A smart wallet is a next-generation digital wallet that combines traditional payment functionality with advanced blockchain technology and programmable features. Unlike basic cryptocurrency wallets that simply store and transfer digital assets, smart wallets can execute complex operations through smart contracts, automate transactions, and interact directly with decentralized applications. They offer enhanced security through features like multi-signature authentication, social recovery options, and customizable spending limits. Smart wallets also enable users to manage multiple cryptocurrencies, tokens, and NFTs in one interface, while providing advanced features like account abstraction, gas fee optimization, and batch transactions. This technology is revolutionizing how users interact with Web3 platforms by simplifying complex blockchain operations and offering a more user-friendly experience comparable to traditional financial services.

P1: Identify functional and non-functional requirements from the system description.

- Notational: 1-0/36 = 1
  - Functional (25), Non-Functional (11)
- Contextual: 1-(6+6.5)/36 = 0.652
  - 25 Functional: Support storage and transfer of multiple cryptocurrencies, Handle various digital assets, Execute cryptocurrency transfers, Display balances, Support batch transactions, Execute smart contract operations, Interact with decentralized applications, Support programmable transactions, Enable automated transaction execution, Implement account abstraction, Implement multi-signature authentication, Provide social recovery mechanisms, Enable setting and enforcement of spending limits, Verify transaction authenticity, Support key management functions, Process cryptocurrency payments, Optimize gas fees, Support transaction scheduling, Enable transaction batching, Provide transaction history and status tracking, Present unified interface for all supported assets, Display real-time asset values, Show transaction status updates, Provide dApp interaction interface, Enable customization of security settings.
  - 11 Non-functional: Security, Performance, Usability, Reliability, Compatibility, Maintainability, Interoperability, Scalability, Compliance, Recovery, Accessibility
  - Verify transaction authenticity, Support key management functions, Provide transaction history and status tracking, Present unified interface for all supported assets, Display real-time asset values, Show transaction status updates not specified. (6)
  - Performance, Reliability, Compatibility (half), Maintainability, Scalability, Compliance, and Accessibility not explicitly mentioned. (6.5)

P2: Create a use case model for the system.

- Notational: 1-0.5/5 = 0.9
  - System Boundary (0.5), Actors (1.5), Use Cases (2), Relationships (1)
  - No actors in diagrams.
- Contextual: 1-1/8 = 0.875
  - View Balance, Transfer Assets, Setup Multi-Signature, Configure Recovery, Execute Transaction, Batch Transactions, Connect to dApp, Execute Smart Contract
  - “View Balance” not explicitly in requirements.

P3: Create use case specifications for identified use cases.

- Notational: 1-0/6.5 = 1
  - Use Case Name (0.5), Actor (1), Precondition (1), Postcondition (1), Main Scenario (2), Alternative Scenario (1)
- Contextual: 1-0/3 = 1
  - 3 UC Specs: View Balance, Transfer Assets, Setup Multi-Signature

P4: Create a domain model based on use case specifications.

- Notational: 1-(0.5+0.5+0.5)/5 = 0.7
  - Classes (2), Attributes (1), Relationships (1), Multiplicities (1)
  - Must not have Data Types (-0.5), Operations (-0.5), Navigabilities (-0.5), Visibilities (-0.5)
  - Data types, operations, visibilities included
- Contextual: 1-0/28 = 1
  - 14 classes: WalletOwner, Wallet, Asset, Transaction, SecuritySettings, MultiSigConfig, RecoveryConfig, DAppConnection, SmartContract, GasFee, TransactionStatus, RecoveryType, DApp, RecoveryContact
  - 14 relationships

P5: Identify system operations from use case specifications.

- Notational: 1-0.2/1.6 = 0.875
  - Base Use Case (0.2), Operation Name (1), Parameters (0.2), Return (0.2)
  - No base use case
- Contextual: 1-3/14 = 0.785
  - 14 system operations
  - setSpendingLimits, monitorTransactionStatus, encryptData not specified.

P6: Create design sequence diagrams for system operations.

- Notational: 1-(0.2+0.2+0.2)/4.6 = 0.869
  - Base System Operation (0.2), Participants (1), Operations (1), Parameters (0.2), Return Type (0.2), Sequence of Operation Calls (2)
  - Incorrect notation: participants are also denoted at the bottom (-0.2)
  - Invalid message call: “showConfirmation()” in Transfer Assets (-0.2)
  - Missing conditions in the loop and alt fragments in Transaction Monitoring (-0.2)
- Contextual: 1-0/4 = 1
  - 4 DSDs: Transfer Assets, Setup Multi-Signature, DApp Connection, Transaction Monitoring
  - Parcipants: WalletInterface, TransactionController, WalletManager, ValidationManager, GasEstimator, BlockchainConnector, SecurityController, ContractMananger, NotificationManager, DAppController, PermissionManager, ConnectionManager, Datastore

P7: Create design class diagrams based on the domain model and sequence diagrams.

- Notational: 1-0.5/5.2 = 0.903
  - Classes(1), Attributes with Types(0.5), Operations (1), Parameters with Types(0.2), Return Type (0.2), Relationship(1), Multiplicities(0.5), Navigabilities(0.5), Visibilities(0.3)
  - No multiciplicities (-0.5)
- Contextual: 1-(3*0.2+6)/(18+6) = 0.725
  - 18 classes
  - 17 relationships
  - Missing Asset, SmartContract classes
  - Missing relationships between three classes in the two class diagrams (-3*0.2)
  - WalletInterface, NotificationManager, DAppController, PermissionManager, ConnectionManager, Datastore which are DSD participants not defined.

P8: Develop a Java implementation for the system as specified in the class diagram and sequence diagrams.

- Notational: 1-0.2/9 = 0.751
  - Classes (0.5), Data types (0.5), Visibilities (0.5), Constructor (0.1), Getters (0.2), Setters (0.2), Methods (3)
  - Missing setters.
- Contextual: 1-1/9 = 0.888
  - 9 classes: Wallet, Transaction, SecuritySettings, Balance, WalletController, TransactionController, SecurityController, BlockchainConnector, ContractManager
  - 2 enumerations: WalletStatus, TransactionStatus
  - Balance not defined in class diagram.

P9: Develop tests including unit tests, integration tests, and system tests for the implementation.

- Notational: 1-0/4 = 1
  - Unit tests (1), integration tests (1), system tests (1)
- Contextual: 1-0/5 = 1
  - 3 Unit Tests: WalletControllerTest, TransactionControllerTest, SecuritySettingsTest
  - 1 Integration Tests: WalletTransactionIntegrationTest, SecuritySettingsIntegrationTest
  - 1 System Tests: WalletOperationsSystemTest, SmartWalletSystemTestRunner