Afsa Mifzal Zorarghirfar
001202200150
Assignment 4 – Research Methodology

Improving Document Data Integrity and Management in the CryptDocs Web-Based System Using SHA-256 Hashing and JWT

Methodology

This methodology section describes the systematic approach used to develop and evaluate a web-based document management system named CryptDocs. It outlines the type of research, overall strategy, tools and technologies used, architectural design, implementation process, and testing methodologies used to achieve the intended objectives.

3.1 Types of Research

This research can be categorized as applied research with strong emphasis on development. The goal is not only to contribute theoretical knowledge but also to create a practical, functional, and secure web-based solution that directly addresses issues related to the integrity and management of internal document in a corporate environment. The development process is design-drive, beginning with problem analysis, followed by system design, implementation, and testing.

3.2 Approach

The overall strategy used in the CryptDocs project is a prototyping approach integrated with agile development principles. This iterative methodology promotes continuous improvement and rapid feedback loops, ensuring that the system evolves quickly from an initial concept into a functional prototype, which is refined progressively. This approach is well-suited for the development of complex systems that require balance between user-friendly front-end experiences and secure back-end functionalities

3.3 Methodological Suitability

The methodology used is well-suited to the research problem for several reasons. Given the dynamic nature of web technologies and the continuous advancement of security threats, the iterative prototyping approach provides the flexibility needed to adapt to new requirements and integrate current best practices in both security and UI/UX design. This approach allows early identification of technical challenges and their solution, thereby reducing risks commonly associated with large-scale software development. Furthermore, CryptDocs prioritize ease of use and user interaction, this development strategy supports continuous validation of the user interface and core functionalities, ensuring that the final product is aligned with real-world corporate document management and integrity needs.

3.4 Hardware and Software

The development environment includes the following components:

- **Hardware**
  - Laptop / Desktop PC with Intel Core i5 or higher, 8 GB of Ram or higher
  - Wi-Fi connection
  - Encrypted storage system
  - Cloud-based or on-premises server for staging and deployment
- **Software**
  - Visual Studio Code
  - XAMPP (MySQL)
  - Node.js + Express.js
  - JSON Web Token (JWT)
  - HTML5, CSS, JavaScript

3.5 Architecture Design, Model, and Algorithm

The CryptDocs system follows a Client-Server architecture based on Model-View-Controller (MVC) design pattern, adapted for a web-based application structure. It consists of three main layers:

- Frontend: Developed using HTML, CSS, and JavaScript to display and manage user interface
- Backend: Implemented with Node.js and Express.js framework, serving as API layer to process frontend requests, perform SHA-256 hashing, and manage authentication.
- Database: Utilizing MySQL to store persistent data, including user credentials, document metadata, and activity log.

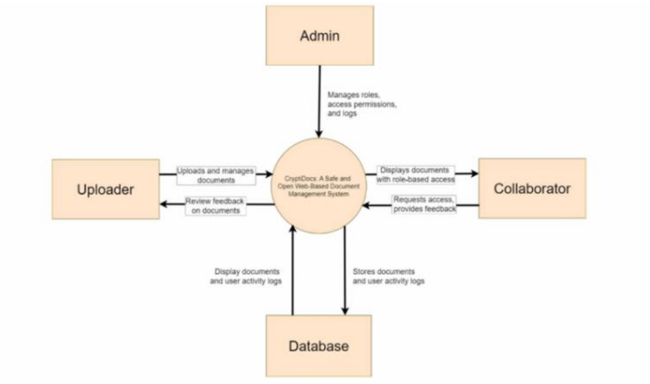
Admin
Manages roles, access permissions, and logs
Uploader
Uploads and manages documents
Review feedback on documents
CryptDocs: A Safe and Open Web-Based Document Management System
Displays documents with role-based access
Requests access, provides feedback
Collaborator
Display documents and user activity logs
Stores documents and user activity logs
Database

Figure 3.1 Architecture Diagram

In *Figure 3.1* illustrate the CryptDocs overall system interaction with external entities. This diagram shows how the Uploader interacts with the system to handle and upload documents.

