A

Project Synopsis

on

The Final Transfer: Protecting your legacy beyond the screen

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2023-27

in

Computer Science And Engineering

By

Harsh Chaudhary (2300290100117) (Team Leader)

Dev Kumar(2300290100097)

Deepanshu Singh(2300290100096)

Abhinav Kumar(2300290100009)

Under the supervision of

Ms. Surbhi Jain

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

Academic Year 2025-26

Abstract

The rapid growth of digital ownership has created a critical issue: when a user passes away, their digital assets—such as cryptocurrency keys, financial passwords, and sentimental photos—often become permanently inaccessible to their grieving families. This phenomenon, known as "Digital Asset Intestacy," leads to significant financial loss and emotional distress.

The objective of this project, **"The Final Transfer,"** is to prevent this data loss by developing a secure, automated legacy management system that functions as a digital "Dead Man's Switch." The proposed solution enables users to store sensitive information in encrypted "Life Vaults" using a **Zero-Knowledge architecture**, ensuring absolute privacy where even the administrators cannot access the data.

The system actively monitors the user's status via a **"Heartbeat Protocol."** If the user becomes unresponsive and a trusted human delegate confirms the situation through the **"Lazarus Verification"** process, the system automatically decrypts and distributes specific asset packets to designated beneficiaries. By automating the inheritance process, this project ensures that digital legacies are preserved and securely transferred without the need for complex legal intervention or insecure password sharing, effectively bridging the gap between lifetime security and post-mortem accessibility.

Introduction

Background & Problem Statement

In the modern digital era, a significant portion of individual wealth and identity resides in cloud-based accounts, cryptocurrency wallets, and encrypted storage. However, a critical vulnerability exists in this ecosystem: **"Digital Asset Intestacy."** When a user passes away or becomes incapacitated, there is no standardized, secure mechanism to transfer their digital credentials to their next of kin.

Traditional estate planning tools (physical wills) are ill-equipped to handle dynamic digital assets like changing passwords or 2FA keys. Conversely, existing technical solutions like password managers create a "Single Point of Failure"—if the master password dies with the user, the data is lost forever. This creates a scenario where billions of dollars in assets and irreplaceable sentimental memories are permanently locked away from grieving families every year.

Gap Analysis

Current solutions fail to bridge the gap between security and accessibility.

Google's Inactive Account Manager is restricted to its own ecosystem, while **standard password managers** typically lack automated "push" protocols, relying instead on beneficiaries to initiate requests. There is a clear need for a platform-agnostic, automated system that prioritizes user privacy while ensuring post-mortem accessibility.

Proposed Solution & Detailed Features "The Final Transfer" is a cryptographically secure web application designed to function as an intelligent "Dead Man's Switch." It automates the transition of digital ownership based on verifiable "Proof of Life" conditions. The system is built upon five core functional pillars:

- **A. Zero-Knowledge "Life Vaults" (Client-Side Encryption):** Security is the foundation of the project. We utilize **Client-Side Encryption (AES-256)**, meaning data is encrypted in the user's browser *before* it reaches the server. The system administrators (us) store only the encrypted "blob" and never possess the decryption keys, ensuring that user data remains private even in the event of a server breach.
- **B. The Heartbeat Protocol (Active Monitoring):** The system employs an automated scheduler to monitor the user's activity. Users must "Check-In" via email or SMS at defined intervals (e.g., every 30 days). Failure to respond activates a "Grace Period" with intensified notifications. If silence persists, the system triggers the "Dormant State."
- **C. The "Lazarus" Verification (Consensus Mechanism):** To prevent false positives (e.g., the user is simply offline due to travel), the system does not release data immediately upon timer expiry. Instead, it contacts pre-designated **"Verifiers"** (e.g., a lawyer or spouse). The system requires human confirmation—and optionally, a consensus (e.g., 2 out of 3 verifiers)—before the decryption process is authorized.
- **D. Granular Packet Distribution:** Recognizing that not all beneficiaries should receive all data, the system allows for asset segmentation. Users create specific "Packets"—**Packet A** (Financial credentials) may be routed to a spouse, while **Packet B** (IP and server keys) is routed to a business partner. This ensures privacy between beneficiaries.
- **E. Emergency "Rapid Release" Protocol:** For urgent scenarios (e.g., medical directives or funeral funds), an optional "Emergency Packet" is implemented. This bypasses the standard 30-day wait time and can be triggered manually by a specific trusted verifier using a unique "Panic Code," subject to a short (6-hour) user veto window to prevent abuse.

