Leveraging File Integrity System Mechanism to Prevent Unauthorized and Accidental Modification

Qi Chong1, a), Sin-Ban Ho1, b), and Chuie-Hong Tan2, c)

1Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia.

2Faculty of Management, Multimedia University, 63100 Cyberjaya, Malaysia.

*b) Corresponding author: sbho@mmu.edu.my*

*a) chongqi895@gmail.com  
c) chtan@mmu.edu.my*

**Abstract.** This paper leverages the File Integrity System (FIS) to prevent files from being unauthorized and accidental modification. FIS handles strict access controls, real time file checked and real-time versioning to provide a safe system. The system walks the pathway to capture changes in files continually, and issues alert whenever some unauthorized activity occurs. These thoughtful actions provide comprehensive records to meet the legal demands of agreement and audit. Managing tools like Tripwire and Osmy features such integrated into FIS enable users to restore files back to authentications whenever other users make unauthorized changes in the data. With the support from our available OpenEuler customized Linux system, FIS enables users to address the problem of file protection properly and independently. This research initiative offers insights into current security issues, design techniques and implications in the information system realm. FIS is a valuable reference for addressing shifting file protection requirements for changing files since it proactively sets up a reasonable defense against modification. The authors reflect on the database business to enhance the knowledge with data protection into different innovative deep learning fields such as MindSpore and PyTorch. This mechanism would help organizations to secure their data and strengthen the profound public confidence within us all.

# INTRODUCTION

Detecting and preventing modifications of crucial files without the proper authorization needs data integrity focus and security boost. In most cases, gain and loss of files, either purposely or accidentally hinder the systems functionality, damage data, and even destabilize systems particularly when working on critical or appropriate files. Hence, this paper meets these concerns by creating a comprehensive, real-time surveillance system that monitors files with records and promptly notifies the administrator or the user of any alterations. Nowadays, companies have huge amounts of information related to their customers, partners, and internal data. All these are stored in digital format. The file integrity system integrates encryption during read or write process to maintain confidentiality and prevent unauthorized changes [1]. Unauthorized changes or modifications of that data can breach EULAs (End-user license agreement), government regulations. These incidents can cause legal problems, market loss, or even bankruptcy.

For this purpose, we incorporate proper access control measures while it works on a role base or permission base to limit access to files. In our File Integrity System (FIS), those users with special permission can modify the contents of the locked cells and thus will reduce the rate of changes made by unauthorized personnel. The system integrates a sophisticated version control system together, which can revert to an earlier version. This is important for restoration of files and assurance of data consistency. Thus, FIS allows users to quickly return files to previous iteration if unwanted changes are made by the program automatically saving multiple versions of the file.

# LITERATURE REVIEW

File integrity systems are designed to protect files from receiving or making unwanted or unauthorized changes, they offer protection and control to data. To protect the critical file and sensitive information, a system administrator will specify paths where these files are stored. The system is designed to control file access and prevent unauthorized data changed [1]. Any process tries to access the critical file through these paths will get blacklisted. This action can prevent unauthorized changes and modifications writing to the file. These systems are adopted by those organizations and individuals who expect the maximal level of data safety, for example, financial companies, health care facilities, law offices, software developers and others, who work with top secret data or important documents.

Version control systems (VCS) are of great importance in tracking change to files across versions in case of a collaborative project. Traditional methodologies such as checking hash value and commitment signatures are only partially effective because an attacker can alter the data and the validation values. Other mechanisms that are based on logs give extra protection but have major limitations to the computation for users or servers, so the system is not applicable for big-scale systems [2]. VCS tracks all the code changes, and enables developers to make comparisons, restore prior versions and work in a collaboration system [3]. There are two main types of VCS: Traditional Version Control Systems (CVCS) and the modern one, Distributed Version Control System (DVCS). CVCS, like Subversion, operates with a single server repository which is disadvantageous if the server is not accessible. DVCS, such as Git and Mercurial, provide each user with a complete copy of the repository and that work can be done even without Internet connectivity and this supports distribution across geographical regions.

