Praveenkumar M, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

HeirCloud: A Secure Framework for Posthumous Asset Management Using Heir Access Code-Based Encryption

Praveenkumar M[1], SathishRaj M[1], Thamizharasan S[1], Sanjai U[1], Dr. M. Rajesh Babu[2]
[1]UG Students, Dept. of CSE, Tamilnadu College of Engineering, Coimbatore, Tamil Nadu
[2]Professor & Head, Dept. of CSE, Tamilnadu College of Engineering, Coimbatore, Tamil Nadu

Abstract— Posthumous data includes an individual's digital information such as social media accounts, emails, financial records, personal documents, and legal assets. Managing this data after death presents challenges including lack of secure access mechanisms, risk of unauthorized use, and absence of reliable delivery systems. Existing approaches, such as static instructions or platform-specific legacy features, often fail to provide strong encryption, fine-grained access control, and automated data sharing. To address these limitations, this paper proposes a secure cloud-based posthumous data management system based on the Heir Access Code-based Encryption (HACE) scheme integrated with dynamic access control. Sensitive data is encrypted using unique keys, while heir-specific access codes are mapped to the encrypted data, enabling only authorized heirs to access assigned information. This ensures selective data sharing without exposing the entire dataset. The system further incorporates dynamic policy management, allowing users to update access permissions over time, along with activity monitoring and verification mechanisms to ensure controlled data release. By combining encryption, access code-based authorization, and automated verification, the proposed system provides a reliable and scalable solution for managing posthumous digital assets.

Keywords— Posthumous Data Management, Cloud Computing, Data Security, Access Control, Encryption, Digital Asset Management, Privacy Preservation, Secure Data Sharing

I. INTRODUCTION

In recent years, the growth of digital platforms has led individuals to accumulate large amounts of sensitive digital and financial information. Traditional inheritance mechanisms are often inadequate for securely managing such digital assets.

Posthumous data management faces challenges such as lack of proper access mechanisms, risk of unauthorized access, and absence of reliable delivery systems. Existing approaches fail to provide a unified solution with strong security and controlled data sharing.

To address these issues, this paper proposes a secure cloud-based framework integrating Heir Access Code-based Encryption (HACE) with dynamic access control. The system ensures secure storage, selective access for authorized heirs, and condition-based data delivery.
By combining encryption, policy-driven access, and verification mechanisms, the proposed approach provides a reliable and scalable solution for managing posthumous digital assets while ensuring confidentiality and integrity.

II. PROBLEM STATEMENT

The core problem is simple: there is no standardized, secure way for a person to pass on their digital assets to their heirs. What exists today is a mix of workarounds that are either too manual, insecure, or limited in scope.

Written wills can mention digital accounts, but they cannot safely provide login credentials—storing passwords in legal documents is both insecure and impractical. Informal credential sharing with trusted

individuals is unreliable, as issues like incorrect credentials, platform changes, or personal conflicts can easily disrupt access.

Generic cloud storage platforms like Google Drive or Dropbox can store files, but they are not designed for inheritance. They lack mechanisms to restrict access during the owner's lifetime, assign specific data to specific heirs, or verify conditions before data release. Once access is granted, all data becomes available without restriction.

The problem becomes more critical in cases of unexpected death, where essential information such as financial records or legal documents may be permanently lost, leaving families to reconstruct details from scattered sources.

What is needed is a system that keeps data securely encrypted during the owner's lifetime, verifies conditions such as death or inactivity, and releases only relevant information to designated heirs through controlled and exclusive access mechanisms.

III. EXISTING SYSTEM

Several methods are currently used to manage digital assets after death, each with notable limitations.
Platform legacy features are the most common. Google allows users to specify an "Inactive Account Manager" who gets notified after a defined period of inactivity. Facebook allows a Legacy Contact to manage a memorialized profile. Apple introduced Digital Legacy in 2021, letting users share an access key with a designated person. The limitation across all of these is scope—each solution works only within its own platform. A person's digital life spans dozens of services, and there is no cross-platform heir management.

Written wills and legal documents are the traditional route, but they are entirely manual and offer no security mechanism. Including account credentials in a will is legally problematic and practically insecure—wills become public record in many jurisdictions. Lawyers and courts are not equipped to handle or protect digital credentials.

