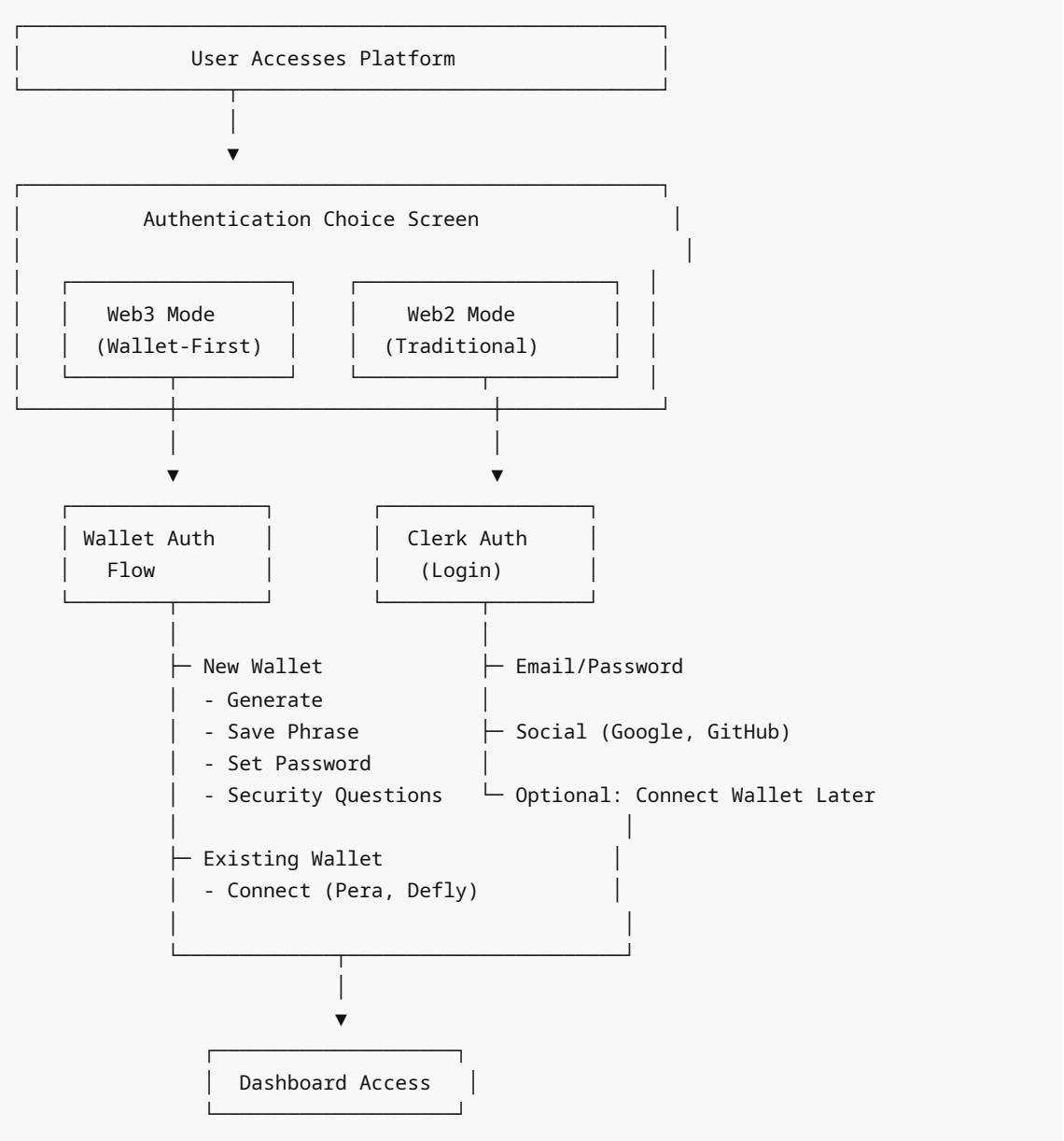
Wallet-Optional Flow Implementation Guide

Overview

This document describes the implementation of **Option A: Wallet-Optional Flow**, which allows users to choose between Web3 (wallet-first) and Web2 (traditional email/social) authentication methods.