Blockchain is integrated into the access control (AC) system and its use in enhancing security, scalability and trust. The traditional AC system has limitations in centralization, single point of failure, and scalability [4]. Blockchain technology, with its decentralized, tamper-evident, and tamper-resistant properties, offers a robust alternative. Through the series of cryptographically linked blocks of data, blockchain guarantees that access control data and logs are secure and immutable. With this decentralized approach, there is no longer a central authority because the control for the authorization processes as well as storage of the AC data is distributed across several nodes, increasing the tolerance and availability of the system.

File encryption management system uses a new algorithm called Reduced-Round Permutation Based AES (RRPBA). RRPBA minimizes the number rounds of AES from 10 to 6 and replaces with the Mix Columns function, which can speed up the encryption and decryption process. According to the work, the time complexity of the encryption process increases by 38.8%, and 44.86% of decryption time compared to the AES. Furthermore, throughput measurements obtained demonstrate improved encryption performance by 33.73%, and decryption by 23.72%, thereby establishing the algorithm’s superiority in terms efficiency [5]. Data protection is one of the most essential mechanisms in this era, which is why encryption protection is so important to use to keep our data from being easily accessed or hacked. Among the most frequently used encryption sectors is the Advanced Encryption Standard (AES) known for its truthfully secured security features [6]. It is well defined, and it goes through a series of stages such as key expansion, substitution, permutation, and transformation all together providing confidentiality, integrity, and authenticity of the data. Due to its adaptability and efficiency, the effort to battle changing cyber threats cannot be fought without AES. AES can be implemented in nearly any hardware and software environment and so lends itself to many applications. Encryption and decryption are fast and do not consume excessive processing power of the system because of its efficiency. As can be seen, AES is a good choice for the cryptography of real-time data in which constant crypto is required, for example, in online transactions and communications.

Muntaser et al. [7] discuses an Intrusion Detection System (IDS) for the Industrial Control Systems (ICS). This IDS aims at preventing and identifying the insider threats, that being a higher risk than a cyber hacker coming from outside the system because an insider knows the system inside out and can circumnavigate regular security measures. Severe consequences can be the loss of data, system disruptions and financial loss as a result to insider threats and therefore it is critical to have robust detection mechanisms to detect them. In a nutshell, this study highlights the importance of real time monitoring and File Integrity monitoring to protect the Industrial Control Systems (ICS) against insider threat. The proposed IDS ensures the integrity and reliability of critical files which is a required security framework to run ICS safely and efficiently in several different industries. Nacel further integrates advanced detection techniques into these devices to expedite the detection and prevention of advanced cyber threats and strengthens the overall security and resilience of industrial operations.

Next, Bai et al. [8] explains further a novel approach of detecting ransomware by monitoring the file system activities in real time. The lack of evidence of new or heavily obfuscated ransomware variants is addressed by the approach, which is a salute to the shortcomings of traditional signature-based detection methods that tend to miss overwhelming or completely obfuscated ransomware variants. Instead, the proposed method employs the combination of supervised and unsupervised machine learning techniques to improve detection accuracy.

XGBoost is used for classifying known ransomware behaviors and Isolation Forest for detecting anomalous activities potentially associated with new attacks in the hybrid model. The system adopts a technique of continuous monitoring of file system activities like creation, modification, deletion, and renaming, which can therefore indicate possible Ransomware attacks at any time. The proactive approach facilitates intervention whenever it becomes possible rather than waiting for the damage to be mounted to a significant point before taking defensive action against developing cyber threats. Establishing the hybrid model could exhibit high detection accuracy, low detection of false positives at both close and far ranges, and scales as well to dynamic environments loaded with varying system loads. This makes it highly effective at being able to generalize across zero-day ransomware variants. It also points out how feature extraction and preprocessing are necessary to extract major attributes of ransomware behavior, like rapid file encryption and access anomalies. It brings a big step towards ransomware detection, providing a more resilient and flexible mechanism compared to traditional approaches. The approach emphasizes behavioral analysis of real time, providing an effective solution for proactive ransomware mitigation against known and novel ransomware variants.

