# **TECHNICAL REQUIREMENTS FOR E WALLET MANAGEMENT SYSTEM**

## **1. Introduction**

An e-wallet management system allows users to make electronic transactions, manage digital payments, and handle various financial activities seamlessly. This document outlines the essential technical requirements for developing a robust e-wallet management system.

## **2. Software Requirements**

### **2.1 Frontend**

* **Technologies**:
  + **HTML**: For structuring the content.
  + **CSS**: For styling and layout.
  + **JavaScript**: For interactivity and dynamic content.
  + **Frameworks**:
    - **React**: For building user interfaces.
    - **Angular**: For creating single-page applications.
    - **Vue.js**: For building interactive web interfaces.
  + **Mobile Development**:
    - **iOS**: Swift programming language for native app development.
    - **Android**: Kotlin or Java for native app development.

### **2.2 Backend**

* **Programming Languages**:
  + **Python**: Popular for server-side applications.
  + **Java**: Known for its scalability and security.
  + **Node.js**: Ideal for event-driven applications.
  + **Ruby on Rails**: For rapid application development.
* **Frameworks**:
  + **Express.js** (Node.js): For building web applications.
  + **Django** (Python): For a high-level Python web framework.
  + **Spring** (Java): For building enterprise-level applications.
* **Database Management System**:
  + **Relational Databases**:
    - **MySQL**: For structured data storage.
    - **PostgreSQL**: Known for advanced features.
    - **Oracle**: For enterprise-level applications.
  + **NoSQL Databases**:
    - **MongoDB**: For flexible data storage.
    - **Firebase**: For real-time database capabilities.

### **2.3 Security**

* **Encryption**:
  + **SSL/TLS**: For secure data transmission.
  + **AES**: Advanced Encryption Standard for secure data storage.
* **Authentication**:
  + **OAuth**: For secure delegated access.
  + **JWT (JSON Web Tokens)**: For user authentication.
  + **Multi-Factor Authentication (MFA)**: To enhance security.
* **Firewalls and Anti-DDoS Measures**: Protect the system from cyber threats.

## **3. Hardware Requirements**

### **3.1 Server**

* **Hosting**:
  + **Dedicated Servers**: For high-performance applications.
  + **Cloud Services**:
    - **AWS (Amazon Web Services)**: Scalable cloud hosting.
    - **Google Cloud**: For machine learning and data analytics.
    - **Microsoft Azure**: Integrated cloud services.
* **Specifications**:
  + Minimum 4-8 CPU cores.
  + 16-32 GB RAM.
  + 100 GB SSD storage, scalable based on user load.

### **3.2 Client Devices**

* **Mobile Devices**: Support for iOS and Android devices.
* **Web Browsers**: Compatibility with major browsers like Chrome, Firefox, and Safari.

## **4. Network Requirements**

* **Internet Connection**: Stable and high-speed internet for both server and client applications.
* **API Integration**: Ability to connect with third-party services, including payment gateways and KYC providers.

## **5. Database Requirements**

* **Database Design**:
  + User profiles, transaction histories, account management.
* **Data Backup and Recovery**:

Regular backups to prevent data loss, including off-site backups.

## **6. Compliance and Regulatory Requirements**

* **KYC (Know Your Customer)**: Verification process to comply with financial regulations.
* **AML (Anti-Money Laundering)**: Measures to detect and prevent money laundering.
* **PCI DSS Compliance**: Adherence to Payment Card Industry Data Security Standards when handling credit card transactions.

## **7. Payment Gateway Integration**

* **Supported Payment Methods**:
  + Credit/Debit Cards.
  + Bank Transfers.
  + Cryptocurrencies.
* **APIs**: Integration with payment processors such as PayPal, Stripe, or local payment gateways for seamless transaction processing.

## **8. Additional Features**

* **User Analytics**: Tools for tracking user behavior and engagement metrics.
* **Notification System**: Push notifications for transactions and account updates (e.g., using Firebase Cloud Messaging).
* **Support System**: In-app support or chatbots for user assistance.

## **9. Testing and Quality Assurance**

* **Testing Frameworks**:
  + **Selenium**: For automated testing of web applications.
  + **Jest**: For JavaScript testing.
  + **Mocha**: For Node.js testing.
* **Load Testing**:
  + Tools like **JMeter** or **LoadRunner** to simulate high traffic and ensure system stability.

## **10. Deployment and DevOps**

* **CI/CD Tools**:
  + **Jenkins** or **GitHub Actions** for continuous integration and deployment.
* **Containerization**:
  + **Docker**: For consistent deployment across various environments.
* **Monitoring Tools**:
  + **New Relic** or **Grafana** for performance monitoring and real-time alerts.