Payload CMS v3 Integration Progress

This document tracks the progress of integrating Payload CMS v3 with SQLite into the existing Next.js 15.4.5 application.

Project Overview

- **Project Path**: `/home/ubuntu/ptnextjs`
- **Next.js Version**: 15.4.5
- **Target CMS**: Payload CMS v3
- **Database**: SQLite
- **Integration Goal**: Headless CMS for content management with static site performance

Integration Steps Progress

✅ Step 1: Set Up Payload CMS v3 with SQLite

Status: ✅ COMPLETED
Completion Date: 2025-01-28

Tasks Completed:
- [x] Created `/cms` folder at project root
- [x] Initialized Node.js app inside `/cms` with `npm init -y`
- [x] Installed required dependencies: `payload@3.49.1`, `@payloadcms/db-sqlite@3.49.1`, `express@5.1.0`
- [x] Installed development dependencies: `typescript`, `@types/node`, `ts-node`
- [x] Created `payload.config.ts` with minimal SQLite configuration
- [x] Configured Payload to use SQLite with filename `./payload.db`
- [x] Created server setup file (`server.mjs`) for running Payload
- [x] Set up TypeScript configuration (`tsconfig.json`)
- [x] Created environment configuration (`.env.example`)
- [x] Verified all dependencies are properly installed

Files Created:
- `/cms/package.json` - Node.js package configuration
- `/cms/payload.config.ts` - Payload CMS configuration with SQLite adapter
- `/cms/server.mjs` - Express server setup for Payload admin
- `/cms/tsconfig.json` - TypeScript configuration
- `/cms/.env.example` - Environment variables template
- `/cms/.gitignore` - Git ignore for CMS-specific files

Technical Implementation:
- **Database**: SQLite adapter configured to use `./payload.db`
- **Admin Panel**: Accessible at `http://localhost:3001/admin`
- **API Endpoint**: Available at `http://localhost:3001/api`
- **Authentication**: Basic user collection with email/password auth
- **TypeScript**: Full TypeScript support with proper type generation

Next Steps Ready:
- Basic user collection configured for admin access
- SQLite database adapter properly configured
- Server setup ready for development and testing
- Ready to add custom collections in Step 2

⏳ Step 2: Define Content Collections

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Create Blog Posts collection
- [ ] Create Partners collection
- [ ] Create Products collection
- [ ] Define field schemas for each collection
- [ ] Set up proper relationships between collections

⏳ Step 3: Set Up Payload Admin Interface

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Configure admin panel access
- [ ] Set up authentication
- [ ] Create admin user account
- [ ] Test CRUD operations in admin interface

⏳ Step 4: Create Data Migration Scripts

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Create migration script for existing blog data
- [ ] Create migration script for existing partner data
- [ ] Create migration script for existing product data
- [ ] Test data migration integrity
- [ ] Backup existing static data

⏳ Step 5: Integrate Payload with Next.js API Routes

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Create API routes for fetching CMS data
- [ ] Replace static data imports with CMS API calls
- [ ] Maintain existing API structure for compatibility
- [ ] Test data fetching in development environment

⏳ Step 6: Update Next.js Components

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Update blog components to use CMS data
- [ ] Update partner components to use CMS data
- [ ] Update product components to use CMS data
- [ ] Maintain static generation compatibility
- [ ] Test all pages and components

⏳ Step 7: Testing and Final Integration

Status: ⏳ PENDING
Target Date: TBD

Planned Tasks:
- [ ] Test complete integration workflow
- [ ] Verify static generation still works
- [ ] Performance testing and optimization
- [ ] Create documentation for content management
- [ ] Deploy and test in production environment

Technical Notes

Current Project Structure

```
/home/ubuntu/ptnextjs/
├── app/                    # Next.js app directory
├── components/             # React components
├── lib/                   # Utility functions and data
├── data/                  # Static data files (to be migrated)
├── cms/                   # 🆕 Payload CMS directory
│   ├── payload.config.ts  # CMS configuration
│   ├── package.json       # CMS dependencies
│   └── payload.db         # SQLite database (generated)
└── [other Next.js files]
```

Dependencies Added

- `payload` - Core Payload CMS
- `@payloadcms/db-sqlite` - SQLite adapter

- `express` - Server framework for Payload

Configuration Highlights

- SQLite database: `./payload.db`
- Admin panel: Will be accessible at `/admin`
- API endpoint: Will be at `/api`
- TypeScript configuration included

Next Steps

1. **Immediate**: Begin Step 2 - Define Content Collections
2. **Priority**: Focus on maintaining existing static site performance
3. **Testing**: Ensure no breaking changes to current functionality

---

Last Updated: 2025-01-28
Next Review: After Step 2 completion