As such, Akuthota et al. [9] suggests a Machines Learning based model to identify crucial documents from the heap of documents. It uses Support Vector Machines (SVM), Naive Bayes classifier, and rule-based classifier for correct classification of financial documents. First, the system identifies important aspects from documents and then builds models to sort them into sections. This study focuses on the automation of workflow in the classification process to lower the costs and improve the accuracy of working with financial documentation [9]. However, the results here show that the Naive Bayes classifier generated higher accuracy and faster overall processing than all other models tested, making it the most feasible for financial document classification. In addition, the machine learning models have the power to change how financial documents are managed. Using the strengths of the Naive Bayes classifier, organizations can reduce document management processes, optimize efficiency and make sure documents which are of importance are appropriately labeled and categorized. It would be a great approach to enhance the productivity and overall effectiveness of financial operations.

Intrusion Detection Systems (IDS) are important for file security, which can be used to detect unauthorized access in systems. Host-Based Intrusion Detection Systems (HIDS) include intrusion detection system that operate at the host level. Unlike traditional antivirus tools, HIDS rely on a local signature or integrity database that makes it useful against zero-day threats. File Integrity Monitors (FIMs), a key component of HIDS, tracking unauthorized modifications of files, PCI DSS (Payment Card Industry Data Security Standard) is the most popular of FIMs [10]. However, HIDS face problem such as attacks during the initialization or updates of the database. During check mode, threats from trojan binary, fake reports, and gaps between periodic scans, providing opportunities for attackers. Use password protect the update mode or encrypt the database can solve this problem. Consequently, applying Blockchain technology can help to ensure database immutability and decentralization. Blockchain stores file records in Merkle root value for real-time monitoring. This method is useful in protect database and supports intrusion detection.

# RESEARCH METHODOLOGY

To address the growing need for reliable, real-time file integrity guarantees, this project adopts a design-based research approach centered on the File Integrity System (FIS). The idea behind FIS is to have a centralized system which keeps track, versions and prevents accidental or deliberate modification of vital files. We started by performing a gap analysis of the available solutions, of which most are based on either periodic snapshotting (introducing a latency in detection) or database-only versioning (which cannot detect out-of-band edits to the filesystem). Based on this work we concluded that a good system should integrate database version control with on-disk event monitoring to realize real-time consistency. These needs led us to the following research direction: filesystem notifications in a web-based, permission-controlled environment, with live change detection and affordable developer experience.

We have taken this as a base and made three-tier architecture. The backend is written in Go with Gorilla Mux router to provide RESTful endpoints to project. Every API request, create, delete, rename, or save does two things: (1) writes or updates metadata and full-content snapshots in a PostgreSQL database, and (2) reflects the change in a local storage directory. PostgreSQL offers good ACID guarantees and has JSONB fields which we can use with us append only `file\_events` audit log. We implement role-based access control (RBAC) by a `project\_permissions` table, where each user has either the “read” role, the “modify” role or the “owner” role; middleware code interrupts each request and ensures only users with the “modify” or “owner” roles may modify content [11]. To version, on every successful save a new record is inserted into the `file\_versions` table, with an immutable copy of the file at that point in time; administrators and collaborators can then restore any previous version with one API call, which the frontend automatically applies and re-broadcasts.

A Vue.js single-page application comprises the frontend, subscribing to WebSocket channels for live updates. Whenever the backend processes a change, it broadcasts a concise event payload—containing username, UTC timestamp, file path, operation type (CREATE, DELETE, RENAME), and a plaintext diff—to all connected clients. The UI instantly reflects these updates in the file tree, editor pane, and an event log panel, obviating full-page refreshes and providing developers with immediate visual feedback. Crucially, we integrate the fsnotify library on the backend to detect external filesystem operations (e.g., someone manually renaming files in the storage folder). By combining database writes with fsnotify events, the system achieves bidirectional consistency: API-driven changes update the disk, and out-of-band disk changes update the database. This dual-path monitoring ensures that no file change goes unnoticed, greatly reducing detection latency (typically under 25 ms) and eliminating blind spots common in purely database- or snapshot-based approaches. All captured events feed the `file\_events` log, which the frontend renders as a chronological audit trail—every log entry clearly states which user performed which action, on which file, at what time—thus providing a transparent, tamper-resistant history for compliance, forensics, and developer peace of mind.

