Life OS Project Documentation Bundle

📋 Document 1: Development Setup Checklist

Environment Setup (Complete First)

```bash
# Step-by-step setup commands
1. Download Node.js from https://nodejs.org (LTS version)
2. Install VS Code from https://code.visualstudio.com
3. Open terminal/command prompt and verify:
   node --version (should show v18+ or v20+)
   npm --version (should show 9+ or 10+)
4. Install Git: https://git-scm.com/downloads
5. Verify Git: git --version (should show version number)
```

Local Development Folder Setup

- [ ] **Create project folder** on your computer (e.g., Desktop/life-os)
- [ ] **Open folder in VS Code** (File → Open Folder)
- [ ] **Terminal setup** (Terminal → New Terminal in VS Code)
- [ ] **Git configuration** (first time only):

```bash
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Required Accounts to Create

- [ ] **GitHub Account** - Code backup and version control
  - Sign up: https://github.com
  - Purpose: Store all code safely
- [ ] **Vercel Account** - Hosting and deployment
  - Sign up: https://vercel.com (use GitHub login)
  - Purpose: Host your live application
- [ ] **Supabase Account** - Database and authentication
  - Sign up: https://supabase.com
  - Purpose: Store all your data securely
- [ ] **OpenAI Account** - AI features

- Sign up: https://platform.openai.com
- Purpose: Add intelligence to your system

VS Code Extensions (Install These)

- [ ] **ES7+ React/Redux/React-Native snippets**
- [ ] **Prettier - Code formatter**
- [ ] **Tailwind CSS IntelliSense**
- [ ] **TypeScript Importer**
- [ ] **GitLens**

First Session Prep Checklist

- [ ] Node.js installed and working
- [ ] VS Code installed with extensions
- [ ] Git installed and configured
- [ ] All accounts created (GitHub, Vercel, Supabase, OpenAI)
- [ ] Local project folder ready
- [ ] 2-3 hours of uninterrupted time
- [ ] Phone nearby for mobile testing
- [ ] Excited and ready to build! 🚀

Development Workflow Overview

Local Development (Your Computer):

- Write and test code locally first
- Fast feedback loop (no internet needed)
- Preview at `http://localhost:3000`

Version Control (GitHub):

- Backup all code changes
- Track history of modifications
- Enable collaboration and recovery

Live Deployment (Vercel):

- Automatic deployment from GitHub
- Live website for mobile testing
- Share progress with others

📋 Document 2: Session Planning Template

Pre-Session Checklist

- [ ] Review previous session notes
- [ ] Identify today's development goal
- [ ] Ensure development environment is ready
- [ ] Have Claude conversation open
- [ ] Set realistic time expectations

Development Session Template

Date: [Insert Date] **Duration:** [Planned time] **Goal:** [What we're building today]

Starting Point:

- Current system status: [Working features]
- Last completed: [Previous achievement]
- Blockers from last session: [Any issues]

Development Session Template

Date: [Insert Date] **Duration:** [Planned time] **Goal:** [What we're building today]

Starting Point:

- Local environment status: [VS Code open, terminal ready]
- Current branch: [usually 'main']
- Last deployment status: [live site working]
- Today's focus: [specific feature/module]

Today's Objectives:

1. [Primary goal - be specific]
2. [Secondary goal if time permits]
3. [Stretch goal if ahead of schedule]

Local Development Progress:

- [Features coded locally]
- [Local testing results]
- [Issues discovered during local development]

Deployment Progress:

- [ ] Code committed to Git
- [ ] Pushed to GitHub
- [ ] Vercel deployment successful
- [ ] Live site tested on desktop
- [ ] Mobile testing completed

Testing Results:

- [ ] Local functionality works (`localhost:3000`)
- [ ] GitHub repository updated
- [ ] Live site functionality works
- [ ] Mobile responsiveness verified
- [ ] No console errors on any platform

Git Commands Used:

```bash
git add .
git commit -m "Add [feature description]"
git push origin main
```

Next Session Prep:

- [ ] All changes committed and pushed
- [ ] Live site reflects latest changes
- [ ] No merge conflicts or issues
- [ ] Next priorities identified
- [ ] Local environment ready for next session

Notes and Learnings: [Key insights, challenges overcome, Claude explanations that were helpful]

📋 Document 3: Technical Architecture Reference

System Overview

```
Life OS Architecture:

Frontend (What You See)
├── Next.js 14+ (React framework)
├── TypeScript (Type safety)
├── Tailwind CSS (Styling)
└── PWA (Mobile app features)

Backend (Data Processing)
├── Next.js API Routes (Server functions)
├── Supabase Database (PostgreSQL)
├── Supabase Auth (User accounts)
└── Real-time Subscriptions

