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: 1-0/12 = 1
  - Functional, Non-Functional
- Contextual: 1-4/15 = 0.733
  - 10 Functional: Payment Functionality (sending/receiving cryptocurrencies, support for various payment methods), Smart Contract Interaction, Security Features (multi-signature authentication, social recovery options, customizable spending limits), Multi-Asset Support, Advanced Features (account abstraction, gas fee optimization, batch transactions).
  - 5 Non-functional: Security (robust measures, compliance, audits), Performance (fast and reliable transaction processing), Usability (user-friendly interface, clear documentation), Reliability (high availability, fault tolerance), Scalability (handling growing user demand, efficient resource utilization).
  - Security, Performance, Reliability, Scalability not explicitly mentioned in requirements.

P2: Create a use case model for the system.

- Notational: 1-5/5 = 0
  - System Boundary (0.5), Actors (1.5), Use Cases (2), Relationships (1)
  - A class diagram generated instead of a use case model (-5).
- Contextual: 1-1/5= 0.8
  - 0 use cases: 1. Create Wallet, 2. Fund Wallet, 3. Send Transaction, 4. Interact with Smart Contract, 5. Manage Wallet
  - Create Wallet not explicitly 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 label referencing to basic flow.
- Contextual: 1-0/3 = 1
  - 3 UC Specs: Create Wallet, Fund Wallet, Send Transaction

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

- Notational: 1-0/5 = 1
  - Classes (2), Attributes (1), Relationships (1), Multiplicities (1)
  - Must not have Data Types (-0.5), Operations (-0.5), Navigabilities (-0.5), Visibilities (-0.5)
- Contextual: 1-0/5 = 1
  - 5 classes: User, Wallet, Transaction, Currency, Amount
  - 2 relationships: User-Wallet, Transaction-Wallet

P5: Identify system operations from use case specifications.

- Notational: 1-(0.2+0.2+0.2)/1.6 = 0.625
  - Base Use Case (0.2), Operation Name (1), Parameters (0.2), Return (0.2)
  - No base use case (-0.2), no parameters (-0.2), no return (-0.2).
- Contextual: 1-7/17 = 0.588
  - 17 system operations: Create Wallet duplicated
  - 7 not covered in UC: View Transaction History, Verify User Credentials, Authorize User Actions, Retrieve Wallet Information, Deploy Smart Contract, Execute Smart Contract Function, Monitor Smart Contract Events not specified

P6: Create design sequence diagrams for system operations.

- Notational: 1-(0.2+0.2+1*0.8)/4.6 = 0.739
  - Base System Operation (0.2), Participants (1), Operations (1), Parameters (0.2), Return Type (0.2), Sequence of Operation Calls (2)
  - No parameters, no return
  - No inner system participants
- Contextual: 1-(3*0.6)/3 = 0.4
  - 3 DSDs: Create Wallet, Fund Wallet, Send Transaction
  - Not DSDs, but SSDs.

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

- Notational: 1-(1+0.5+0.5)/5.2 = 0.615
  - 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 relationships, no multiplicities, no navigabilities
- Contextual: 1-0/5 = 1
  - 5 classes: User, Wallet, Transaction, Currency, Amount
  - 0 relationships

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

- Notational: 1-(0.1+0.2+0.2+1)/5 = 0.7
  - Classes (0.5), Data types (0.5), Visibilities (0.5), Constructor (0.1), Getters (0.2), Setters (0.2), Methods (3)
  - No constructors, getters, setters.
  - User methods having no implementation.
- Contextual: 1-0/5 = 1
  - 5 classes: User, Wallet, Transaction, Currency, Amount

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

- Notational: 1-(1*0.7)/3 = 0.766
  - Unit tests (1), integration tests (1), system tests (1)
  - No concrete system tests. Only textual description
- Contextual: 1-0/4 = 1
  - 3 Unit Tests
  - 1 Integration Tests