# SSA Form-Assist: Technical Specification & Implementation Plan

## Executive Summary

A Progressive Web Application (PWA) that simplifies SSA Adult Function Report (SSA-3373) completion by mapping Blue Book disability listings to functional limitations, using AI to generate appropriate phrasing while maintaining HIPAA-level security through local-first, encrypted data storage.

---

## 1. Technical Architecture

### 1.1 Core Technology Stack

#### Frontend Framework

- **Recommendation: React 18+** with TypeScript
  - Rationale: Mature ecosystem, excellent PWA support, strong typing for complex data models
  - Alternative: Svelte (lighter weight, but smaller ecosystem for this use case)

#### State Management

- **Recommendation: Zustand** or Context API + useReducer
  - Rationale: Simpler than Redux for local-first architecture, sufficient for this scope

#### PWA Infrastructure

- **Service Worker**: Workbox 7+ for asset caching and offline functionality
- **Manifest**: Web App Manifest for installability
- **Storage**: IndexedDB via `idb` wrapper library

#### UI Framework

- **Recommendation: Tailwind CSS** + **Radix UI** or **shadcn/ui**
  - Rationale: Accessibility-first components critical for disabled users, highly customizable

### 1.2 Security Architecture

#### Encryption Implementation

```
typescript

// Key Derivation
Algorithm: PBKDF2 (consider Argon2 for better resistance)
Parameters:
  - Iterations: 600,000+ (OWASP 2023 recommendation)
  - Salt: 16 bytes, cryptographically random
  - Hash: SHA-256
  - Derived Key Length: 256 bits

// Data Encryption
Algorithm: AES-256-GCM
Implementation: Web Crypto API (native browser support)
```

### Critical Security Measures

1. **Zero-Knowledge Architecture**: Passphrase never leaves client, never stored
2. **Memory Management**: Clear sensitive data from memory after use
3. **PBKDF2 Salt Storage**: Store salt unencrypted in IndexedDB (required for key derivation)
4. **Session Management**: Keep MEK in memory only during active session
5. **Auto-lock**: Implement timeout (15-30 min) requiring passphrase re-entry

## 1.3 Data Storage Architecture

### IndexedDB Schema

```typescript
Database: 'SSAFormAssist'
Version: 1

ObjectStores:
  1. 'config' (keyPath: 'id')
     - id: 'user-config'
     - salt: Uint8Array (16 bytes)
     - selectedLLM: 'gemini' | 'openai' | 'claude'
     - encryptedAPIKeys: string (AES-encrypted JSON)
     - encryptedCloudConfig: string (AES-encrypted JSON)
     - lastModified: timestamp

  2. 'reports' (keyPath: 'id', autoIncrement: false)
     - id: UUID
     - encryptedData: string (entire report encrypted)
     - title: string (encrypted separately for list view)
     - lastModified: timestamp
     - lastSyncTimestamp: timestamp | null
     - syncStatus: 'synced' | 'pending' | 'conflict'

  3. 'bluebook-cache' (keyPath: 'listingId')
     - listingId: string
     - category: string
     - title: string
     - criteria: JSON
     - functionalDomains: string[]
     - keywords: string[]
     - version: string (for update tracking)
```

---

# 2. Blue Book Data Model

## 2.1 Static Asset Structure

### File Organization

```
/public/data/bluebook/
  ├── index.json (full listing index)
  ├── categories/
  │   ├── 1.00-musculoskeletal.json
  │   ├── 2.00-special-senses.json
  │   ├── 3.00-respiratory.json
  │   └── ... (13 categories total)
  └── version.json (SSA Blue Book version tracking)
```

### Data Structure per Listing

typescript

```typescript
interface BlueBookListing {
  listingId: string;          // "1.04"
  category: string;           // "1.00 Musculoskeletal System"
  title: string;              // "Disorders of the spine"
  fullCitation: string;       // For legal reference

  criteria: {
    letter: string;           // "A", "B", etc.
    description: string;      // Full criterion text
    medicalFindings: string[]; // Required medical evidence
    functionalLoss: {
      domain: 'physical' | 'mental' | 'social' | 'concentration';
      specificLimitation: string;
      severity: 'marked' | 'extreme' | 'moderate';
    }[];
  }[];

  // Critical for AI prompt generation
  functionalMapping: {
    ssaQuestionId: string;    // Maps to SSA-3373 question number
    relevantCriteria: string[]; // Which criteria letters apply
    keyPhrases: string[];     // Seed phrases for AI
  }[];

  comorbidityTags: string[];  // For compound disability logic
  lastUpdated: string;        // Blue Book revision date
}
```