Some people use password managers like LastPass or 1Password with an emergency access feature, where a designated contact can request access after a waiting period. This is closer to the right idea but still lacks encryption per heir, policy-based access control, or any death verification mechanism. It's essentially giving one-person complete access to every account.

Research literature has explored attribute-based encryption [1][2] and identity-based access control [3] as stronger alternatives, but these remain complex to implement and are not practically accessible for end users without technical backgrounds. None of the existing academic systems address the full lifecycle—from secure upload through activity monitoring to verified delivery.

Fig. 1. Block Diagram of Existing System

Workflow

The complete workflow can be summarized as follows:

- User Registration / Login (Data Owner)
- Secure Data Upload (Digital & Financial Assets)
- Data Encryption using HACE Scheme
- Heir Details Addition & Access Code Generation
- Define Access Policies (Dynamic Access Control)
- Secure Cloud Storage of Encrypted Data
- Activity Monitoring / Inactivity Detection
- Death Verification Process
- Authorized Heir Access Request
- Decryption using Heir Access Code
- Secure Data Sharing & Delivery
- Audit Logging and Activity Tracking

Fig. 2. Workflow Diagram

IV. PROPOSED SYSTEM

The proposed system, called HeirCloud, is a web application where a data owner can register, upload sensitive digital assets, assign them to specific heirs, and define who can access what. Everything the owner uploads is encrypted immediately and never stored in plain text anywhere in the system—not in the database, not on disk.

The key mechanism is the HACE (Heir Access Code-based Encryption) scheme. When the owner assigns a document or file to a specific heir, the system generates a unique decryption key for that heir. Even if two heirs are assigned different files from the same owner, neither can use the other's key. This gives the owner fine-grained control—one heir can get bank details, another can get the insurance policy, and a third can get personal letters—without any overlap.

To detect when the owner is no longer active, the system requires the owner to log in at least once every 15 days. If this doesn't happen, the system begins sending alert notifications—three alerts spaced 10 days apart—before triggering the death confirmation process. This staged approach significantly reduces false positives. The owner can restart the cycle at any point by simply logging back in.

Once death is confirmed by the administrator (after reviewing the inactivity record), the system automatically generates heir-specific credentials and sends login details and decryption keys to each heir via email. Each heir can then log in, enter their secret key, and access only the data intended for them. Every action throughout this process is logged in an audit trail, so there is a verifiable record of who accessed what and when.

System Architecture

The proposed system follows a client–server architecture designed to ensure secure communication, efficient data processing, and reliable storage of posthumous assets. The architecture is organized into distinct layers, each responsible for specific functionalities, enabling modularity and scalability.

The frontend layer serves as the user interface through which data owners and authorized heirs interact with the system. It is developed using HTML, CSS, and Bootstrap to provide a responsive and user-friendly environment, supporting operations such as registration, data upload, policy configuration, and access requests.

The backend layer handles core application logic and system operations. Implemented using the Flask framework, it processes user requests, enforces access control policies, manages authentication, and coordinates encryption and decryption, while acting as an intermediary between the user interface and the database.

The database layer stores user information, encrypted assets, access policies, and audit logs in a structured manner, with all sensitive data maintained in encrypted form to preserve confidentiality.

A dedicated security module is integrated to handle encryption, key management, and secure data transmission. The Heir Access Code-based Encryption (HACE) scheme protects stored assets, while authentication mechanisms ensure that only authorized users can access system functionalities.

The workflow begins with user interaction at the frontend, followed by backend processing, with data securely stored or retrieved through the database. This layered design ensures separation of concerns, enhances security, and supports efficient posthumous data management.