2.4 Objectives of the Project

- To implement a secure, timer-based logic for detecting user inactivity.
- To demonstrate the application of Zero-Knowledge proofs in a consumer web app.
- To provide a seamless, user-friendly dashboard for managing complex cryptographic keys.

2.5 Scope of the Project The project scope includes the development of a **MERN stack** web application, the integration of **Cron jobs** for scheduling, and **Nodemailer** for the notification

ecosystem. The system handles text-based secrets (passwords, seeds) and document storage. It does not include direct integration with banking APIs for transaction execution.

2.6 Technologies Used

- **Frontend:** React.js, Tailwind CSS.
- **Backend:** Node.js, Express.js.
- **Database:** MongoDB (NoSQL).
- **Security:** AES-256 (Encryption), Bcrypt (Hashing), JWT (Auth).
- **Services:** Node-Cron (Scheduler), Nodemailer (Email).

Literature Survey

The domain of Digital Legacy Management (DLM) has seen various attempts at solutioning, ranging from corporate-specific features to complex blockchain protocols. A review of the existing technological landscape reveals three primary categories of "Dead Man's Switch" implementations: **Ecosystem-Specific Tools**, **Commercial Password Managers**, and **Decentralized Smart Contracts**.

Ecosystem-Specific Solutions (Corporate Tools)

The most prominent example in this category is **Google's Inactive Account Manager** (formerly "Google Death" features).

- **Mechanism:** This tool allows users to set a timeout period (e.g., 3, 6, or 12 months) of inactivity. Once the account is flagged as inactive, Google notifies a pre-selected trusted contact and shares data from specific Google services (Gmail, Photos, Drive).
- **Limitations:** The primary drawback is its **"Walled Garden"** nature. It is strictly limited to the Google ecosystem. It cannot transfer assets that exist outside Google's servers, such as online banking credentials, external cryptocurrency wallets, or subscription services (Netflix, AWS). Furthermore, it operates on a "data dump" model rather than a secure, granular credential transfer.

Commercial Password Managers (The "Pull" Model)

Market-leading password managers like **LastPass** and **1Password** have introduced "Emergency Access" features.

- **Mechanism:** These systems typically use a **"Pull Protocol."** The beneficiary (heir) must request access to the vault. The system then emails the original user. If the user does not deny the request within a set waiting period (e.g., 7 days), access is granted.
- **Limitations:**

1. **Initiation Burden:** The process relies on the beneficiary *knowing* that the account exists and *initiating* the request. If the heir is unaware of the digital assets, they remain lost.
2. **Privacy Violation:** Most of these tools function on an "All-or-Nothing" basis. Granting emergency access typically exposes the *entire* vault to the beneficiary. This is problematic for users who may wish to share financial passwords with a spouse but keep personal journals or business intellectual property private.
3. **Physical Vulnerability:** 1Password, for example, relies on a printed "Emergency Kit" (PDF) containing the Secret Key. If this physical paper is lost or stolen, the security model collapses.

Blockchain-Based Inheritance (Smart Contracts)

With the rise of Web3, several decentralized applications (dApps) like **SafeHaven (Inheriti)** and generic Ethereum **Smart Contract Wills** have emerged.

- **Mechanism:** These solutions use blockchain smart contracts to hold tokens or keys, which are released when certain network conditions are met (often verified by "oracles" or decentralized validators).
- **Limitations:**
    1. **Technical Barrier:** These solutions require the user to possess significant technical knowledge (managing gas fees, wallet addresses). This makes them unsuitable for the average non-technical population.
    2. **Cost Volatility:** Executing smart contracts on main-nets (like Ethereum) incurs "Gas Fees," which can be prohibitively expensive during times of network congestion.
    3. **Immutability Risks:** Once a smart contract is deployed, it is difficult to alter. If a user changes their mind about a beneficiary, updating the "Will" often requires deploying a new contract and paying fees again.

Academic Research & Gap Analysis

Research on Digital Legacy Management Systems: Theoretical, Systemic and Users' Perspective (2021) highlights that over **70% of internet users** do not have a digital will due to the complexity of existing tools. Further studies in the *IEEE Access Journal* suggest that while Zero-Knowledge encryption is the gold standard for privacy, it has rarely been successfully integrated with automated legacy triggers in a consumer-friendly web application.

Conclusion of Survey: The existing landscape is fragmented. Corporate tools are too limited in scope; password managers rely on manual initiation by heirs; and blockchain solutions are

too complex for mass adoption. There is a distinct absence of a **Platform-Agnostic, Automated "Push" System** that combines the ease of use of Web2 with the privacy guarantees of Zero-Knowledge encryption. **"The Final Transfer"** is designed specifically to fill this gap by offering a granular, automated, and secure asset transfer protocol.