## 2.2 SSA-3373 Question Mapping

### Create a separate mapping file:

typescript

```typescript
// /public/data/ssa-3373-mapping.json
interface SSA3373Question {
  questionId: string;
  sectionNumber: string;
  questionText: string;
  expectationType: 'narrative' | 'checklist' | 'scale';
  functionalDomain: string[];
  characterLimit: number;

  // Links to Blue Book
  relevantListingCategories: string[];

  // AI Prompt hints
  promptContext: {
    tone: 'objective' | 'descriptive';
    focus: string[];  // ["daily activities", "duration", "frequency"]
    avoidances: string[];  // ["speculation", "medical diagnosis"]
  };
}
```

---

# 3. AI Integration Architecture

## 3.1 Abstraction Layer

typescript

```typescript
// services/llm/LLMService.ts
interface LLMProvider {
  generateResponse(prompt: PromptConfig): Promise<LLMResponse>;
  validateAPIKey(key: string): Promise<boolean>;
  estimateCost(inputTokens: number, outputTokens: number): number;
}

class LLMService {
  private providers: Map<string, LLMProvider>;

  constructor() {
    this.providers.set('gemini', new GeminiProvider());
    this.providers.set('openai', new OpenAIProvider());
    this.providers.set('claude', new ClaudeProvider());
  }

  async generate(
    provider: string,
    userInputs: FunctionalInputs,
    bluebookContext: BlueBookListing[],
    ssaQuestion: SSA3373Question
  ): Promise<GeneratedText> {
    const prompt = this.buildPrompt(userInputs, bluebookContext, ssaQuestion);
    return this.providers.get(provider).generateResponse(prompt);
  }
}
```

## 3.2 Prompt Engineering Strategy

### Structured Prompt Template

```typescript
interface PromptConfig {
  system: {
    role: "You are an expert assistant helping complete SSA Adult Function Reports";
    constraints: [
      "Use objective, professional language",
      "Focus on functional limitations, not diagnoses",
      "Reference activities of daily living",
      "Use present tense",
      "Avoid medical speculation"
    ];
  };

  context: {
    bluebookCriteria: string;      // Relevant listing text
    userFunctionalInputs: {        // Plain language inputs
      question: string;
      userAnswer: string;
      severity: 'mild' | 'moderate' | 'severe';
    }[];
  };

  task: {
    ssaQuestion: string;
    characterLimit: number;
    outputFormat: "Generate 5-7 sentences addressing the question";
  };

  examples?: {                     // Few-shot learning
    input: string;
    goodOutput: string;
    reasoning: string;
  }[];
}
```

## 3.3 API Key Management

### Security Best Practices

1. **Storage**: Encrypt with MEK before storing in IndexedDB
2. **Transmission**: Never log or transmit keys except to official APIs
3. **Validation**: Test keys with minimal API call before saving
4. **Scope Limitation**: Store only necessary scopes (read/write for cloud sync)

### Cost Management

```typescript
interface CostEstimator {
  estimateTokens(text: string): number;
  calculateCost(provider: string, inputTokens: number, outputTokens: number): {
    amount: number;
    currency: 'USD';
    breakdown: {
      input: number;
      output: number;
    };
  };

  // Warn user before expensive operations
  shouldWarnUser(estimatedCost: number): boolean;
}
```

---

## 4. Cloud Synchronization Architecture

### 4.1 Sync Strategy: Local-First with Conflict Resolution

#### Sync Flow

```typescript
enum SyncStatus {
  SYNCED = 'synced',
  PENDING = 'pending',
  SYNCING = 'syncing',
  CONFLICT = 'conflict',
  ERROR = 'error'
}

interface SyncManager {
  // Triggered by: user action, connectivity change, periodic check
  async syncToCloud(reportId: string): Promise<SyncResult>;

  // Conflict resolution strategies
  resolveConflict(
    local: EncryptedReport,
    remote: EncryptedReport,
    strategy: 'last-write-wins' | 'user-choice' | 'merge'
  ): Promise<EncryptedReport>;

  // Background sync registration (if supported)
  registerBackgroundSync(): void;
}
```

