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: 1-0/18 = 1
  - Functional, Non-Functional
- Contextual: 1-3/16 = 0.812
  - 8 Functional: Digital Asset Storage and Transfer, Smart Contract Execution, Automated Transactions, Interaction with Decentralized Applications, Enhanced Security Features, Account Abstraction, Gas Fee Optimization, Batch Transactions.
  - 8 Non-functional: Security, Usability, Performance, Reliability, Compatibility, Scalability, Maintainability, Interoperability.
  - Reliability, Scalability, Maintainability not explicitly covered.

P2: Create a use case model for the system.

- Notational: 1-0/5 = 1
  - System Boundary (0.5), Actors (1.5), Use Cases (2), Relationships (1)
- Contextual: 1-0/7 = 1
  - 7 use cases: 1. Manage Digital Assets, 2. Execute Smart Contracts, 3. Automate Transactions, 4. Interact with dApps, 5. Enhance Security, 6. Optimize Gas Fees, 7. Execute Batch Transactions
  - All use cases covered

P3: Create use case specifications for identified use cases.

- Notational: 1-(1*0.3)/6.5 = 0.953
  - Use Case Name (0.5), Actor (1), Precondition (1), Postcondition (1), Main Scenario (2), Alternative Scenario (1)
  - Alternative scenario having no reference to basic flow.
- Contextual: 1-0/7 = 1
  - 7 UC Specs: Manage Digital Assets, Execute Smart Contracts, Automate Transactions, Interact with dApps, Enhance Security, Optimize Gas Fees, Execute Batch Transactions

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

- Notational: 1-1/5 = 0.74
  - Classes (2), Attributes (1), Relationships (1), Multiplicities (1)
  - Must not have Data Types (-0.5), Operations (-0.5), Navigabilities (-0.5), Visibilities (-0.5)
  - No multiplicities
- Contextual: 1-0.3/8 = 0.962
  - 8 classes
  - 7 relationships
  - Transaction in the relationship with GasOptimization not defined (-0.3).

P5: Identify system operations from use case specifications.

- Notational: 1-0/1.6 = 1
  - Base Use Case (0.2), Operation Name (1), Parameters (0.2), Return (0.2)
- Contextual: 1-1/8 = 0.875
  - 8 system operations
  - Authenticate User not defined in use cases

P6: Create design sequence diagrams for system operations.

- Notational: 1-(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)
  - No concrete operation name, no parameter, no return
- Contextual: 1-0/8 = 1
  - 8 DSDs
  - Participants: User, Smart Wallet, Authentication Service, Blockchain Network, Automation Engine, dApp, Security Engine

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

- Notational: 1-(0.2+0.2+0.5+0.5+0.3)/5.2 = 0.673
  - 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 parameters, no return types, no multiplicities, no navigabilities, no visibilities.
- Contextual: 1-4/(8+4) = 0.666
  - 8 classes: User, DigitalAsset, SmartContract, AutomatedTransaction, dApp, SecuritySetting, GasOptimization, BatchTransaction
  - Authentication Service, Blockchain Network, Automation Engine, Security Engine 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+0.2)/5 = 0.92
  - Classes (0.5), Data types (0.5), Visibilities (0.5), Constructor (0.1), Getters (0.2), Setters (0.2), Methods (3)
  - Missing getters, setters.
- Contextual: 1-(8*0.8)/8 = 0.2
  - 8 classes: User, DigitalAsset, SmartContract, AutomatedTransaction, dApp, SecuritySetting, GasOptimization, BatchTransaction
  - All methods missing implementation.

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

- Notational: 1-(1*0.5+1*0.5)/3 = 0.666
  - Unit tests (1), integration tests (1), system tests (1)
  - Showing partial implementation for unit and integration tests.
- Contextual: 1-(1*0.5)/5 = 0.9
  - 2 Unit Tests: UserTest, DigitalAssetTest
  - 1 Integration Tests: IntegrationTest
  - 1 System Tests: SystemTest
  - Assert statements in SystemTest not testing integration.