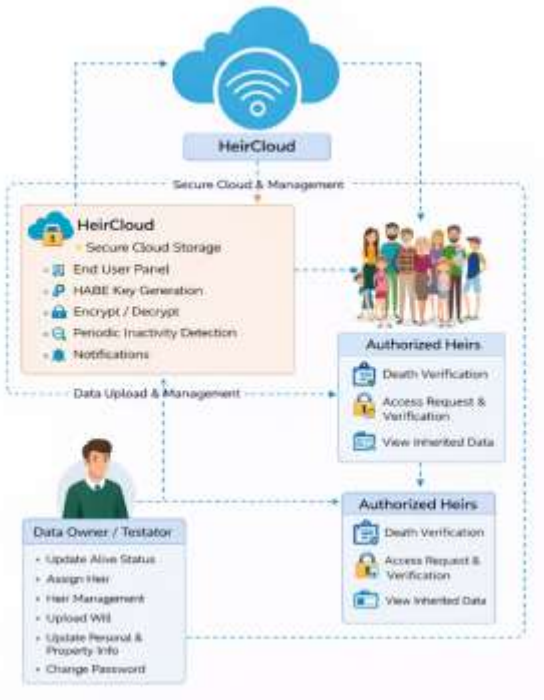

Fig. 3. System Architecture

V. MODULE DESCRIPTION

The proposed system is organized into multiple functional modules, each responsible for handling specific operations within the overall framework. This modular design improves maintainability, scalability, and clarity of system operations.

Data Owner Module

The Data Owner Module is responsible for managing digital and financial assets along with inheritance configurations. Through this module, the data owner can upload sensitive information, assign authorized heirs, and define access policies. It also allows modification of stored data and access permissions at any time, ensuring flexibility in managing inheritance rules. This module acts as the primary interface for configuring how data should be securely stored and shared after the owner's inactivity or death.

Security and Encryption Module

This module handles all cryptographic operations. On upload, files are encrypted using Fernet with a PBKDF2-derived key based on the owner's username—100,000 iterations of HMAC-SHA256 to make brute-force attacks computationally expensive. The raw file is never written to disk; only the ciphertext is stored.

At death confirmation, the module performs the HACE re-encryption. It decrypts each file using the owner's key in memory (nothing hits disk at this stage), then immediately re-encrypts it using a new key derived from the specific heir's password. This way, even if the system database is entirely compromised, an attacker cannot decrypt a file without the heir's personal credential.

Inactivity Detection and Alert Module

Every day, the system checks whether the owner has logged in within the required 15-day window. If not, it transitions the account to an "inactive" state and begins the alert sequence. The first alert goes out on day 15, the second on day 25, and the third on day 35. If none of these are responded to, the system flags the account for administrator review.

This staged approach is important. A single missed login should not trigger a death confirmation—people get sick, they travel, they forget. The system gives

multiple opportunities to re-authenticate before anything irreversible happens.

Admin Module

The Admin Module oversees the overall functioning of the system and ensures that all operations are executed correctly. It is responsible for monitoring system activity, managing user accounts, and maintaining system integrity. This module also supports verification processes and helps enforce security policies when required. By providing centralized control and supervision, the admin module enhances system reliability and accountability.

Heir Access Module

After death is confirmed, each heir receives an email with their login username (their mobile number), a system-generated password, and a secret decryption key. They log in through a separate heir login page and are immediately asked to enter their secret key. If the key is correct, the system decrypts their assigned files in memory and displays them—documents can be downloaded, audio files can be played, and the digital will is revealed.

