System Architecture Document

AI-Powered Personal Productivity System

2.1 Architecture Overview

High-Level Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                    CLIENT LAYER                                 │
├─────────────────────────────────────────────────────────────────┤
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐           │
│  │  React PWA   │  │   Service    │  │    Local     │           │
│  │  (Vite)      │  │   Worker     │  │    LLM       │           │
│  └──────────────┘  └──────────────┘  └──────────────┘           │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐           │
│  │  IndexedDB   │  │  Local State │  │   Claude     │           │
│  │  Storage     │  │  (Zustand)   │  │  Connector   │           │
│  └──────────────┘  └──────────────┘  └──────────────┘           │
└─────────────────────────────────────────────────────────────────┘
                          │
              ┌───────────┴───────────┐
              │  SYNC LAYER           │
              │  WebSocket/SSE        │
              └───────────┬───────────┘
                          │
┌─────────────────────────────────────────────────────────────────┐
│                    EDGE LAYER                                   │
├─────────────────────────────────────────────────────────────────┤
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐           │
│  │ Vercel Edge  │  │     CDN      │  │    Rate      │           │
│  │ Functions    │  │   Caching    │  │   Limiting   │           │
│  └──────────────┘  └──────────────┘  └──────────────┘           │
└─────────────────────────────────────────────────────────────────┘
                          │
┌─────────────────────────────────────────────────────────────────┐
│                   BACKEND LAYER                                 │
├─────────────────────────────────────────────────────────────────┤
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐           │
│  │ API Routes   │  │  Cron Jobs   │  │  WebSocket   │           │
│  │ (Next.js)    │  │  (Vercel)    │  │  Server      │           │
│  └──────────────┘ └──────────────┘ └──────────────┘   │
└──────────────────────────────────────────────────────────┘
               │
┌──────────────────────────────────────────────────────────┐
│               DATA LAYER               │
├──────────────────────────────────────────────────────────┤
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐   │
│ │  Supabase  │ │  Realtime  │ │  Storage  │   │
│ │ PostgreSQL │ │ Channels  │ │  Bucket  │   │
│  └──────────────┘ └──────────────┘ └──────────────┘   │
└──────────────────────────────────────────────────────────┘
```

Component Architecture

```
src/
├── components/
│   ├── core/
│   │   ├── Editor/          # Tiptap-based rich text editor
│   │   ├── TaskCard/        # Task display component
│   │   ├── TaskList/        # Virtual scrolling list
│   │   ├── Calendar/        # Calendar view component
│   │   └── KanbanBoard/     # Drag-drop kanban view
│   ├── ai/
│   │   ├── LLMProvider/     # Local LLM integration
│   │   ├── ClaudeConnector/ # Claude chat bridge
│   │   ├── TaskParser/      # NLP task parsing
│   │   └── Scheduler/       # AI scheduling engine
│   ├── sync/
│   │   ├── SyncManager/     # Conflict resolution
│   │   ├── OfflineQueue/    # Offline operation queue
│   │   └── RealtimeSync/    # WebSocket handler
│   └── shared/
│       ├── Layout/          # App shell and navigation
│       ├── ThemeProvider/   # Theme management
│       └── ErrorBoundary/   # Error handling
├── hooks/
│   ├── useLocalStorage/     # Persistent local state
│   ├── useOfflineSync/      # Offline sync logic
│   ├── useAIAssistant/      # AI integration hook
│   └── useRealtimeData/     # Live data subscription
├── lib/
│   ├── encryption/          # Crypto implementation
│   ├── database/            # Database abstraction
│   ├── ai/           # AI/LLM utilities
│   └── sync/           # Sync algorithms
├── api/
│   ├── tasks/          # Task CRUD operations
│   ├── sync/           # Sync endpoints
│   ├── auth/           # Authentication
│   └── cron/           # Scheduled jobs
└── workers/
    ├── service-worker.js    # PWA service worker
    ├── sync-worker.js       # Background sync
    └── ai-worker.js         # AI processing worker
```

2.2 Data Architecture

Database Schema

```sql
```

```sql
-- Core Tables

CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  encrypted_settings JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE tasks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  parent_id UUID REFERENCES tasks(id) ON DELETE CASCADE,
  title VARCHAR(500) NOT NULL,
  content JSONB, -- Encrypted rich text content
  status VARCHAR(50) DEFAULT 'pending',
  priority INTEGER DEFAULT 0,
  due_date TIMESTAMP WITH TIME ZONE,
  completed_at TIMESTAMP WITH TIME ZONE,

  -- Scheduling
  scheduled_for TIMESTAMP WITH TIME ZONE,
  duration_minutes INTEGER,

  -- Recurrence
  recurrence_pattern JSONB,
  recurrence_parent_id UUID REFERENCES tasks(id),