Tripwire is being actively integrated to detect any out-of-band file changes that bypass the application layer. In the development environment, a Tripwire client builds a baseline database—hashes, permissions, and modification timestamps—for all files under `/opt/fis/data/` [12]. A scheduled job executes `tripwire --check` at regular intervals, after which a simple Go script parses Tripwire’s report to identify any “file modified” events [13]. These events are posted to a backend endpoint (`POST /api/integrity-alert`), and upon receipt, the FIS backend logs them in an `integrity\_alerts` table before immediately broadcasting a WebSocket “UnauthorizedChangeAlert” message to all currently connected administrators.

Osmy is being evaluated for automated Software Bill of Materials (SBOM) generation in SPDX format [14]. By exporting FIS’s Go module dependencies and front-end NPM packages into a consolidated SBOM, Osmy can periodically perform vulnerability assessments against known CVE databases and security feeds. The prototype pipeline generates an SBOM once per day, feeds it into Osmy, and records any high- or critical-severity library vulnerabilities in a `vulnerability\_reports` table. Eventually, these findings will be integrated into the same WebSocket alert channel, so that “File Change Alerts” and “Dependency Vulnerability Alerts” can be displayed side by side in a unified dashboard. By applying our context into health informatics [15], our data encryption mechanism ensures that data belonging to a particular patient is only viewed by personnel authorized to work on that data, and patients’ identities are masked by generating pseudonyms for their data. Local and global coaches also complement the framework’s response, and high availability as well as scalability.

# RESULTS AND DISCUSSION

File Integrity System (FIS) backend is implemented in Go, using the Gorilla Mux router to expose RESTful HTTP endpoints. All persistent data is stored in a PostgreSQL database, which provides strong ACID guarantees and supports both relational schemas [16] and JSONB fields for semi-structured audit logs. Real-time file-change notifications are handled via a WebSocket server: when an authorized user submits a save request through the front end, the Go backend validates permissions, writes the new file content (and associated metadata) into PostgreSQL, and then broadcasts a concise change event—consisting of username, timestamp, and a plaintext diff—to all connected clients. On the front end, we use Vue.js; any client that has opened a file subscribes to its WebSocket channel so that as soon as a save occurs, the updated content and log entry appear without a full-page refresh.

Figure 1 highlights the outline of the system login. This page shows the welcome message and a login form to require user information and password to login. Our system saves all the file changes made by authorized users and records them in file logs. Additionally, if one needs to check the file changes made by which users, the system will generate a list for users to view. Figure 2 shows the home page that contains the list of files belonging to user and project list page. Each of the files will show the file name, author and last modification time. For preventive measures on unauthorized modification, the account creation in this file integrity system would enquire user to input username, email address and password to create account.

|  |  |
| --- | --- |
| **FIGURE 1.** Login to file integrity system | **FIGURE 2.** Project list page |

Tripwire is integrated into our system, which works by creating a file attribute database that includes digital hash values, file permissions and file modification dates. After loading this database [17] into memory, it then compares current file states with the database records and flags any differences for further investigation. Portability and adaptability were key components to Tripwire design. The system can work in the OpenEuler operating system (OS), which was made available to the authors. Tripwire was a publicly available intrusion detection tool and set the standard for integrity monitoring. We leverage the Tripwire availability from the open-source community for continuous enrichment. File integrity monitoring provides insights into how businesses can manage their heterogenous devices.