### Conflict Resolution Logic

```typescript
// Last-Write-Wins (Recommended for MVP)
if (Math.abs(local.lastModified - remote.lastModified) > 60000) {
  // More than 1 minute apart - clear winner
  return local.lastModified > remote.lastModified ? local : remote;
} else {
  // Close timestamps - prompt user
  return await this.promptUserForChoice(local, remote);
}
```

## 4.2 Cloud Adapter Pattern

typescript

```typescript
interface CloudAdapter {
  name: string;
  authenticate(): Promise<AuthToken>;
  upload(filename: string, encryptedData: Blob): Promise<UploadResult>;
  download(filename: string): Promise<Blob>;
  list(prefix?: string): Promise<CloudFile[]>;
  delete(filename: string): Promise<boolean>;
}

class OneDriveAdapter implements CloudAdapter {
  private clientId: string;
  private msalInstance: PublicClientApplication;

  constructor(clientId: string) {
    this.clientId = clientId;
    this.msalInstance = new PublicClientApplication({
      auth: { clientId },
      cache: { cacheLocation: 'sessionStorage' }
    });
  }

  async authenticate(): Promise<AuthToken> {
    const request = {
      scopes: ['Files.ReadWrite', 'offline_access']
    };
    return await this.msalInstance.acquireTokenPopup(request);
  }

  async upload(filename: string, data: Blob): Promise<UploadResult> {
    const token = await this.getValidToken();
    const url = `https://graph.microsoft.com/v1.0/me/drive/root:/SSAFormAssist/${filename}:/content`;

    const response = await fetch(url, {
      method: 'PUT',
      headers: {
        'Authorization': `Bearer ${token.accessToken}`,
        'Content-Type': 'application/octet-stream'
      },
      body: data
    });

    return await response.json();
  }
}
```

**Filename Convention**

Format: report-{UUID}.encrypted
Example: report-a3b2c1d4-e5f6-7890-abcd-ef1234567890.encrypted

Metadata file: report-{UUID}.meta.json (stores last modified, title hash)

---

# 5. Implementation Plan

## Phase 1: Foundation (Weeks 1-3)

### Week 1: Project Setup & Security Core

- [ ] Initialize React + TypeScript + Vite project
- [ ] Configure PWA (Service Worker, Manifest)
- [ ] Implement encryption module (Web Crypto API wrapper)
- [ ] Create IndexedDB schemas and wrapper functions
- [ ] Build passphrase entry/setup flow
- [ ] Unit tests for encryption/decryption

### Week 2: Blue Book Data Integration

- [ ] Structure Blue Book JSON files (start with 3 categories)
- [ ] Build Blue Book browser component (search, hierarchy)
- [ ] Create SSA-3373 question mapping
- [ ] Implement data caching in Service Worker
- [ ] Build version checking mechanism

### Week 3: Core UI & Navigation

- [ ] Design system setup (Tailwind + component library)
- [ ] Dashboard (report list, create new)
- [ ] Report creation wizard (multi-step form)
- [ ] Settings page (LLM selection, API key management)
- [ ] Accessibility audit (WCAG 2.1 AA minimum)

## Phase 2: AI Integration (Weeks 4-5)

### Week 4: LLM Service Layer

- [ ] Implement LLMService abstraction
- [ ] Create provider implementations (Gemini, OpenAI, Claude)
- [ ] Build prompt engineering templates
- [ ] API key validation and storage
- [ ] Cost estimation module

**Week 5: Report Generation**

- [ ] Functional input collection forms
- [ ] AI generation trigger and loading states
- [ ] Generated text display and editing
- [ ] Copy-to-clipboard functionality
- [ ] Error handling and retry logic

## Phase 3: Cloud Sync (Weeks 6-7)

**Week 6: OneDrive Integration**

- [ ] Implement CloudAdapter interface
- [ ] OneDrive OAuth flow (MSAL)
- [ ] Upload/download encrypted reports
- [ ] Sync status indicators
- [ ] Token refresh handling

**Week 7: Sync Logic & Conflict Resolution**

- [ ] Background sync registration
- [ ] Connectivity detection
- [ ] Conflict detection and resolution UI
- [ ] Sync history/logging
- [ ] Manual sync trigger