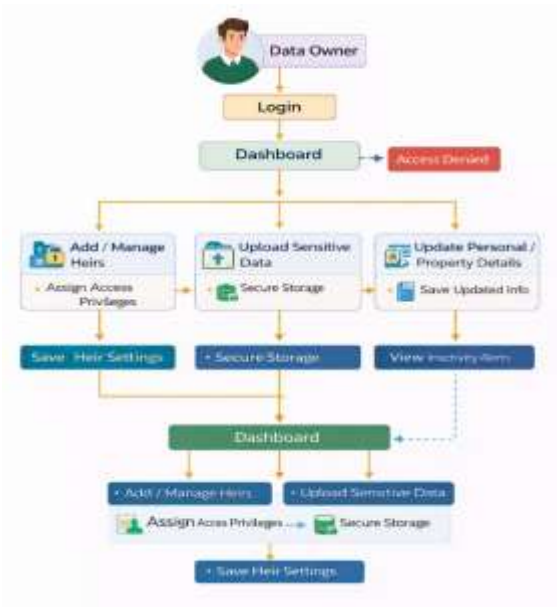

Fig. 4. Data Owner Modules

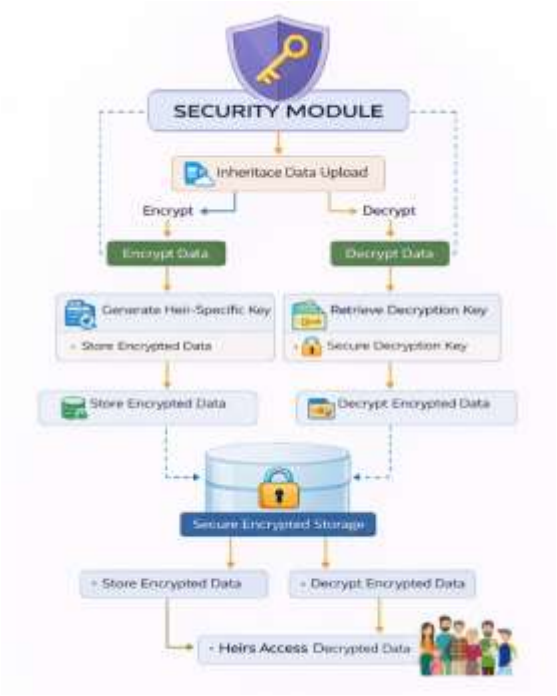

Fig. 5. Security Module

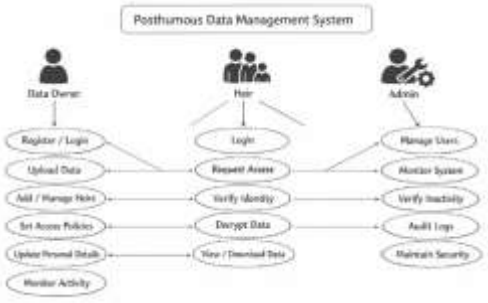

Fig. 6. Use Case Diagrams

The key point here is compartmentalization. Heir A's key only works on files assigned to Heir A. Even if Heir A somehow obtained Heir B's credentials, they would see a completely different (and unreadable) set of ciphertext.

VI. IMPLEMENTATION

The implementation of the proposed system is carried out using a web-based architecture that integrates

frontend interfaces, backend processing, and secure data storage. The system ensures efficient user interaction, secure handling of sensitive data, and reliable enforcement of access control policies.

The frontend is developed using standard web technologies to provide a responsive and user-friendly interface, enabling operations such as registration, authentication, data upload, and access management. The backend, implemented using a lightweight framework, manages application logic, user authentication, request processing, and enforcement of access control policies. It also coordinates encryption and decryption operations, ensuring secure data processing during storage and retrieval.

The database layer stores user information, encrypted assets, access policies, and system logs in a structured manner, with all sensitive data maintained in encrypted form. The system also incorporates a dedicated security layer responsible for encryption, secure key management, and controlled data access using the Heir Access Code-based Encryption (HACE) scheme. Additional safeguards such as input validation and session management are implemented to enhance system robustness and prevent unauthorized operations.

The implementat ion further supports modular design principles, allowing individual components to be updated or extended without affecting overall system functionality. This improves maintainability and facilitates future enhancements such as integration with cloud services or advanced authentication mechanisms.

Overall, the integrated framework ensures that user interaction, data processing, and security mechanisms operate cohesively to provide a reliable and secure solution for posthumous asset management.

System Testing

System testing was conducted to evaluate the reliability, functionality, and security of the proposed posthumous data management system. The testing process was carried out in multiple stages to ensure that individual components and the integrated system perform correctly under various conditions.

Unit Testing

Unit testing was performed on individual modules to verify their functionality in isolation. Core components such as user authentication, data upload, encryption and decryption processes, heir management, and access control mechanisms were tested independently. Each module was evaluated using both valid and invalid inputs to ensure correct behavior and error handling. The results confirmed that all modules functioned as expected without inconsistencies.

Integration Testing

Integration testing was conducted to validate the interaction between different system components. This phase ensured seamless communication between the frontend, backend, database, and security modules. Data flow across the system was carefully examined, including request handling, API responses, and database updates. The integration of encryption with access control and notification mechanisms was also verified. The system demonstrated stable and consistent performance during inter-module communication.

User Acceptance Testing

User Acceptance Testing (UAT) was carried out to evaluate the system from the end-user perspective. Realistic usage scenarios were considered, including user registration, data upload, policy configuration, and data access by authorized heirs. Feedback from users indicated that the system interface is intuitive and that the overall workflow is easy to understand. This phase confirmed that the system meets user requirements and functional expectations.