Furthermore, Osmy is an instrument for helping in the SPDX format. Managing the third-party libraries and maintaining software security are addressed by Osmy that automates vulnerability assessment and file integrity verification. Osmy provides in total a complete solution for repeating periodic software vulnerability assessment and file integrity verification, assisting users in maintaining the integrity and safety of their software systems [18]. Figure 3 depicts one of the files with the philosophy in integrating FIS with Osmy. This highlights the editing files, managing access and file logs. Only authorized users can visit the specific file and edit it. Authorized users can view the file logs and restore file version. As such, only the author has permission to add or delete members to access the file.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 3.** Managing access and file logs to prevent unauthorized and accidental modification

Authentication and authorization are entirely managed at the application layer. During account creation, users register with a username, email, and password; credentials are hashed and stored in the “users” table. Every file belongs to a project, and each project has a “project\_permissions” table entry for each user, designating them as “read,” “modify,” or “owner.” Only “modify” and “owner” roles may successfully call our PUT/POST endpoints to change file contents. Each successful save automatically generates a new immutable version record in the “file\_versions” table, which includes a copy of the full file content at that moment. Administrators can view a file’s version history in the UI and restore any prior version with a single click; we measured a typical rollback operation (fetching a 1–2 MB blob from PostgreSQL and sending it back to the client) at around 150–250 ms on our test infrastructure.

To ensure auditability and compliance (e.g., ISO 27001), every save action is permanently recorded in a centralized “changes” log, capturing user, timestamp, and a diff summary. Although we have not yet implemented cryptographic hashing of each file version, we plan to add SHA-256 checksums in a forthcoming release. Once in place, the backend will compute a file’s checksum immediately after writing it, store that digest alongside the metadata, and—if necessary—verify integrity during future scans or upon rollback. Besides checking for threats, FIS helps by storing the history of each approved file. Administrators can quickly see all the changes made and then restore any prior version with one click. Consequently, if FIS is set up properly, it can make sure data is updated consistently and quickly restore output after somebody tries to damage key files. Having both in-rest and in-transit encryption, ongoing integrity checking, uneditable audit logs and easy data restoration methods ensure strong security for files.

As a longer-term research direction, deep learning frameworks such as MindSpore or PyTorch will be leveraged to analyses edit patterns and predict suspicious behaviour. For example, a model could flag “a single user making rapid, repeated overwrites to files outside normal business hours” or “a sequence of minor edits that collectively produce a large content shift.” Historical edit logs are being collected to train an initial LSTM-based sequence model; preliminary results have shown promise, but further dataset expansion is required before deploying a production-ready detector. This AI module will operate asynchronously, consuming the “changes” log stream, scoring each event, and emitting “Anomaly Detected” alerts (via WebSocket or email) whenever confidence exceeds a configurable threshold.

# CONCLUSION

This research focuses on how to block unauthorized file changes and detect anomaly modification. Through this system, one can track and restore unauthorized modification. Blockchain for access control provides a decentralized mechanism to manage permissions with encryption protection. The sensitive file classification provides priority protection to ensure the critical data receives the highest-level security. Real-time monitoring [19] and real-time intrusion detection are used to identify threats, minimizing potential damage and maintain system integrity. These measures form a comprehensive security system of our file integrity. Some are advancing version control systems with tools from AI (artificial intelligence), machine learning [20], and even deep learning [21] like the MindSpore and PyTorch frameworks, which can be applied further into our FIS, so that unauthorized changes can be predicted and prevented, based on user behavior. All these mechanisms can assist organizations like finance [22] in proactively managing risk and automating recovery processes. Another very plausible direction is the refinement of blockchain technologies for access control, which can greatly improve their scalability [23] and efficiency [24], especially in huge systems [25]. FIS is designed to be able to process advanced predictive techniques in the future. Using models like those created with MindSpore or PyTorch would allow the system to notice behavioural anomalies to spot unexpected large edits at 2 AM from an unusual IP address, as this would not be something a typical user would do at that time. Analysing past action on documents, these tools can guess where documents are likely to be compromised, then prevent access to the file or make users confirm a secondary action before editing.

