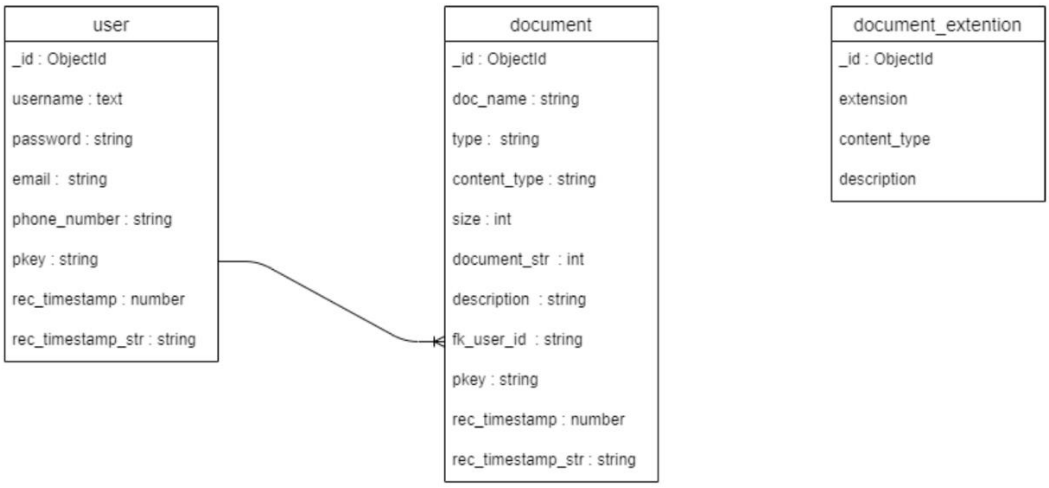

Figure 5. ERD Document Management System

3.4 REST API Design

In implementing API and Hashing methods in the document processing information system (Document Management System), a full API design was carried out for a Representational State Transfer Application. RESTfull API is a communication bridge between DMS and information systems outside DMS in developing DMS by implementing RESTfull API. In Table 2, the API design specifications can be seen.

Table 2. DMS API Specifications

Enpoint API	Description
Put-file	API for adding and changing documents in DMS
Get-file	API to retrieve documents stored in DMS

a. PUT-FILE API Design

In the put-file API, information systems running outside the DMS can send document information to be stored in the DMS and used to change document information stored in the DMS. This API uses a combination of HTTP GET and POST request methods. The HTTP GET method sends the username and password of users registered in the DMS. The HTTP POST method is used to send document information. The description of the put-file API design can be seen in Table 3 below:

Table 3. PUT-FILE API Design Description

Endpoint	**/v1/put-file?username=xxxx&password=xxxxxx**		
Method	**GET, POST**		
Headers	Label	Value	
	Content-Type	multipart/form-data;	
Params	Label	Type	Description
	username	String	username registered on the Document Management System (DMS)
	password	String	password that matches the username sent and registered in the Document Management System (DMS)

DOI: 10.31695/IJASRE.2025.2.1

Body	Label	Type	Description
	doc_name	String	The name of the document to be saved into the system
	document	File	Document files to be saved into the system
	description	String	Description of the document to be saved into the system
	pkey	String	Document key stored in the Document Management System (DMS), filled in if you want to make changes to document data stored before Key of documents stored in the Document Management System (DMS), filled in if you're going to make changes to document data stored before

b. GET-FILE API Design

In the get-file API, information systems outside the DMS can send information to access documents stored in the DMS. This API uses the HTTP GET request method. The HTTP GET method sends usernames, user passwords and document keys registered in the DMS. The HTTP GET method also aims to make the API be called directly in the HTML code that functions to display documents such as images and PDFs. The description of the get-file API design can be seen in Table 4 below:

Table 4. GET-FILE API Design Description

Endpoint	**/v1/get-file?username=xxxx&password=xxxxxx&pkey=xxxx**		
Method	**GET**		
Headers	Label	Value	
	Content-Type	multipart/form-data;	
Params	Label	Type	Description
	username	String	username registered on the Document Management System (DMS)
	password	String	password that matches the username sent and registered in the Document Management System (DMS)
	pkey	String	The document key that you want to access from the Document Management System (DMS). The document key is obtained from the response api /put-file

DOI: 10.31695/IJASRE.2025.2.1

3.5 Interface Design

The information system interface is a communication intermediary between the user and the system. The appearance of this user interface is based on the activity diagram that was designed previously. The design of the interface display can be seen in Figures 6, 7, 8 and 9 below:

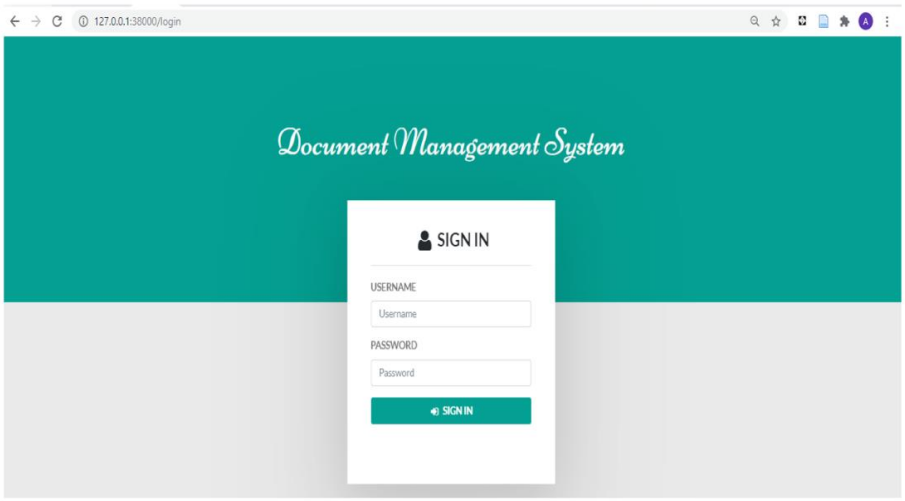

Figure 6. Login Page Interface

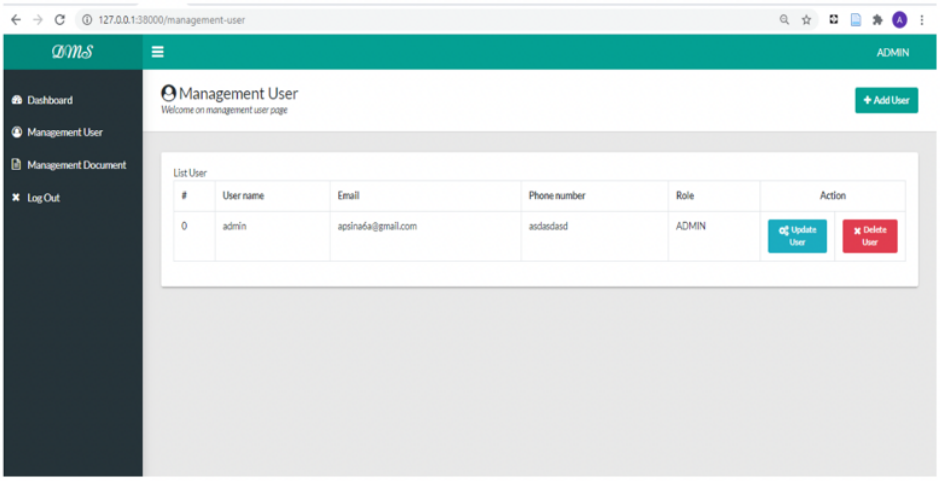

Figure 7. User Management Page

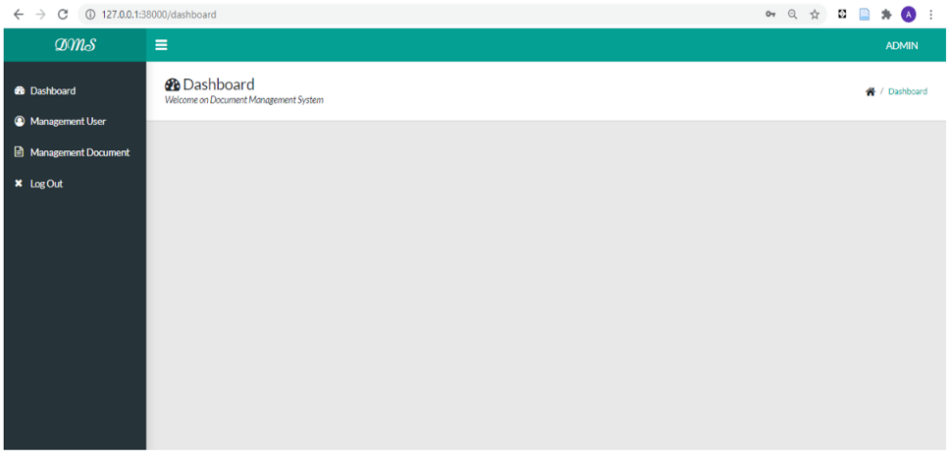

Figure 8. Main Page Interface (Dashboard)