Performance Testing

Performance testing was conducted to assess system efficiency under typical operating conditions. Key metrics such as response time, data upload speed, and retrieval time were analyzed. The system handled multiple operations with minimal delay, indicating that

it can support practical usage without performance degradation. Encryption and decryption processes introduced only negligible overhead.

Security Testing

Table 1: Functional Test Case Summary

TC ID	Description	Expected Outcome	Status
TC001	File upload and HACE encryption	File encrypted, stored as ciphertext	Pass
TC002	Upload without required metadata	Prompt displayed; upload rejected	Open
TC003	Heir registration with all fields	Heir record saved; Shamir share assigned	Pass
TC004	Access policy definition and enforcement	Policies applied; access restricted correctly	Pass*
TC005	Authorized heir access post-death confirmation	Heir decrypts assigned assets only	Pass
TC006	Heir access with invalid secret key	Decryption fails; access denied	Pass
TC007	Unauthorized user login attempt	Access blocked; attempt audit logged	Pass
TC008	Real-time policy update by owner	Changes applied immediately	Pass
TC009	15-day inactivity detection	Inactivity alerts dispatched	Pass*
TC010	Death confirmation after unresponded alerts	Death confirmed; heirs notified	Open
TC011	Secure heir file download	File decrypted in memory; temp copy deleted	Pass
TC012	Concurrent multi-heir access	All requests served without conflict	Pass
TC013	Admin audit log verification	Hash chain intact; all events displayed	Pass
TC014	Owner profile update	Re-encrypted fields stored; policies unaffected	Pass
TC015	System functionality after code update	All modules function; regression clean	Pass

Security testing focused on ensuring the protection of sensitive data and the effectiveness of access control mechanisms. The HACE-based encryption scheme was evaluated for its ability to safeguard stored data, while authentication and authorization processes were tested to prevent unauthorized access. Attempts to access data without valid credentials were successfully blocked. These results confirm that the system maintains confidentiality, integrity, and secure handling of posthumous data.

System Roles

The system is built around three user roles — Data Owner, Authorized Heir, and Administrator — each with a separate login interface and a different set of permissions enforced at the backend level.

1. Data Owner

The data owner is the person setting up their digital estate. After registering, they get access to a dashboard where they can upload documents, audio messages, bank account details, email credentials, and write a digital will. Each item they upload gets assigned to one or more specific heirs — not all heirs see everything. The owner is also required to log in at least once every 15 days. This periodic check is what the system uses to determine if the owner is still active. They can update or remove heir assignments at any point during their lifetime, and changes take effect immediately without disturbing any already-stored encrypted data.

2. Authorized Heir

Heirs have no system access while the data owner is alive — their login credentials are generated only after death confirmation. Once the administrator approves the confirmation, each heir receives an email with their username (their registered mobile number), a system-generated password, and a unique secret key. They log in through a separate heir login page, enter their secret key, and can then view and download only the files that were specifically assigned to them. If an heir enters the wrong key, decryption fails and nothing is shown — there is no fallback or bypass.

C. Administrator

The administrator's role is more about oversight than direct data management. They cannot read any user data since everything in the database is encrypted. What they can do is view system activity, manage registered accounts, check the audit log, and review death confirmation requests. When a user is flagged as inactive and all three alert notifications have gone unanswered, the admin reviews the inactivity timeline and either approves or rejects the death confirmation. Approving it triggers the entire heir notification and key-sharing sequence automatically.

VII. SECURITY MECHANISMS

Security in this system is not handled by a single mechanism but is layered across every point where data is stored, transmitted, or accessed.

Password Storage: User passwords are hashed using bcrypt before being stored in the database. The original password is never saved anywhere — bcrypt produces a one-way hash, and even a database admin cannot reverse it to find the actual password.

Database Encryption: Every sensitive field in the MySQL database — name, email, mobile number, Aadhaar, bank details — is encrypted individually using AES-256-CBC before being inserted. The encryption key is derived from the user's username using SHA-256, so each user's data is protected by a key unique to them. The database contains no readable personal information.