# References

1. I.H. Montano, I. De La Torre Díez, J.J.G. Aranda, J.R. Diaz, S.M. Cardín, and J.J.G. López, “Secure File Systems for the Development of a Data Leak Protection (DLP) Tool Against Internal Threats,” in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, (IEEE, Madrid, Spain, 2022), pp. 1–7. https://doi.org/10.23919/CISTI54924.2022.9820170
2. X. Xu, W. Wang, J. Lin, Z. Yang, H. Fan, and Q. Wang, “LiTIV: A Lightweight Traceable Data Integrity Verification Scheme for Version Control Systems,” in *2022 International Conference on Computer Communications and Networks (ICCCN)*, (IEEE, Honolulu, HI, USA, 2022), pp. 1–10. https://doi.org/10.1109/ICCCN54977.2022.9868883
3. V. Singh, A. Singh, A. Aggarwal, and S. Aggarwal, “DevOps based migration aspects from Legacy Version Control System to Advanced Distributed VCS for deploying Micro-services,” in *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, (IEEE, Bangalore, India, 2021), pp. 1–5. https://doi.org/10.1109/CSITSS54238.2021.9683718
4. S.-B. Ho, N.A. Ahmad, I. Chai, C.-H. Tan, and S.-L. Chean, “Modeling Reliable Intelligent Blockchain with Architectural Pattern,” in *Emerging Trends in Intelligent Computing and Informatics*, edited by F. Saeed, F. Mohammed, and N. Gazem, (Springer International Publishing, Cham, 2020), pp. 1122–1131. https://doi.org/10.1007/978-3-030-33582-3\_105
5. J.S. Baladhay, H.V. Gamido, and E.M. De Los Reyes, “Large file encryption in a reduced-round permutation-based AES file management system,” IJEECS **34**(3), 2021 (2024). https://doi.org/10.11591/ijeecs.v34.i3.pp2021-2031
6. N.C. Nelakuditi, N.K. Namburi, J. Sayyad, D.V. Rudraraju, R. Govindan, and P.V. Rao, “Secure File Operations: Using Advanced Encryption Standard for Strong Data Protection,” IJSSE **14**(03), 1007–1014 (2024). https://doi.org/10.18280/ijsse.140330
7. B. Al-Muntaser, M.A. Mohamed, and A.Y. Tuama, “Real-Time Intrusion Detection of Insider Threats in Industrial Control System Workstations Through File Integrity Monitoring,” IJACSA **14**(6), (2023). https://doi.org/10.14569/IJACSA.2023.0140636
8. H. Bai, Y. Hu, Q. Liu, J. Zhang, L. Xu, and H. Lin, “Ransomware detection on windows systems using file system activity monitoring and a hybrid xgboost-isolation forest approach,” (2024). https://doi.org/10.22541/au.172893916.67101182/v1
9. K. Akuthota, A. Ganesh, B.R. A, and S. Depuru, “Machine Learning Models for Classification of Sensitive Financial Documents,” in *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, (IEEE, Hamburg, Germany, 2023), pp. 334–340. https://doi.org/10.1109/ICCCMLA58983.2023.10346685
10. A. Salman, M.S. Khan, S. Idrees, F. Akram, M. Junaid, and A.L. Malik, “File Integrity Checkers: Functionality, Attacks, and Protection,” in *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, (IEEE, Rawalpindi, Pakistan, 2022), pp. 1–6. https://doi.org/10.1109/ICoDT255437.2022.9787428
11. S. Das, R. Priyadarshini, M. Mishra, and R.K. Barik, “Leveraging Towards Access Control, Identity Management, and Data Integrity Verification Mechanisms in Blockchain-Assisted Cloud Environments: A Comparative Study,” JCP **4**(4), 1018–1043 (2024). https://doi.org/10.3390/jcp4040047
12. S.B. Ho, S.L. Chean, I. Chai, and C.H. Tan, “Engineering Meaningful Computing Education: Programming Learning Experience Model,” in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, (IEEE, Macao, Macao, 2019), pp. 925–929. https://doi.org/10.1109/IEEM44572.2019.8978920