Architecture

Authentication Flow Diagram



Implementation Details

1. New Files Created

Authentication Screens

- `src/views/authentication/AuthChoice.vue`
  - Presents users with choice between Web3 and Web2 authentication
  - Features comparison and help dialog
  - Saves user preference to localStorage
- `src/views/authentication/WalletAuth.vue`
  - Complete Web3 wallet-first authentication flow
  - Supports both new wallet creation and existing wallet connection
  - Implements DFD workflow: Wallet → Password → Security Questions → Access

Security Services

- `src/services/walletEncryption.ts`
  - Client-side encryption for wallet data using Web Crypto API
  - AES-GCM encryption with PBKDF2 key derivation
  - 100,000 iterations for strong password protection
  - Never sends unencrypted seed phrases to server
- `src/services/securityQuestions.ts`
  - Manages security questions for account recovery
  - Stores hashed answers (SHA-256) - never plain text
  - Requires 3 unique questions for setup
  - Supports verification and recovery flows

2. Modified Files

Routing

- `src/router/PublicRoutes.ts`
  - Added `/auth-choice` route
  - Added `/wallet-auth` route
  - Both routes accessible without authentication

3. Authentication Modes

Web3 Mode (Wallet-First)

New Wallet Creation Flow:

1. User clicks "Create New Wallet"
2. Wallet generated client-side using `generateAlgorandWallet()`
3. 25-word mnemonic phrase displayed (user must save)
4. User sets password to encrypt wallet
5. User answers 3 security questions
6. Wallet encrypted and stored in localStorage
7. Redirected to dashboard

Existing Wallet Connection Flow:

1. User clicks "Connect Existing Wallet"
2. ConnectWallet modal opens
3. User selects provider (Pera, Defly, WalletConnect)

4. Wallet connects via @txnlab/use-wallet-vue
5. Redirected to dashboard

Web2 Mode (Traditional)

Flow:

1. User clicks "Continue with Email/Social"
2. Redirected to `/login` (Clerk authentication)
3. User logs in with email/password or OAuth
4. Optional: Connect wallet later from settings
5. Redirected to dashboard

4. Security Features

Client-Side Encryption

```
// Password-based encryption (never sent to server)
const walletData = {
  mnemonic: "...",
  address: "...",
  createdAt: Date.now()
};

const encrypted = await encryptWallet(walletData, userPassword);
storeEncryptedWallet(encrypted); // Stored in localStorage
```

Key Security Principles

1. ✔ All encryption happens client-side
2. ✔ Passwords never leave the browser
3. ✔ Seed phrases never transmitted to server
4. ✔ Uses PBKDF2 with 100,000 iterations
5. ✔ AES-GCM for authenticated encryption
6. ✔ Security answers are hashed (SHA-256)
7. ✔ No plain text sensitive data storage

Recovery Options

- **Password Recovery**: Use security questions to decrypt wallet
- **Wallet Export**: Export encrypted backup as JSON
- **Wallet Import**: Import encrypted backup from JSON

5. Data Storage

localStorage Keys

- `auth_mode` : 'web3' | 'web2' (user preference)
- `encrypted_wallet` : Encrypted wallet data (Web3 mode only)
- `security_answers` : Hashed security question answers
- `web3_authenticated` : Boolean flag for Web3 auth status
- `active_wallet` : Current connected wallet address

6. User Experience Flow

First-Time User (Web3 Mode)

```
1. Land on platform
2. Click "Get Started" → Redirects to /auth-choice
3. Choose "Web3 Mode"
4. Choose "Create New Wallet"
5. Click "Generate Wallet"
6. Save 25-word recovery phrase
7. Confirm saved (checkbox)
8. Set strong password (8+ chars)
9. Answer 3 security questions
10. Success! → Dashboard
```

First-Time User (Web2 Mode)

```
1. Land on platform
2. Click "Get Started" → Redirects to /auth-choice
3. Choose "Web2 Mode"
4. Sign up with email or Google/GitHub
5. Success! → Dashboard
6. (Optional) Connect wallet later
```

Returning User (Web3 Mode)

```
1. Navigate to /wallet-auth
2. Select "Connect Existing Wallet"
3. Choose wallet provider
4. Approve connection in wallet app
5. Success! → Dashboard
```

Returning User (Web2 Mode)

```
1. Navigate to /login
2. Enter email/password or use OAuth
3. Success! → Dashboard
```

API Integration

Wallet Registration Endpoint

```
// POST /api/wallet/register
{
  "walletAddress": "ALGO_ADDRESS_HERE",
  "publicKey": "...",
  "authMode": "web3"
}
```

Wallet Authentication Endpoint

```
// POST /api/wallet/auth
{
  "walletAddress": "ALGO_ADDRESS_HERE",
  "signature": "MESSAGE_SIGNED_BY_WALLET",
  "timestamp": 1699999999
}
```

Configuration

Environment Variables

No new environment variables required. The system works with existing Clerk configuration.