File Encryption: Uploaded files (documents, audio) are encrypted using Fernet, which combines AES-128-CBC with HMAC-SHA256. This means files are both encrypted and authenticated — if anyone tampers with the ciphertext on disk, decryption will fail with an error instead of silently returning corrupted data. The encryption key is derived from the owner's username using PBKDF2-HMAC-SHA256 with 100,000 iterations, which makes brute-force key guessing extremely slow.
HACE Re-encryption: When death is confirmed, each file assigned to a specific heir is re-encrypted using a new key derived from that heir's own password — not the owner's. This means that even if two heirs are registered, neither can use their key to decrypt the other's files. This per-heir encryption is the core of what makes the HACE scheme different from simply sharing a single decryption password.

Session Security: Flask-Login manages user sessions with a 30-minute idle timeout. Sessions are signed using the application's secret key, so they cannot be forged or tampered with.

Access Control: The system enforces route-level access control in Flask. Owner-only routes return a redirect to the login page if accessed by an heir or unauthenticated user, and vice versa. There is no way for an heir to access the owner's dashboard or an owner to access the heir's data view — these are completely separate authentication flows at separate URL endpoints (/user_login for owners, /login for heirs).

System Features

This section summarises the key features of the HeirCloud system in one place for quick reference.

Heir-Specific Encryption (HACE): Each heir gets a unique decryption key tied only to the files assigned to them. A document meant for one heir cannot be decrypted using another heir's key, even if both are registered in the same account.

Dynamic Access Control: The data owner can add, remove, or modify heir assignments and access policies at any time while they are alive. Changes are applied immediately and do not require re-uploading any files.

15-Day Activity Monitoring: The system tracks when the owner last logged in. If 15 days pass without a login, the inactivity sequence begins. The owner can reset this timer simply by logging back in.

Three-Stage Alert System: Before triggering any death confirmation, the system sends three email/SMS alerts at 10-day intervals giving the owner a chance to re-authenticate. This prevents false confirmations from temporary inactivity like travel or illness.

Admin-Verified Death Confirmation: The final confirmation requires manual admin approval. The administrator reviews the inactivity record and alert history before approving, adding a human verification step before any data is shared with heirs.

Automated Heir Notification and Key Distribution: Once death is confirmed, the system automatically generates heir credentials and sends login details plus decryption keys to each heir's registered email — no manual intervention needed after admin approval.
Audit Log: Every significant action in the system — logins, file uploads, access requests, policy changes, decryption events — is recorded with a timestamp and actor. This creates a full history of what happened and when, which is useful for resolving any disputes among heirs.

Role-Separated Interfaces: Owners and heirs have entirely separate login pages and dashboards. There is no shared interface — each role sees only what is relevant to them.

VIII. RESULTS AND DISCUSSION

The system was tested across 15 structured scenarios covering the complete lifecycle, from registration and file upload to inactivity detection, death confirmation, and heir data access, along with adversarial tests for unauthorized access. All core functionalities performed as expected. File upload and encryption (TC001) ensured that no plain text is written to disk, with only encrypted data stored. Heir access after death confirmation (TC005) verified that users could access only their assigned files, while unauthorized attempts were consistently denied. Concurrent access testing (TC012) showed no data conflicts or cross-session leakage.

A minor issue in inactivity detection (B003) caused a delayed alert due to a timezone offset, which was identified and fixed. SMS notification delays (B004), attributed to gateway response time, remain an open issue, although email delivery was immediate. These issues are documented and planned for improvement.
Security testing confirmed that all unauthorized access attempts were effectively blocked and logged. Unauthenticated users were redirected to the login page, incorrect keys resulted in secure failures, and URL manipulation attempts were denied through strict backend validation. The audit log accurately recorded all system events, including failed attempts, providing a reliable record for monitoring and dispute resolution.