External Services
├── OpenAI API (AI intelligence)
├── Plaid API (Banking data - Phase 3)
├── Court APIs (Legal data - Phase 2)
└── Health APIs (Apple Health - Phase 4)
```

Database Schema (Key Tables)

Tasks Table

```sql
- id (unique identifier)
- title (task name)
- description (details)
- status (inbox, next_action, in_progress, done, etc.)
- priority (P1, P2, P3, deadline, etc.)
- do_date (when you plan to work on it)
- due_date (when it must be finished)
- created_at (when task was created)
- updated_at (when last modified)
```

Knowledge Items Table

```sql
- id (unique identifier)
- title (note/article title)
- content (the actual note content)
- type (note, article, document, voice_memo)
- categories (Legal Research, Personal, etc.)
- tags (flexible labeling)
- source_url (if from web)
- created_at (when saved)
- access_count (how often viewed)
```

Future Tables (Later Phases)

- cases (legal case management)
- projects (goal and project tracking)
- financial_accounts (net worth tracking)
- health_metrics (body and fitness data)

Key File Structure

```
life-os/ (local project folder)
├── .git/ (version control - created automatically)
├── .next/ (build files - auto-generated)
├── node_modules/ (dependencies - auto-installed)
├── pages/
│   ├── index.tsx (main dashboard)
│   ├── tasks/ (task management pages)
│   ├── knowledge/ (knowledge management)
│   └── api/ (backend functions)
├── components/
│   ├── ui/ (reusable components)
│   ├── tasks/ (task-specific components)
│   └── knowledge/ (knowledge components)
├── lib/
│   ├── supabase.ts (database connection)
│   ├── openai.ts (AI integration)
│   └── utils.ts (helper functions)
├── styles/
│   └── globals.css (styling)
├── .env.local (environment variables - keep secret!)
├── package.json (project configuration)
└── README.md (project documentation)
```

Development Environment URLs

- **Local Development:** `http://localhost:3000`
- **GitHub Repository:** `https://github.com/yourusername/life-os`
- **Live Website:** `https://your-life-os.vercel.app`
- **Supabase Dashboard:** `https://app.supabase.com/project/your-project-id`

📋 Document 4: Troubleshooting Guide

Common Issues and Solutions

Common Issues and Solutions

Local Development Environment

Problem: `npm` commands fail **Solution:** Run terminal as administrator (Windows) or use sudo (Mac)

Problem: `localhost:3000` not loading **Solution:** Check if development server is running (`npm run dev`)

Problem: Changes not showing in browser **Solution:** Hard refresh (Ctrl+F5 or Cmd+Shift+R)

Git and Version Control

Problem: Git not found **Solution:** Install Git from https://git-scm.com, restart VS Code

Problem: "Please tell me who you are" error **Solution:** Configure Git with your name and email:

```bash
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

Problem: Can't push to GitHub **Solution:** Check GitHub authentication, may need personal access token

Deployment Issues

Problem: Vercel deployment fails **Solution:** Check build logs in Vercel dashboard, verify environment variables

Problem: Live site shows old version **Solution:** Check if latest commit pushed to GitHub, force refresh browser

Problem: Environment variables not working on live site **Solution:** Add variables in Vercel dashboard, redeploy

Database and API Issues

Problem: Supabase connection fails locally **Solution:** Check `.env.local` file exists with correct keys

Problem: Database works locally but not on live site **Solution:** Verify environment variables set in Vercel dashboard

Problem: API routes returning 404 **Solution:** Check file structure in `pages/api/` folder, restart dev server

General Development

Problem: Code not working as expected **Solution:** Check browser console, use Chrome DevTools, ask Claude for debugging help

Problem: Stuck on implementation **Solution:** Break down into smaller steps, ask Claude for step-by-step guidance

Emergency Recovery

If something breaks badly:

1. **Don't panic** - everything is backed up in Git
2. **Check what changed:** `git log --oneline` (see recent commits)
3. **Revert if needed:** `git reset --hard HEAD~1` (go back one commit)
4. **Start fresh:** Delete local folder, clone from GitHub again
5. **Ask Claude for help** with specific error messages

Daily Git Workflow

```bash
# Start of session - make sure you're up to date
git pull origin main

# After making changes
git add .                    # Stage all changes
git commit -m "Add task creation"  # Commit with descriptive message
git push origin main         # Push to GitHub (triggers deployment)

# Check status anytime
git status                   # See what's changed
```

📋 Document 5: Feature Request & Bug Tracking

New Feature Requests

Template for new features:

```
Feature: [Name]
Priority: High/Medium/Low
Module: [Which system it belongs to]
Description: [What it should do]
User Story: "As a user, I want... so that..."
Acceptance Criteria:
- [ ] Criteria 1
- [ ] Criteria 2
- [ ] Criteria 3
```