Feature Flags

```
// To enable/disable authentication modes
const ENABLE_WEB3_MODE = true;  // Set to false to hide Web3 option
const ENABLE_WEB2_MODE = true;  // Set to false to hide Web2 option
```

Testing Checklist

Web3 Mode - New Wallet

- [ ] Wallet generation works
- [ ] Mnemonic phrase displays correctly (25 words)
- [ ] Password validation works
- [ ] Password strength indicator functions
- [ ] Security questions can be selected
- [ ] No duplicate questions allowed
- [ ] Encrypted wallet saved to localStorage
- [ ] User redirected to dashboard
- [ ] Wallet shows as connected

Web3 Mode - Existing Wallet

- [ ] ConnectWallet modal opens
- [ ] Pera wallet connection works
- [ ] Defly wallet connection works
- [ ] WalletConnect works
- [ ] User redirected to dashboard
- [ ] Wallet shows as connected

Web2 Mode

- [ ] Redirects to Clerk login
- [ ] Email/password login works
- [ ] Google OAuth works

- [ ] GitHub OAuth works
- [ ] User redirected to dashboard
- [ ] Can optionally connect wallet later

Security

- [ ] Encrypted wallet data is not readable in localStorage
- [ ] Password is never stored in plain text
- [ ] Security answers are hashed
- [ ] Seed phrase never sent to server
- [ ] Password recovery with security questions works

UI/UX

- [ ] Auth choice screen displays correctly
- [ ] Theme toggle works on all auth screens
- [ ] Mobile responsive design works
- [ ] Loading states display properly
- [ ] Error messages are clear
- [ ] Success messages display
- [ ] Back navigation works

Migration Guide

For Existing Users

Existing users with Clerk accounts continue to work without changes. They can:

1. Continue using Web2 mode (Clerk)
2. Optionally connect a wallet to access blockchain features
3. Switch to Web3 mode if desired (requires wallet setup)

Data Migration

No data migration required. Both authentication systems work in parallel.

Future Enhancements

Planned Features

1. **Wallet Recovery Flow**: Complete UI for password recovery using security questions
2. **Multi-Wallet Support**: Allow users to connect multiple wallets
3. **Hardware Wallet Support**: Ledger, Trezor integration
4. **Biometric Authentication**: Face ID, Touch ID for mobile
5. **2FA/MFA**: Additional security layer for both modes
6. **Social Recovery**: Allow trusted contacts to help recover account
7. **Account Linking**: Link Clerk account with wallet address

Backend Enhancements

1. **Wallet-Signed Authentication**: Use wallet signatures for API authentication
2. **Session Management**: JWT tokens signed by wallet

3. **Rate Limiting**: Prevent brute force attacks on recovery
4. **Audit Logging**: Log all authentication attempts
5. **Analytics**: Track which auth mode users prefer

Troubleshooting

Common Issues

Issue: "Failed to encrypt wallet"

- **Solution**: Ensure browser supports Web Crypto API. Update to modern browser.

Issue: "Security questions not saving"

- **Solution**: Check that exactly 3 unique questions are answered.

Issue: "Wallet connection fails"

- **Solution**: Ensure wallet app is installed and unlocked.

Issue: "Password recovery not working"

- **Solution**: Answers are case-insensitive. Check for typos.

Best Practices

For Users

1. **Always save your recovery phrase** in a secure location
2. **Use a strong password** (12+ characters, mix of types)
3. **Remember security question answers** exactly as entered
4. **Never share your seed phrase** with anyone
5. **Enable 2FA** when available

For Developers

1. **Never log sensitive data** (passwords, seed phrases)
2. **Always validate user input** before encryption
3. **Use constant-time comparison** for security answers
4. **Implement rate limiting** on recovery attempts
5. **Regular security audits** of encryption implementation

Compliance & Legal

Data Protection

- Wallet data encrypted client-side only
- No PII stored with Web3 mode (unless user provides)
- Security answers hashed - not reversible
- GDPR compliant (user controls all data)

User Responsibilities

- Users are responsible for:
  - Safeguarding recovery phrases
  - Remembering passwords
  - Keeping security answers secure

    - Understanding wallet custody implications

Support Documentation

For End Users

Create help articles covering:

1. "What is a wallet?"
2. "Web3 vs Web2: Which should I choose?"
3. "How to save my recovery phrase"
4. "How to recover my account"
5. "How to connect an existing wallet"

For Administrators

1. Monitoring authentication metrics
2. Handling support requests
3. Security incident response
4. Key rotation procedures

Conclusion

The Wallet-Optional Flow successfully implements Option A, providing:

- ✅ Flexibility for both Web2 and Web3 users
- ✅ Strong security with client-side encryption
- ✅ DFD-compliant workflow for Web3 mode
- ✅ Seamless existing user experience
- ✅ No breaking changes to current system
- ✅ Foundation for future enhancements

This implementation maintains security best practices while offering users the freedom to choose their preferred authentication method.