DOI: 10.31695/IJASRE.2025.2.1

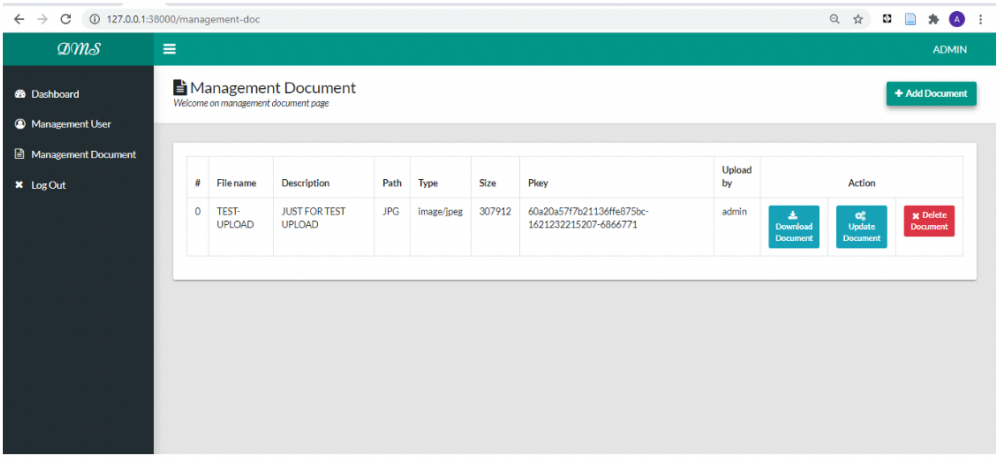

Figure 9. Document Management Page

3.6 Document Management System (DMS) Testing

After testing with the Document Management System (DMS), the test results were obtained and arranged based on the system's functions. The results of the system testing can be seen in the following Table 5:

Table 5. Document Management System Test Results

No.	Test Scenario	Expected results	Test Results
1	Perform the login process using the username and password registered in the system.	The username and password authentication process is successful. The system displays the main page and a successful login notification.	Succeed
2	Perform the process of adding users to the system with the following data: - username: TEST-ADD USER - email : test_add@gmail.com - Phone Number : 0812345010203 - Role : User	The user addition process is successful; the system displays a notification that the addition process was successful and displays the data of the newly added user.	Succeed
3	Performing the process of changing users on the system with data: - username: TEST-ADD USER becomes - username: TEST- UPDATE USER	The user change process is successful; the system displays a notification that it was successful and shows the newly changed user data.	Succeed
4	Performing the process of deleting users on the system with data: - username: TEST- UPDATE USER	The user deletion process was successful, and the system displayed a notification that the deleted user data did not appear again.	Succeed
5	Perform the process of adding documents to the system with the following data: - Document name: TEST-ADD DOCUMENT - Document: TEST-DOCUMENT.JPEG - Description: TEST UPLOAD DOCUMENT FROM PORTAL	When the document addition process is successful, the system displays a notification that it was successful and displays the newly added document data.	Succeed

DOI: 10.31695/IJASRE.2025.2.1

6	Carry out the process of changing documents in the system with data: - Document name: TEST-ADD DOCUMENT - Description: TEST UPLOAD DOCUMENT FROM PORTAL Becomes - Document name: TEST-UPDATE DOCUMENT - Description: TEST UPDATE DOCUMENT FROM PORTAL	The document change process is successful; the system displays a notification that it was successful and shows the newly changed document data.	Succeed
7	Perform the process of deleting documents on the system with data: - Document name: TEST-UPDATE DOCUMENT	When the document deletion process is successful, the system displays a notification that it was successful, and the deleted document data does not appear again.	Succeed
8	Performing the logout process	The user will be logged out of the system, and the system displays the login page.	Succeed

3.7 REST API Testing

The next step is a system test with the REST API, designed and implemented in the previous chapter. RESTful API testing can be done using Postman software. The following are the test results for each API:

a. PUT-FILE API Testing

The API put-file testing flow is carried out with a scenario of trying to add a new document to the DMS. The expected result is that the document is successfully added to the DMS, and the DMS provides a response or feedback that the process was successful with JSON format data and contains information about the document that was added. The implementation of the data sent in the API put-file request can be seen in Table 6 below:

Table 6. Data Request API Put-File

Label	Type	Value
username	Text	admin
Password	Text	qwerty
doc_name	Text	TEST- UPLOAD FROM API
document	File	TEST-DOCUMENT.JPG
description	Text	THIS DOCUMENT UPLOADED BY API
Pkey	Text	None