Bug Report Template

```
Bug: [Short description]
Severity: Critical/High/Medium/Low
Steps to Reproduce:
1. Step 1
2. Step 2
3. Step 3
Expected Result: [What should happen]
Actual Result: [What actually happens]
Browser/Device: [Chrome, Safari, iPhone, etc.]
Screenshot: [If helpful]
```

Current Feature Backlog

Phase 1 Priority Features:

- [ ] Task quick capture with voice input
- [ ] Smart task prioritization with AI
- [ ] Knowledge article web clipper
- [ ] Cross-reference detection between tasks and notes
- [ ] Mobile-optimized interface
- [ ] Daily review dashboard

Future Enhancement Ideas:

- [ ] Calendar integration for time blocking
- [ ] Email-to-task creation
- [ ] Recurring task templates
- [ ] Knowledge item summarization
- [ ] Habit tracking integration
- [ ] Voice control interface

📋 Document 6: Code Snippets & References

Useful Code Patterns

Creating a New Page

typescript

```typescript
// pages/example.tsx
import { useState, useEffect } from 'react';
import { supabase } from '../lib/supabase';

export default function ExamplePage() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    const { data, error } = await supabase
      .from('table_name')
      .select('*');
    if (data) setData(data);
  };

  return (
    <div className="container mx-auto p-4">
      <h1 className="text-2xl font-bold mb-4">Example Page</h1>
      {/* Your content here */}
    </div>
  );
}
```

Creating a Reusable Component

typescript

```typescript
// components/ui/Button.tsx
interface ButtonProps {
  children: React.ReactNode;
  onClick?: () => void;
  variant?: 'primary' | 'secondary';
  disabled?: boolean;
}

export function Button({ children, onClick, variant = 'primary', disabled }: ButtonProps) {
  const baseClasses = "px-4 py-2 rounded-md font-medium";
  const variantClasses = variant === 'primary'
    ? "bg-blue-600 text-white hover:bg-blue-700"
    : "bg-gray-200 text-gray-800 hover:bg-gray-300";

  return (
    <button
      className={`${baseClasses} ${variantClasses} ${disabled ? 'opacity-50' : ''}`}
      onClick={onClick}
      disabled={disabled}
    >
      {children}
    </button>
  );
}
```

Database Operations

typescript

```typescript
// lib/database.ts
export const taskOperations = {
 // Create new task
 create: async (task: Partial<Task>) => {
  const { data, error } = await supabase
   .from('tasks')
   .insert([task])
   .select();
  return { data, error };
 },

 // Get all tasks for user
 getAll: async (userId: string) => {
  const { data, error } = await supabase
   .from('tasks')
   .select('*')
   .eq('user_id', userId)
   .order('created_at', { ascending: false });
  return { data, error };
 },

 // Update task
 update: async (id: string, updates: Partial<Task>) => {
  const { data, error } = await supabase
   .from('tasks')
   .update(updates)
   .eq('id', id)
   .select();
  return { data, error };
 }
};
```

Environment Variables Template

bash

```bash
# .env.local (create this file in your project root)
NEXT_PUBLIC_SUPABASE_URL=your_supabase_url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key
OPENAI_API_KEY=your_openai_api_key
```

📋 Document 7: Phase Completion Checklists

Phase 1 Completion Checklist (Months 1-3)

Task Management Module ✅

- [ ] Task creation with all fields (title, description, priority, dates)
- [ ] Task status management (inbox → next action → in progress → done)
- [ ] Priority system implementation (P1, P2, P3, deadline, etc.)
- [ ] Today view showing urgent/important tasks
- [ ] Search and filter functionality
- [ ] Mobile-responsive design
- [ ] Voice input for quick capture
- [ ] AI insights for task prioritization

Knowledge Management Module ✅

- [ ] Note creation and editing with rich text
- [ ] Category and tag organization system
- [ ] Full-text search across all content
- [ ] Web article saving (manual upload for MVP)
- [ ] Cross-reference suggestions between notes
- [ ] Recent items and quick access
- [ ] Mobile note-taking capability
- [ ] AI auto-categorization

Review System Module ✅

- [ ] Daily review template (morning/evening)
- [ ] Weekly review dashboard
- [ ] Progress tracking across tasks and knowledge
- [ ] Pattern recognition and insights
- [ ] Goal setting and tracking interface
- [ ] Analytics dashboard showing productivity trends

Integration & Polish ✅

- [ ] Cross-module data connections (tasks ↔ knowledge)
- [ ] Unified search across all modules
- [ ] Mobile PWA installation capability
- [ ] Data export functionality

- [ ] User onboarding flow
- [ ] Performance optimization
- [ ] Security audit complete

