**Groupe 6 MEMBERS: REG NUMBER**

**UWINGABIYE NADINE 23 RP01384**

**NSENGIYUMVA JEAN BOSCO 23 RP01326**

**NSHIMIYIMANA DIEUDONNE 23 RP00022**

**IGIHOZO FELLECITE 23 RP02056**

**TOPIC: MUTUAL LIFE INSURANCE APPLICATION**

**Initial Requirement Specification (IRS) for Mutual Life Insurance Application**

**1. Introduction**

This document outlines the initial requirements for a mutual life insurance application built on the blockchain. This application aims to provide a secure, transparent, and decentralized platform for managing mutual life insurance policies. This IRS will detail the functional and non-functional requirements, system overview, and key components specific to a mutual life insurance application.

**2. System Overview**

The Mutual Life Insurance application is a smart contract-based system deployed on a blockchain platform. It allows users to:

* Purchase life insurance policies
* Pay premiums periodically
* Claim insurance benefits
* Validate and verify claims through a decentralized mechanism

**3. Functional Requirements**

**User Management:**

* Users can create an account and log in to the application.
* Users can view their existing policies and claims.

**Policy Management:**

* Users can purchase a new life insurance policy.
* Users can view policy details including premium amounts, coverage, and beneficiaries.
* Users can pay premiums using cryptocurrency.
* The system can send reminders for premium payments.

**Claims Management:**

* Users can file a claim.
* The system integrates with an external oracle service for verifying claims (e.g., death certificate verification).
* Claims are reviewed and validated through a decentralized voting mechanism by other policyholders or designated authorities.
* Approved claims trigger automatic payout to the beneficiaries.

**4. Non-Functional Requirements**

**Security:**

* All user data and policy information must be securely stored on the blockchain.
* Access to sensitive functions (e.g., editing policy details, approving claims) should be restricted to authorized users.
* The smart contract code should be thoroughly audited to prevent vulnerabilities.

**Performance:**

* The application should be responsive and provide a smooth user experience.
* Transaction fees for interacting with the smart contract should be kept minimal.

**Scalability:**

* The application should be designed to accommodate an increasing number of users and policies.

**Usability:**

* The application interface should be user-friendly and intuitive.
* Users should be able to easily understand and manage their policies and claims.

**5. Components**

**Smart Contract:**

* The core logic resides in a smart contract deployed on the blockchain. It stores policy data, manages premium payments, and handles claims processing.

**User Interface (UI):**

* A web-based or mobile application provides a user interface for interacting with the smart contract.

**Oracle Service:**

* An external service verifies claims (e.g., death certificate verification) and triggers the smart contract for benefit payouts.

**Voting Mechanism:**

* A decentralized voting mechanism for validating claims, ensuring transparency and fairness in claim approvals.

**6. Data Flow**

1. User creates an account and logs in to the application.
2. User purchases a new life insurance policy via the UI.
3. The UI interacts with the smart contract, storing policy data (owner, premiums, beneficiaries) on the blockchain.
4. User pays premiums through the UI, which interacts with the smart contract.
5. User can manage their existing policies (view details, pay premiums) through the UI, which interacts with the smart contract.
6. Upon filing a claim, the smart contract integrates with an oracle service to verify the claim.
7. Verified claims are voted on by other policyholders or authorities through the smart contract.
8. Approved claims trigger the smart contract to automatically distribute the insurance benefits to the beneficiaries.
9. The UI displays transaction history and relevant information for the policyholder and beneficiaries.

**7. Security Considerations**

* Secure user authentication and authorization mechanisms are needed to prevent unauthorized access.
* The smart contract code should be rigorously audited to identify and address potential vulnerabilities.
* Secure storage solutions should be employed for private user data (e.g., off-chain storage with encryption).

**8. Integration Requirements**

* The application needs to integrate with a blockchain platform that supports smart contracts.
* Integration with an external oracle service is necessary for claim verification.
* Integration with a decentralized voting mechanism for claim validation.

**9. Success Criteria**

* The application allows users to purchase and manage their life insurance policies securely on the blockchain.
* The application facilitates secure and transparent claims processing and benefit distribution.
* The application is user-friendly and provides a smooth user experience.

**10. References**

* Solidity Programming Language Documentation (<https://docs.soliditylang.org/>)
* Blockchain Platforms with Smart Contract Support (e.g., Ethereum, Binance Smart Chain)