Detailed responses from the results of the API put-file request test can be seen in Table 7 below:

Table 7. Response API Put-File

Status code http	200
Time	39.38 second
Size	469 Bytes
Type	Json

DOI: 10.31695/IJASRE.2025.2.1

Data	{
	"message_id": "API_CALL_1621841484143124_1411550",
	"message_action": "PROCESS_ADD_SUCCESS",
	"message_desc": "PROCESS ADD SUCCESS", "message_data": {
	"pkey": "60ab566dd7af679d31116337-1621841517804- 1358582",
	"name": "TEST- UPLOAD FROM API",
	"type": "JPG",
	"Content-Type": "image/jpeg"
	},
	"message_title": "RESPONSE"
	}

The document display on DMS added via the put-file API can be seen in Figure 10 below:

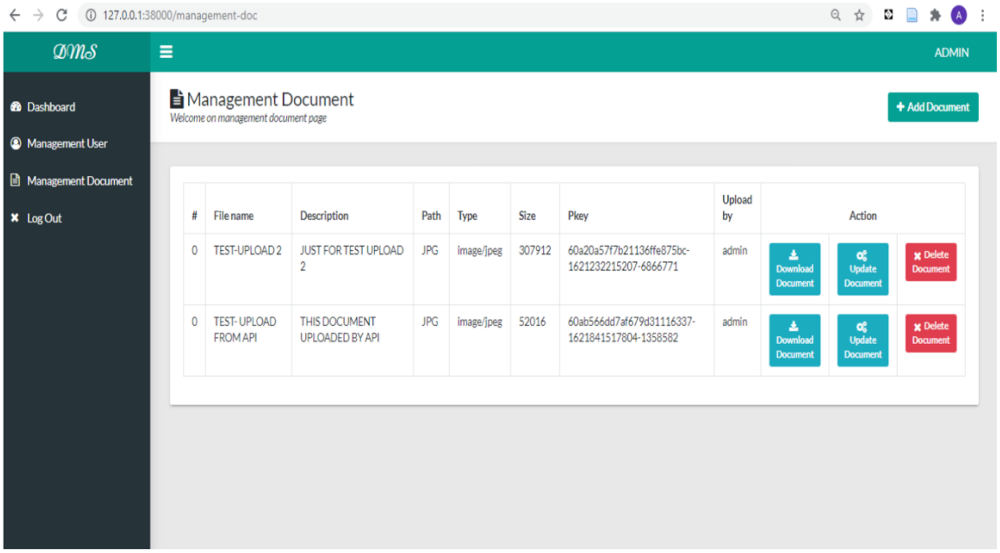

Figure 10. DMS document view

b. GET-FILE API Testing

The GET-FILE API testing flow is carried out with a scenario of accessing a new document added from the PUT-FILE API test above. The expected result is that DMS sends the requested document in the request. The document sent is in the form of binary data. The GET-FILE API request testing is carried out via Postman, browser and html code displaying the document. The implementation of the data sent in the GET-FILE API request can be seen in Table 8 below:

Table 8. Data Request GET-FILE API

Label	Type	Value
username	Text	admin
Password	Text	qwerty
pkey	Text	60ab566dd7af679d31116337-1621841517804-1358582

The document is automatically downloaded for the GET-FILE API test results via a browser in the form of an image document. For the response sent by DMS using a browser, it can be seen in Figure 11 below:

DOI: 10.31695/IJASRE.2025.2.1

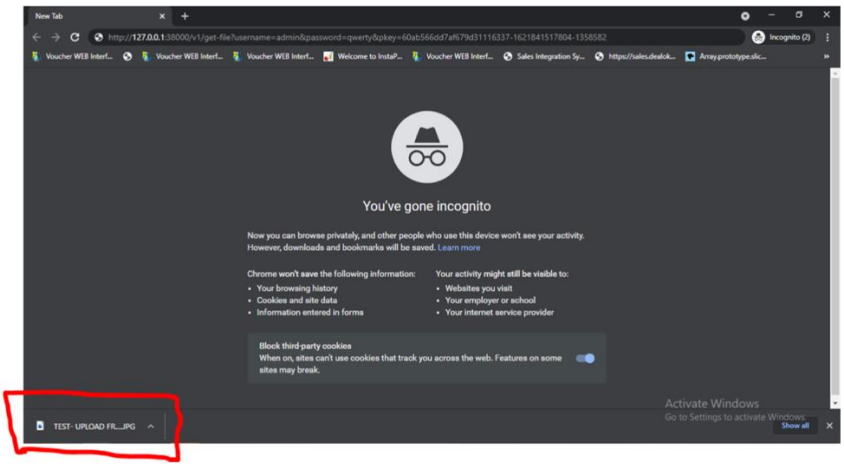

Figure 11. Response GET-FILE API on Browser

4. CONCLUSION

After analyzing the existing system and designing the proposed system, it can be concluded:

a. Document Management System (DMS) development using an Application Programming Interface (API) can connect other running Information Systems with the DMS.
b. The base64 hashing algorithm can be used to change or encrypt the original form of a document into a string so that document storage in a Document Management System (DMS) can use a database storage medium.
c. Storing documents in a database eliminates the need to access documents from outside the Document Management System (DMS) without authentication or validation of document access rights.
d. Storing documents in a database allows users to prepare backup data or document backups.

REFERENCES

[1] S. Sunardi, I. Riadi, and P. A. Raharja, "Analisis Application Programming Interface Pada Mobile E-Voting Menggunakan Metode Test-Driven Development," *Techno (Jurnal Fak. Tek. Univ. Muhammadiyah Purwokerto)*, vol. 20, no. 2, p. 87, Nov. 2019, doi: 10.30595/techno.v20i2.4266.

[2] K. Aryasa and Y. T. Paulus, "Implementasi Secure Hash Algorithm-1 Untuk Pengamanan Data Dalam Library Pada Pemrograman Java," *Creat. Inf. Technol. J.*, vol. 1, no. 1, p. 57, 2015, doi: 10.24076/citec.2013v1i1.10.

[3] P. Prabancono, "Konsep fungsi hash kriptografis," pp. 1–5.

[4] J. B. Sanger, "Desain Dan Implementasi mekanisme Tanda Tangan Dijital Dalam Pertukaran Data Dengan Hash MD5 Dan Enkripsi/Dekripsi Menggunakan Algoritma RSA," *J. Lasallian*, vol. 12, no. 2, pp. 1–12, 2015.

[5] P. Arsih, "Analysis of Document Management Systems Electronic Secret News," *Conf. Senat. STT Adisutjipto Yogyakarta*, vol. 4, 2018, doi: 10.28989/senatik.v4i0.219.

[6] I. Akbar and E. Gautama, "Locating Identification the Nearest Social Security Organizing Agency for Health Recipient Hospital Using the Haversine Formula and Black Box Method," *Int. J. Adv. Sci. Res. Eng.*, vol. 09, no. 03, pp. 25–36, 2023, doi: 10.31695/IJASRE.2023.9.3.4.

[7] F. P. P. Putra, "Pengembangan Sistem Presensi Untuk Work From Home (Wfh) Dan Work From Office (Wfo) Selama Pandemi Covid-19," *J. Sains, Nalar, dan Apl. Teknol. Inf.*, vol. 1, no. 2, pp. 66–74, 2022, doi: 10.20885/snati.v1i2.9.

[8] N. Falih and Sarika, "Sistem Kehadiran Mahasiswa Menggunakan Qr Code Berbasis Restful Api," *JIRE (Jurnal Inform. Rekayasa Eletronika)*, vol. 3, no. 2, pp. 120–128, 2020.

[9] M. Mujito and A. Bagus Susilo, "Aplikasi Kriptografi File Menggunakan Metode Blowfish dan Metode Base64 pada Dinas Kependudukan dan Pencatatan Sipil Kota Tangerang Selatan," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 5, no. 2, pp. 54–60, 2016, doi: 10.32736/sisfokom.v5i2.39.

[10] R. Tinendung[1], S. Khairani[2], and S. Sundari[3], "Aplikasi Messaging dengan Algoritma Base 64 untuk Mengamankan Data Pesan Berbasis Web," *Algoritm. J. Ilmu Komput. dan Inform.*, vol. 06, no. 01, pp. 104–111, 2022.

DOI: 10.31695/IJASRE.2025.2.1

[11] N. F. Ginting and M. Ginting, "Perbandingan Kriptografi RSA dengan Base64," *Jtiust*, vol. 2, no. 2, pp. 2548–1916, 2017.

[12] E. Gunadhi and A. P. Nugraha, "Penerapan Kriptografi Base64 Untuk Keamanan URL (Uniform Resource Locator) Website Dari Serangan SQL Injection," *J. Algoritm.*, vol. 13, no. 2, pp. 391–398, Feb. 2017, doi: 10.33364/algoritma/v.13-2.391.

Correspondence author: Elliana Gautama, elliana@perbanas.id

DOI: 10.31695/IJASRE.2025.2.1