## Phase 4: Polish & Testing (Weeks 8-9)

**Week 8: Testing & Bug Fixes**

- [ ] End-to-end testing (Playwright)
- [ ] Security audit (encryption, key management)
- [ ] Performance optimization (code splitting, lazy loading)
- [ ] Offline functionality testing
- [ ] Cross-browser testing (Chrome, Safari, Firefox, Edge)

**Week 9: Documentation & Deployment**

- [ ] User documentation (setup guide, FAQ)
- [ ] Privacy policy and disclaimers
- [ ] Deploy to hosting (Vercel/Netlify)
- [ ] Set up error monitoring (Sentry)
- [ ] Beta testing with target users

### Phase 5: Post-MVP Enhancements (Weeks 10+)

- [ ] Complete all 13 Blue Book categories
- [ ] Comorbidity logic (compound disabilities)
- [ ] Export to PDF functionality
- [ ] Multi-cloud support (Google Drive, Dropbox)
- [ ] Blue Book update notification system

---

## 6. Critical Implementation Recommendations

### 6.1 Security Non-Negotiables

1. **NEVER compromise on encryption**: If Web Crypto API fails, block functionality rather than storing plaintext
2. **Passphrase requirements**: Enforce minimum 12 characters, mix of character types
3. **Session timeout**: Clear MEK from memory after 15-30 minutes of inactivity
4. **Audit logging**: Log all encryption/decryption events (not data, just operations)

### 6.2 UX Priorities for Disabled Users

1. **Keyboard navigation**: Every feature must be fully keyboard accessible
2. **Screen reader support**: Proper ARIA labels, semantic HTML
3. **High contrast mode**: Support system preference, manual toggle
4. **Simplified language**: Avoid technical jargon in UI
5. **Progress indicators**: Clear feedback for long operations (AI generation, sync)

### 6.3 Performance Targets

- **Initial Load**: < 3 seconds on 3G connection
- **Time to Interactive**: < 5 seconds
- **AI Generation**: 5-15 seconds (show progress indicator)
- **Offline Capability**: 100% functional for core features
- **IndexedDB Operations**: < 100ms for reads, < 500ms for encrypted writes

### 6.4 Maintenance Strategy

#### Blue Book Updates

```typescript
interface UpdateChecker {
  checkForUpdates(): Promise<{
    currentVersion: string;
    latestVersion: string;
    updateAvailable: boolean;
    criticalUpdate: boolean;  // Affects legal accuracy
  }>;

  applyUpdate(version: string): Promise<void>;
  notifyUser(update: UpdateInfo): void;
}
```

#### Version Strategy

- Check for updates: On app launch + weekly background check
- Critical updates: Block AI generation until applied
- Minor updates: Notify user, allow deferral

---

## 7. Testing Strategy

### 7.1 Unit Tests (Jest + Testing Library)

```typescript
// Critical test coverage
- Encryption/decryption correctness
- PBKDF2 key derivation
- IndexedDB CRUD operations
- LLM prompt generation
- Cloud adapter methods
- Sync conflict resolution logic
```

### 7.2 Integration Tests

```typescript
// End-to-end user flows
- New report creation to AI generation
- Passphrase setup and login
- Cloud sync with simulated conflicts
- Offline -> Online transition
- API key validation and usage
```

### 7.3 Security Testing

```typescript
// Penetration testing scenarios
- IndexedDB inspection (data should be encrypted)
- Network traffic analysis (no plaintext PHI)
- XSS/CSRF attack vectors
- Service Worker cache poisoning
- API key extraction attempts
```

---

## 8. Deployment Architecture

### 8.1 Hosting Recommendation

#### Primary: Vercel

- Rationale: Excellent PWA support, automatic HTTPS, edge caching
- Configuration:

json

```
{
  "headers": [
    {
      "source": "/(.*)",
      "headers": [
        { "key": "X-Frame-Options", "value": "DENY" },
        { "key": "X-Content-Type-Options", "value": "nosniff" },
        { "key": "Referrer-Policy", "value": "strict-origin-when-cross-origin" },
        { "key": "Permissions-Policy", "value": "geolocation=(), microphone=(), camera=()" }
      ]
    }
  ]
}
```

**Alternative: Netlify** (similar configuration)

## 8.2 CI/CD Pipeline

```yaml
# GitHub Actions workflow
name: Deploy SSA Form-Assist

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: npm ci
      - run: npm run test
      - run: npm run test:e2e

  security-scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: npm audit
      - uses: snyk/actions/node@master

  deploy:
    needs: [test, security-scan]
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
```