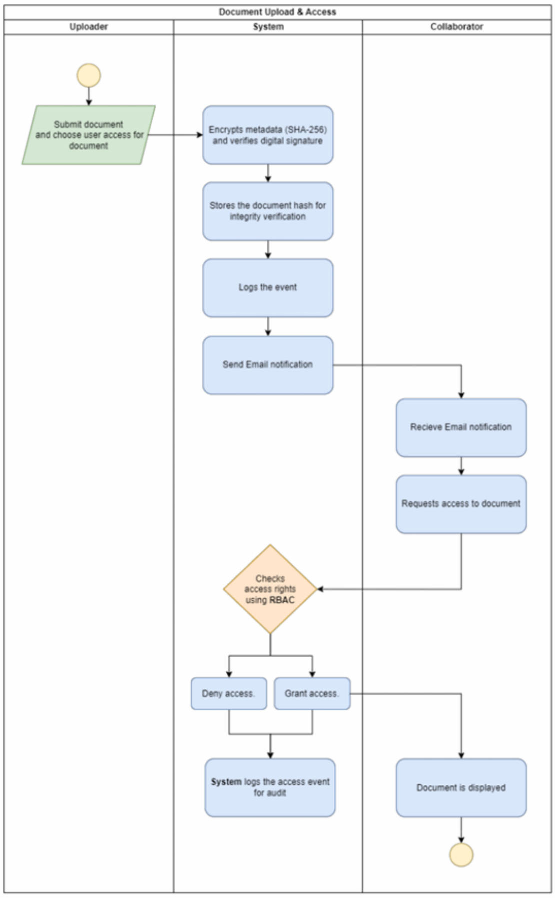
Document Upload & Access
Uploader
System
Collaborator
Submit document and choose user access for document
Encrypts metadata (SHA-256) and verifies digital signature
Stores the document hash for integrity verification
Logs the event
Send Email notification
Recieve Email notification
Requests access to document
Checks access rights using RBAC
Deny access.
Grant access.
System logs the access event for audit
Document is displayed

Figure 3.2 Workflow Diagram

Figure 3.2 shows the flow process of "Document Upload & Acess" including the hashing mechanism and the steps involved in accessing the document.

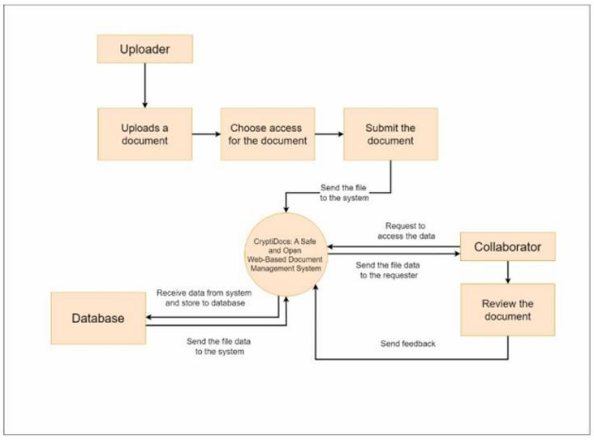
Uploader
Uploads a document
Choose access for the document
Submit the document
Send the file to the system
CryptiDocs: A Safe and Open Web-Based Document Management System
Request to access the data
Collaborator
Send the file data to the requester
Review the document
Send feedback
Receive data from system and store to database
Database
Send the file data to the system

Figure 3.3 Data Flow Diagram

Figure 3.3 displays the detailed Data Flow Diagram (DFD) and outlines the main processes involved in the upload and document access functions within CryptDocs. The diagram illustrates how the Uploader initiates the process by uploading a file, which is then processed by the CryptDocs system. It also shows how a Collaborator sends a request and receives access to the document, including the flow of data back to the user. Lastly, it depicts how file data is stored persistently and retrieved from the database.

3.6 Implementation Detail

This solution was implemented using JavaScript across the entire stack, with Node.js and Express.js for the backend, “mysql2” for database access, and JWT and bcrypt for authentication. The frontend was built using HTML, CSS, and JavaScript. Challenges such as integration of dynamic database, CSS layout conflicts, and JWT session management were addressed through iterative debugging.

3.7 Data Collection Process

For this research, data collection involved the creation and management of synthetic data within the CryptDocs system, including:

- User Data: Dummy user accounts (Employee ID, email, password, role) were registered directly through the registration page.
- Document Data: Various types of documents (PDF, DOCX, XLSX. etc) were manually uploaded through the system’s upload interface. Document’s metadata (sender, receiver, division, comment) inputted manually by user. File contents were stored on the server, while SHA-256 hashes saved in database.
- Activity Log Data: All user activities automatically recorded by the backend system.