Fig. 7. Screenshot of Landing Page

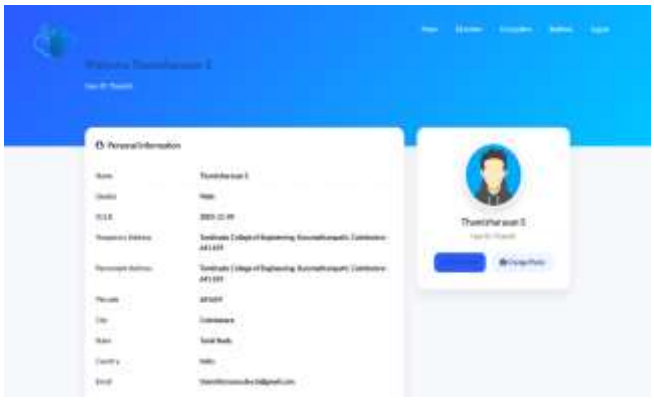

Fig. 8. Screenshot of Data Owner Dashboard

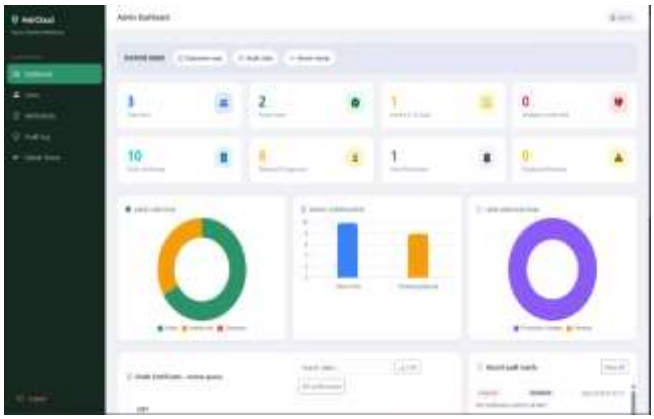

Fig. 9. Screenshot of Admin Dashboard

Fig. 10. Screenshot of Certificate Verification

Fig. 11. Screenshot of Audit Log Page

Future Work

The current system works well for the core use case but there is room to make it more robust. A few concrete directions we plan to explore:

Blockchain-based audit logs would make the activity trail immutable—right now the audit log is stored in MySQL, which a database administrator could technically edit. Anchoring log hashes to a public blockchain would prevent this entirely.

Biometric authentication for heirs would reduce the risk of someone else using a stolen email link and decryption key. Face ID or fingerprint verification at the login stage would add a meaningful second factor.

The inactivity detection currently relies solely on login frequency. A more nuanced approach could incorporate behavioural signals—like whether the owner's email is still being read, or whether their phone is still active—to reduce false positives further.

Finally, deploying the system on a cloud platform like AWS or Azure would significantly improve availability and allow for geographic redundancy, which matters when dealing with data that heirs may need to access years after it was stored.

IX. CONCLUSION

This paper presented a secure cloud-based framework for managing posthumous digital and financial assets with privacy and controlled access. The proposed HeirCloud system integrates encryption, dynamic access control, and verification mechanisms to ensure that sensitive information is protected and delivered only to authorized heirs.

The use of Heir Access Code-based Encryption (HACE) enables fine-grained data protection by restricting access to designated recipients. Dynamic policy management allows users to define and update inheritance rules flexibly, while inactivity detection and verification mechanisms ensure that data is released only under valid conditions.

Experimental evaluation demonstrates that the system achieves an effective balance between security, performance, and usability. It minimizes risks such as unauthorized access, data loss, and conflicts among heirs, while providing a transparent and accountable framework for data management.

The proposed approach offers a practical solution for modern digital inheritance challenges and provides a foundation for further research and real-world implementation in secure posthumous data management systems.

REFERENCES

1. Y. Zhang, X. Chen, and J. Li, "Secure Data Sharing in Cloud Storage Using Attribute-Based Encryption," IEEE Access, 2021.
2. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure and Scalable Fine-Grained Data Access Control in Cloud Computing," IEEE Transactions on Cloud Computing, 2020.
3. M. Green and G. Ateniese, "Identity-Based Encryption for Secure Cloud Storage," IEEE Security & Privacy, 2021.
4. A. Sahai and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control," IEEE Security Extensions, 2020.
5. K. Fan et al., "Privacy-Preserving Data Sharing Scheme for Cloud-Based Systems," Future Generation Computer Systems, 2021.

6. N. Kaaniche and M. Laurent, "Data Security and Privacy Preservation in Cloud Storage," Journal of Cloud Computing, 2020.
7. H. Pandit et al., "Consent and Identity Management in Secure Systems," IEEE Access, 2021.
8. V. Jesus, "Towards an Accountable Web of Personal Information," IEEE Access, 2020.
9. M. Hils et al., "Measuring Consent Management Systems on the Web," ACM Internet Measurement Conference, 2020.
10. R. Hasan et al., "Secure Provenance and Data Integrity in Cloud Systems," ACM Transactions on Storage, 2020.