---

# 9. Risk Mitigation

## 9.1 Technical Risks

| Risk | Impact | Mitigation |
|---|---|---|
| User forgets passphrase | **Critical** - Data loss | Implement optional recovery key export (encrypted), clear warnings during setup |
| Blue Book outdated | **High** - Legal inaccuracy | Automated update checker, version display, manual update trigger |
| LLM API cost spike | **Medium** - User expense | Token limit per generation, cost estimation UI, cached suggestions |
| Browser compatibility | **Medium** - Exclusion | Progressive enhancement, fallback UI for unsupported browsers |
| Cloud sync failure | **Low** - Inconvenience | Robust retry logic, local-first ensures no data loss, manual export option |

### 9.2 Legal/Compliance Risks

**Disclaimer Template** (must be prominent)

> IMPORTANT NOTICE:
> This application is NOT affiliated with the Social Security Administration.
> This tool provides guidance only and does not constitute legal or medical advice.
> You are solely responsible for the accuracy of information submitted to the SSA.
> All data is stored encrypted on your device and (optionally) your personal cloud.
> We do not have access to your information.

---

## 10. Success Metrics & Analytics

### 10.1 Key Performance Indicators

#### User Engagement

- Reports created per user
- Completion rate (wizard started → AI generated)
- Time saved (estimated vs. manual completion)

#### Technical Performance

- App load time (p50, p95)
- Offline usage percentage
- Cloud sync success rate
- Encryption operation latency

#### User Satisfaction

- Net Promoter Score (via optional survey)
- Feature usage distribution
- Support request volume/type

### 10.2 Privacy-Preserving Analytics

**Recommendation: Plausible Analytics** or **Fathom**

- No cookies, no personal data collection
- Aggregate metrics only
- GDPR/CCPA compliant by default

```typescript
// Track only non-sensitive events
analytics.track('report_created', { llm_provider: 'anonymized' });
analytics.track('ai_generation_completed', { time_taken_ms: 12000 });
// NEVER track: report content, user inputs, medical data
```

---

## 11. Total Estimated Effort

**Development:** 9 weeks (1 full-time developer) **Testing:** 2 weeks (concurrent with development)
**Documentation:** 1 week **Deployment & Refinement:** 1 week

**Total:** ~13 weeks to production-ready MVP

---

## Appendix A: Technology Alternatives Analysis

### Frontend Framework

| Option | Pros | Cons | Recommendation |
|---|---|---|---|
| React | Mature, large ecosystem, hiring pool | Bundle size, complexity | ✅ **Recommended** |
| Svelte | Smaller bundle, simpler syntax | Smaller ecosystem, fewer developers | Good alternative |
| Vue | Balanced approach, good DX | Less TypeScript maturity | Viable option |

### Encryption Library

| Option | Pros | Cons | Recommendation |
|---|---|---|---|
| Web Crypto API | Native, no dependencies, audited | Browser support (97%+) | ✅ **Recommended** |
| CryptoJS | Broad compatibility | Larger bundle, maintenance concerns | Fallback only |

### Cloud Storage

| Provider | Integration Complexity | User Base | Recommendation |
|---|---|---|---|
| OneDrive | Medium (MSAL.js) | High (Microsoft accounts) | ✅ **MVP** |
| Google Drive | Medium (Google APIs) | Very High | Post-MVP |
| Dropbox | Low (simple OAuth) | Medium | Post-MVP |

---

## Appendix B: Code Quality Standards

typescript

```
// TypeScript configuration
{
  "compilerOptions": {
    "strict": true,
    "noUncheckedIndexedAccess": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true
  }
}

// ESLint rules (critical)
{
  "rules": {
    "no-console": ["error", { "allow": ["warn", "error"] }],
    "no-eval": "error",
    "no-implied-eval": "error",
    "@typescript-eslint/no-explicit-any": "error"
  }
}

// Code review checklist
- [ ] All user inputs sanitized
- [ ] No sensitive data in console logs
- [ ] Error messages don't leak system info
- [ ] Accessibility tested with screen reader
- [ ] Encryption validated in tests
```

---

**Document Version:** 1.0

**Last Updated:** 2025-11-05

**Status:** Ready for Development