3.8 Testing and Evaluation

The developed CryptDocs system underwent comprehensive testing to validate its functionality and non-functional attributes.

- Functional Testing:
  Functional testing was conducted using a black-box testing approach to verify that all main features such as registration, login, upload, sending, receiving worked as planned. Test cases were designed to cover a wide range of usage scenarios and input validation to ensure consistent and expected behavior.

- Non-functional Test:
  Non-functional testing focused on evaluating system quality aspects that are not directly tied to functionality but are crucial to overall performance and user experience. All tests indicated that the CryptDocs system passed key parameters successfully.

Parameter	Requirement	Expected Result	System Response	Status	Explanation
Performance	System should be able to hold up until 10.000 document	The system remains stable even when a large number of users or actions are performed.	The system functions effectively even after numerous actions are tested.	Pass	Multiple documents being uploaded, viewed, and users logging in simultaneously don't affect the system's functionality.
Reliability	The system should be able to recover quickly from unexpected failures or disruptions and maintain availability.	If there's an error the system should recover by itself,	The system uses MySQL with proper error handling and monitoring to ensure availability.	Pass	The system never went down after multiple checks.
Compatibility	The system should be able to use in desktop browser.	The features and system display keep up working throughout all browsers.	Interface and features work well in every browser.	Pass	Already tested in Windows and Mac, and all the features works properly

Response Time	Each feature needs to respond in less than 5 seconds	Uploading, opening document, dashboard access should not be slow to respond	The system responds in under 5 seconds consistently.	Pass	Accessing dashboard or uploading files are consistently responds in under 5 seconds and there’s no delay
Safety	SHA-256 hashing, role-based access, encryption, and activity logging must all be implemented in the system to guarantee security.	The data or files must be encrypted, user access must be managed through roles, and all activities must be consistently recorded in activity log	To guarantee data integrity and tracking, the system utilizes the use of encryption and fixed logging.	Pass	The system already implements SHA-256 hashing, JWT token, and all activities recorded in activity log.

3.9 Limitation

This research successfully developed and tested CryptDocs, but security testing was limited to hashing and JWT authentication. Advanced penetration tests and large-scale performance evaluation were not conducted. Testing was done in a local environment with limited users and data. Additionally, the system lacks integration with enterprise tools or cloud storage, which limits its scalability for production use.

REFERENCES

Aldya, A. P., Rahmatulloh, A., & Arifin, M. N. (2019). Stateless authentication with JSON Web Tokens using RSA-512 algorithm. *Jurnal Infotel, 11*(2), 36. https://doi.org/10.20895/infotel.v11i2.427

Ariska, N. (2022). Implementasi fungsi hash dengan algoritma SHA-256 pada aplikasi duplicate image scanner. *Jurnal Sains dan Teknologi Informasi, 1*(4), 112–120. https://doi.org/10.47065/jussi.v1i4.2292

Gunawan, R., & Rahmatulloh, A. (2019). JSON Web Token (JWT) untuk authentication pada interoperabilitas arsitektur berbasis RESTful Web Service. *Jurnal Edukasi dan Penelitian Informatika (JEPIN), 5*(1), 74. https://doi.org/10.26418/jp.v5i1.27232

Herman, H., Wijaya, R., Miharja, S., & Wilson. (2022). Implementasi algoritma AES-128 dan SHA-256 dalam perancangan aplikasi pengamanan file dokumen. *Jurnal TIMES, 10*(2), 80–87. https://doi.org/10.51351/jtm.10.2.2021666

Hutagalung, J., Ramadhan, P. S., & Sihombing, S. J. (2023). Keamanan data menggunakan Secure Hashing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada digital signature. *Jurnal Teknologi Informasi dan Ilmu Komputer, 10*(6), 1213–1222. https://doi.org/10.25126/jtiik.1067319

Irfan, Z. N. (2024). *Implementasi Secure Hash Algorithm 256 (SHA-256) dan algoritma Rivest Shamir Adleman (RSA) untuk mendeteksi perbedaan pada dokumen PDF* [Tugas akhir, UPN "Veteran" Yogyakarta].