13. X.-X. Cho, S.-B. Ho, and C.-H. Tan, “Improving asthma treatment adherence by integrating weather information with responsive web technique,” (Virtual Conference, 2024), p. 030005. https://doi.org/10.1063/5.0216655
14. R. Kishimoto, T. Kanda, Y. Manabe, K. Inoue, and Y. Higo, “Osmy: A Tool for Periodic Software Vulnerability Assessment and File Integrity Verification using SPDX Documents,” in *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, (IEEE, Rovaniemi, Finland, 2024), pp. 445–449. https://doi.org/10.1109/SANER60148.2024.00052
15. R. Haque, S.-B. Ho, I. Chai, C.-W. Teoh, A. Abdullah, C.-H. Tan, and K.S. Dollmat, “Intelligent Health Informatics with Personalisation in Weather-Based Healthcare Using Machine Learning,” in *Innovative Systems for Intelligent Health Informatics*, edited by F. Saeed, F. Mohammed, and A. Al-Nahari, (Springer International Publishing, Cham, 2021), pp. 29–40. https://doi.org/10.1007/978-3-030-70713-2\_4
16. W.-X. Ong, S.-B. Ho, and C.-H. Tan, “Enhancing Migraine Management System through Weather Forecasting for a Better Daily Life,” JIWE **2**(2), 201–217 (2023). https://doi.org/10.33093/jiwe.2023.2.2.15
17. I. Ibriwesh, S.-B. Ho, I. Chai, and C.-H. Tan, “A Controlled Experiment on Comparison of Data Perspectives for Software Requirements Documentation,” Arab J Sci Eng **42**(8), 3175–3189 (2017). https://doi.org/10.1007/s13369-017-2425-2
18. S.B. Ho, I. Chai, and C.H. Tan, “AN EMPIRICAL INVESTIGATION OF METHODS, FOR TEACHING DESIGN PATTERNS WITHIN, OBJECT-ORIENTED FRAMEWORKS,” Int. J. Info. Tech. Dec. Mak. **06**(04), 701–722 (2007). https://doi.org/10.1142/S021962200700271X
19. J.-L. Goh, S.-B. Ho, and C.-H. Tan, “Weather-Based Arthritis Tracking: A Mobile Mechanism for Preventive Strategies,” Journal of Informatics and Web Engineering  **3**(1), 210–225 (2024). https://doi.org/10.33093/jiwe.2024.3.1.14
20. R. Haque, S.-B. Ho, I. Chai, and A. Abdullah, “Parameter and Hyperparameter Optimisation of Deep Neural Network Model for Personalised Predictions of Asthma,” JAIT **13**(5), (2022). https://doi.org/10.12720/jait.13.5.512-517
21. M.T.-T. Yong, S.-B. Ho, and C.-H. Tan, “Migraine Generative Artificial Intelligence based on Mobile Personalized Healthcare,” Journal of Informatics and Web Engineering **4**(1), 275–291 (2025). https://doi.org/10.33093/jiwe.2025.4.1.20
22. J.-X. Kok, S.-B. Ho, and C.-H. Tan, “Enhancing Financial Literacy: A Progressive Web Application Approach for Malaysian Youth,” Journal of Informatics and Web Engineering **4**(1), 53–69 (2025). https://doi.org/10.33093/jiwe.2025.4.1.5
23. Ho, S.-B., Chai, I., & Tan, C.-H. (2013). Leveraging Framework Documentation Solutions for Intermediate Users in Knowledge Acquisition. *International Journal of Information Science*, *3*(1), 13-23.,” (n.d.). https://www.sciepub.com/reference/159549
24. “Improved Adam-based Feedforward Deep Neural Network Model for Personalized Asthma Predictions,” JSMS **13**(2), (2023). https://doi.org/10.33168/JSMS.2023.0217
25. C.-W. Teoh, S.-B. Ho, K.S. Dollmat, and I. Chai, “An Evolutionary Algorithm-Based Optimization Ensemble Learning Model for Predicting Academic Performance,” in *2022 11th International Conference on Software and Computer Applications*, (ACM, Melaka Malaysia, 2022), pp. 102–107. https://doi.org/10.1145/3524304.3524320