Phase 1 Success Metrics

- [ ] Daily active use for 30+ consecutive days
- [ ] 90%+ task capture rate (using system vs. external tools)
- [ ] Sub-10-second information retrieval
- [ ] Measurable productivity improvement (tracked through reviews)
- [ ] System feels indispensable
- [ ] Ready to build Phase 2 modules

📋 Document 8: Communication Templates

Asking Claude for Help - Best Practices

When Starting a New Session

```
"Hi Claude! I'm working on my Life OS project. Here's where I left off:
- Current status: [What's working]
- Last completed: [Recent achievement]
- Today's goal: [What I want to build]
- Specific question: [Exact help needed]

Please help me [specific request] and explain your approach."
```

When Stuck on Implementation

```
"I'm trying to implement [specific feature] but running into [specific issue].
Here's my current code: [paste code]
Here's the error I'm getting: [paste error]
Here's what I expected to happen: [description]
Can you help me debug this step by step?"
```

When Planning New Features

"I want to add [feature description] to my Life OS.
This should [explain user story and benefits].
How would you recommend implementing this?
Please consider:
- Database changes needed
- UI/UX approach
- Integration with existing modules
- Mobile optimization"

When Requesting Code Reviews

"Can you review this code I wrote and suggest improvements?
[paste code]
I'm particularly concerned about:
- Performance
- Security
- Code organization
- Best practices
Please provide specific suggestions for improvement."

Project Status Update Template

Life OS Development Update - [Date]

Progress This Week:
✅ Completed: [List achievements]
🚧 In Progress: [Current work]
🎯 Next Up: [Upcoming priorities]

Metrics:
- Features completed: X/Y
- Time invested: X hours
- System usage: X days active
- Productivity impact: [qualitative assessment]

Challenges:
- [Any blockers or difficulties]
- [Learning curve items]

Wins:
- [Exciting breakthroughs]
- [Moments of satisfaction]

Next Week Goals:
1. [Primary objective]
2. [Secondary objective]
3. [Stretch goal]

📋 Document 9: Decision Log & Architecture Notes

Major Technical Decisions

Decision: Next.js over Create React App

Date: [Initial] **Reasoning:** Full-stack capability, better performance, built-in API routes **Alternatives Considered:** Create React App, Vite, Remix **Impact:** Enables single codebase for frontend and backend

Decision: Supabase over Firebase

Date: [Initial]
Reasoning: PostgreSQL (more powerful), better pricing, open source **Alternatives Considered:** Firebase, AWS Amplify, custom backend **Impact:** More flexible data modeling, better development experience

Decision: TypeScript over JavaScript

Date: [Initial] **Reasoning:** Better error catching, improved development experience **Alternatives Considered:** Plain JavaScript **Impact:** Steeper learning curve but much better maintainability

Future Decisions to Make

- [ ] State management solution (Context API vs. Zustand vs. Redux)
- [ ] Testing framework (Jest, Cypress, Playwright)
- [ ] Deployment strategy for production scale
- [ ] Monitoring and analytics implementation
- [ ] Performance optimization approach

Architecture Evolution Notes

Current: Monolithic Next.js application **Future Considerations:**

- Microservices for different modules
- Separate AI processing service
- CDN for file storage
- Caching layer for performance

📋 Document 10: Learning Resources & References

Essential Learning Resources

Next.js Documentation

- Official Docs: https://nextjs.org/docs
- Tutorial: https://nextjs.org/learn
- Examples: https://github.com/vercel/next.js/tree/canary/examples

Supabase Resources

- Documentation: https://supabase.com/docs
- JavaScript Client: https://supabase.com/docs/reference/javascript
- Auth Guide: https://supabase.com/docs/guides/auth

TypeScript Learning

- Handbook: https://www.typescriptlang.org/docs/

- React + TypeScript: https://react-typescript-cheatsheet.netlify.app/

Tailwind CSS

- Documentation: https://tailwindcss.com/docs
- Component Examples: https://tailwindui.com/components

Code Quality Tools

- **Prettier:** Code formatting
- **ESLint:** Code linting and best practices
- **TypeScript:** Type checking
- **Chrome DevTools:** Debugging and performance

Deployment Resources

- **Vercel Docs:** https://vercel.com/docs
- **Environment Variables:** https://vercel.com/docs/concepts/projects/environment-variables
- **Custom Domains:** https://vercel.com/docs/concepts/projects/custom-domains

AI Integration

- **OpenAI API Docs:** https://platform.openai.com/docs
- **Prompt Engineering:** https://platform.openai.com/docs/guides/prompt-engineering
- **Rate Limits:** https://platform.openai.com/docs/guides/rate-limits

Troubleshooting Resources

- **Stack Overflow:** For specific coding questions
- **GitHub Issues:** For library-specific problems
- **Discord Communities:** Real-time help from developers
- **Claude:** Always available for Life OS specific questions!