Komputer, S., Syahroni, A. W., Dewi, N. P., Ramadhani, N., & Said, B. (2024). Uji keamanan back end aplikasi berbasis website menggunakan metode Black Box Testing. *Jurnal Ilmiah Komputer, 19*(2), 215–226.

Loui Pattinama, Y., Ferdiansyah, Susanti, I., & Painem. (2023). Implementasi REST API web service dengan otentifikasi JSON Web Token untuk aplikasi properti. *Informatik: Jurnal Ilmu Komputer, 19*(1), 81–89. https://doi.org/10.52958/iftk.v19i1.5724

Mahfud, I., & Hadi Utomo, P. (2022). Implementasi sistem kriptografi RSA signature dengan SHA-256 pada mekanisme autentikasi REST API. *Prosiding Seminar Nasional Teknoka, 6*(2502), 84–92. https://doi.org/10.22236/teknoka.v6i1.431

Manish Rana, E., et al. (2023). Enhancing data security: A comprehensive study on the efficacy of JSON Web Token (JWT) and HMAC SHA-256 algorithm for web application security. *International Journal on Recent and Innovation Trends in Computing and Communication, 11*(9), 4409–4416. https://doi.org/10.17762/ijritcc.v11i9.9930

Maulana, R., Teknik, S., & Cryptography, A. (2023). *Cryptographic analysis of JSON Web Token*.

Nadimsyah, A. Z., Ezar, M., & Rivan, A. (2024). Implementasi Secure Hash Algorithm-256 dan Advanced Encryption Standard untuk verifikasi tanda tangan digital. *Jurnal of Secure Hashing, 6*(1), 229–239. https://doi.org/10.47065/josh.v6i1.5942

Nainggolan, S. (2022). Implementasi algoritma SHA-256 pada aplikasi duplicate document scanner. *RESOLUSI: Rekayasa Teknik Informatika dan Informasi, 2*(5), 201–213. https://djournals.com/resolusi

Nashikhuddin, A. Y., Karaman, J., & Litanianda, Y. (2023). Implementasi API RESTful dengan JSON Web Token (JWT) pada aplikasi e-commerce Thrifty Shop untuk otentikasi dan otorisasi pengguna. *METHOMIKA: Jurnal Manajemen Informatika dan Komputerisasi Akuntansi, 7*(2), 239–246. https://doi.org/10.46880/jmika.vol7no2.pp239-246

Ngemba, H. R., Ifandi, -, Hendra, S., & Dwi Arsana, I. G. N. A. K. (2023). Implementasi enkripsi data MD5 dan SHA-256 pada sistem informasi peminjaman buku tanah. *Techno.Com, 22*(3), 654–664. https://doi.org/10.33633/tc.v22i3.8299

Priyatna, R., & Waluyo, S. (2022). Implementasi RESTful dengan JWT untuk booking barang di Primajaya Multisindo. In *Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)* (pp. 1040–1047).

Ramdani, F. C., Rahmatulloh, A., & Shofa, R. N. (2023). Implementasi JSON Web Token pada authentication dengan algoritma HMAC SHA-256. *SISTEMASI: Jurnal Sistem Informasi, 12*(1), 194–205. http://sistemasi.ftik.unisi.ac.id

Shah, N. H. M., Asmawi, A., & Yasin, S. M. D. (2025). Improving collection of data type evidence and the integrity of evidence collected using SHA-256 hashing algorithm for web browsers. *Journal of Theoretical and Applied Information Technology, 103*(2), 375–383.

Shofiana, A., Mardiansyah, S. S., & Parabi, M. I. (2024). Implementasi REST API menggunakan JSON Web Token (JWT) pada sistem monitoring KPI berbasis mobile di PT Industri Kereta API (Persero). *Jurnal Teknologi dan Informasi, 12*(2), 101–111.

Wijaya, B. M. E., & Sutanto, F. A. (2023). Implementasi sistem autentikasi JSON Web Token pada aplikasi Fieldrent menggunakan algoritme SHA-512. *Progresif: Jurnal Ilmiah Komputer, 19*(2), 901–911. http://ojs.stmik-banjarbaru.ac.id/index.php/progresif/article/view/1367