  -- AI Context
  ai_context JSONB, -- Encrypted AI suggestions/context
  embedding VECTOR(384), -- For semantic search

  -- Metadata
  tags TEXT[],
  dependencies UUID[],
  position REAL, -- For ordering

  -- Versioning
  version INTEGER DEFAULT 1,
  version_history JSONB[],

  -- Timestamps
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  deleted_at TIMESTAMP WITH TIME ZONE,

  -- Indexes
  INDEX idx_user_status (user_id, status),
  INDEX idx_due_date (due_date),
  INDEX idx_scheduled (scheduled_for),
  INDEX idx_position (user_id, position)
);

CREATE TABLE sync_log (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  device_id VARCHAR(255) NOT NULL,
  operation VARCHAR(50) NOT NULL,
  entity_type VARCHAR(50) NOT NULL,
  entity_id UUID NOT NULL,
  changes JSONB NOT NULL,
  vector_clock JSONB NOT NULL,
  synced_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  INDEX idx_sync_user_device (user_id, device_id, synced_at)
);

CREATE TABLE automation_rules (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  name VARCHAR(255) NOT NULL,
  trigger_type VARCHAR(50) NOT NULL,
  trigger_config JSONB NOT NULL,
  action_type VARCHAR(50) NOT NULL,
  action_config JSONB NOT NULL,
  is_active BOOLEAN DEFAULT true,
  last_triggered_at TIMESTAMP WITH TIME ZONE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  INDEX idx_automation_active (user_id, is_active)
);

-- Materialized Views for Performance

CREATE MATERIALIZED VIEW task_statistics AS
SELECT
```

```
  user_id,
  DATE_TRUNC('day', created_at) as date,
  COUNT(*) as total_tasks,
  COUNT(CASE WHEN status = 'completed' THEN 1 END) as completed_tasks,
  AVG(EXTRACT(EPOCH FROM (completed_at - created_at))/3600)::INTEGER as avg_completion_hours
FROM tasks
GROUP BY user_id, DATE_TRUNC('day', created_at);

CREATE INDEX idx_task_stats ON task_statistics(user_id, date);
```

Data Flow Patterns

Create Task Flow

```
1. User Input → Local LLM Parser
2. Parse Results → Task Object Creation
3. Encrypt Sensitive Fields → IndexedDB Save
4. Queue Sync Operation → Background Sync Worker
5. Sync to Supabase → Real-time Broadcast
6. Update Other Devices → Conflict Resolution
```

Sync Flow

```
1. Local Change Detection → Version Vector Update
2. Diff Calculation → Compressed Delta
3. Encrypted Upload → Supabase Transaction
4. Real-time Notification → Connected Clients
5. Conflict Detection → CRDT Merge
6. Local State Update → UI Refresh
```

2.3 Security Architecture

Encryption Model

```
javascript
```

```
// Client-Side Encryption Architecture
{
  "masterKey": {
    "derivation": "PBKDF2",
    "iterations": 1750000,
    "salt": "user-specific-salt",
    "length": 256
  },
  "dataEncryption": {
    "algorithm": "AES-256-GCM",
    "ivLength": 16,
    "tagLength": 16
  },
  "keyRotation": {
    "frequency": "90 days",
    "versioning": true,
    "backwardCompatible": true
  },
  "storageKeys": {
    "local": "encrypted-with-device-key",
    "cloud": "encrypted-with-master-key",
    "shared": "encrypted-with-derived-key"
  }
}
```

Authentication Flow

```
1. User Login Request
   ├── Email/Password → Supabase Auth
   ├── OAuth Provider → Provider Auth → Supabase
   └── Magic Link → Email Verification → Supabase

2. Token Management
   ├── Access Token (15 min) → API Requests
   ├── Refresh Token (7 days) → Token Renewal
   └── Session Cookie (httpOnly, secure, sameSite)

3. Authorization
   ├── JWT Claims Validation
   ├── Resource Ownership Check
   └── Rate Limiting per User
```

2.4 Technology Stack

```yaml
Frontend:
  - React 18.2 with TypeScript
  - Vite 5.0 build tool
  - Tiptap 2.0 editor
  - Zustand state management
  - Tailwind CSS 3.4
  - Shadcn/ui components

Backend:
  - Next.js 14 API routes
  - Vercel Edge Functions
  - Supabase PostgreSQL
  - Supabase Realtime

AI/ML:
  - Ollama local runtime
  - Mistral-7B model
  - Claude connector API

DevOps:
  - GitHub Actions CI/CD
  - Vercel deployment
  - Sentry monitoring
  - Playwright testing
```

Document Version: 1.0.0
Last Updated: January 2024
Next Review: February 2024