Methodology/ Planning of work

To ensure the successful completion of "The Final Transfer," we will follow a step-by-step development process. This approach divides the work into five clear phases, ensuring that every feature—from secure data storage to the final transfer of assets—is built correctly and tested thoroughly.

Phase 1: Planning & Requirements (The "What")

Before writing code, we define exactly what the system needs to do.

- **Goal Definition:** Identify the core problem—how to securely pass digital information to a beneficiary after a specific event.
- **Feature Listing:** List essential features (e.g., User Registration, Secure Vault, Beneficiary Nomination, Verification System).
- **Feasibility Study:** Confirm the technology needed (Database, Encryption tools) is available and workable.

Phase 2: System Design (The "How")

In this phase, we create the "blueprints" for the project.

- **Workflow Diagrams:** Draw flowcharts showing how a user uploads data and how the system detects when to transfer it.
- **Database Design:** Structure how user data and files will be stored efficiently.
- **UI/UX Design:** Sketch simple, user-friendly screen layouts so users of all ages can navigate the app easily.

Phase 3: Development (The Building)

This is the coding phase where the design becomes a reality.

- **Frontend Development:** Build the user interface (Login screens, Dashboard, Upload forms) using HTML/CSS/React (or chosen framework).
- **Backend Development:** Write the logic that handles password encryption, file storage, and the "trigger" mechanism for the transfer.
- **Integration:** Connect the frontend visual interface with the backend database.

Phase 4: Testing (The Check)

We rigorously check the system to prevent errors and ensure security.

- **Functional Testing:** Ensure every button and form works as expected.
- **Security Testing: Crucial Step.** Verify that no one can access the "Vault" except the user or the verified beneficiary.
- **User Acceptance:** Test the "Transfer" process to ensure it happens smoothly and accurately.

Phase 5: Finalization & Documentation

The final steps to wrap up the project.

- **Bug Fixing:** Resolve any issues found during testing.
- **Documentation:** Write a simple user manual explaining how to use the platform.
- **Final Review:** Present the completed working project.

Facilities required for proposed work

To support the high-security and scalability demands of **"The Final Transfer,"** a comprehensive technology stack is required. This stack covers Web & App Development, Blockchain, Advanced Security, and modern Cloud Infrastructure.

1. Front-End (Web Interface)

- **React.js:** For building a dynamic and responsive web dashboard where users manage their "Vault."
- **Tailwind CSS:** For rapid, responsive UI design that works on all screen sizes.
- **Redux:** For managing the state of the application across complex user flows.

2. Mobile Application (App Interface)

- **Kotlin :** To develop a secure cross-platform mobile application (Android & iOS) with a single codebase, ensuring "Dead Man's Switch" notifications reach the user anywhere.

3. Back-End & Blockchain

- **Node.js & Express.js:** To build secure RESTful APIs that connect the interfaces to the database.
- **Smart Contracts (Solidity):** Used to create an immutable "Will Agreement" on the blockchain, ensuring the transfer logic is tamper-proof.

4. Database & Decentralized Storage

- **MongoDB:** A NoSQL database to store user profiles and metadata flexibly.

- **IPFS (InterPlanetary File System):** For storing encrypted asset files in a decentralized manner, ensuring no single server failure results in data loss.

5. Infrastructure, DevOps & Cloud (New)

- **Docker:** Used for **Containerization**. It packages the application (frontend, backend, and services) into isolated containers to ensure the app runs consistently on any machine or server.
- **Kubernetes (K8s):** Used for **Orchestration**. It automatically manages, scales, and deploys the Docker containers, ensuring the "Vault" stays online even during high traffic.
- **Cloud Provider (AWS / Google Cloud / Azure):** To host the infrastructure. We will utilize cloud computing resources (like EC2 or GKE) to run the Kubernetes clusters securely in the cloud.

6. Security & Cryptography

- **AES-256 Encryption:** Military-grade encryption standard to secure user files before storage.
- **SHA-256 Hashing:** To maintain data integrity and secure passwords.
- **SSL/TLS Certificates:** Ensuring an encrypted pipeline for all data in transit.

References

Research Papers:

- Digital Inheritance in Web3: A Case Study of Soulbound Tokens and the Social Recovery Pallet within the Polkadot and Kusama Ecosystems
- Digital Inheritance & Privacy
- Research Papers: Blockchain & Smart Contracts
- "The Digital Beyond": An online resource dedicated to digital legacy and estate planning.
- "Google Inactive Account Manager": Google's official tool for post-mortem data handling
- Beyond Life: A Digital Will Solution for Posthumous Data Management (2025)