# Day Lewis Clinical Pharmacist Training Platform - Project Overview

## Table of Contents

1. [Project Summary](#X54beaf6b7c173cfe2e76ed88e7f376987be7b82)
2. [Technical Architecture](#X2149b8bfa40d07ace7a1f59777abf926e2a055e)
3. [Core Features & Requirements](#X7d064f30e50dd2cf65ac2f194eb33b4af8701aa)
4. [Compliance & Standards](#X25ee38ea3c5f5123b5e509209aba4b1a74b34f3)
5. [Database Schema](#X6133caf47c5ca38d4ca9b34847ec956a314f651)
6. [API Endpoints](#X59bac15bb24905ac2bacdc7b1b4b35140d82127)
7. [Development Roadmap](#X2a72d9f254dd7eff624d70d0c0be09366270d6f)
8. [File Structure](#X4457fbc5a1bcf8bfe1d068cb4f9877e6a4af203)
9. [Deployment Strategy](#Xb0d5d70d9b95ced38cbc8873cf3082346dea51e)
10. [Migration Plan](#Xf7bc1e37d216a2da34d408fca2bfaf2bfc9db9b)
11. [Testing Strategy](#X021ba6005f0d214bb58a6ec078bf94acd80af97)
12. [Security Considerations](#X16f1648241fa3d6d73716a0819a4a70c6aeee0e)
13. [Monitoring & Maintenance](#Xdda549ce5449ac3e44e4132517af088cf2f3913)
14. [Questions & Decisions Pending](#X1a5cd4b2750d320f29e24b879849226039423e2)
15. [Reference Commands](#Xc22ca4f5a6926a26ba611ec498b4a04788f2aab)
16. [Document History](#X1857f575f971821bdaf752ff864dd2342686374)

## 1. PROJECT SUMMARY

### Business Context

Day Lewis, a UK private pharmacy chain, is partnering with experienced pharmacists and doctors to create a comprehensive digital training platform for clinical pharmacist students within their organization.

### Key Stakeholders

* **Day Lewis Pharmacy Chain**: Primary client and platform owner
* **Content Creators**: Experienced pharmacist/doctor team providing educational materials
* **Clinical Pharmacist Students**: Primary end users accessing training modules
* **Clinical Pharmacist Leaders**: Secondary users monitoring student progress and providing guidance

### Core Value Proposition

* Centralized, scalable clinical pharmacy education platform
* Role-based access with leader oversight capabilities
* Interactive learning through quizzes and case studies
* Progress tracking and assessment tools
* Monetization through subscription-based premium content

### Timeline

* **MVP Target**: September 17th, 2024
* **Phase 2 (Payments)**: Q4 2024
* **Full Platform Launch**: Q1 2025

## 2. TECHNICAL ARCHITECTURE

### Tech Stack Decisions

**Frontend:** - **Framework**: Next.js 14 (App Router) - **Styling**: Tailwind CSS (consistent with existing HTML styling) - **UI Components**: Headless UI or Radix UI - **State Management**: React Context + useReducer (minimal state with Supabase real-time) - **Form Handling**: React Hook Form with Zod validation

**Backend & Database:** - **Platform**: Supabase (PostgreSQL + Auth + Real-time + Storage) - **Database**: PostgreSQL 15+ (managed by Supabase) - **Authentication**: Supabase Auth (built-in email/password + domain validation) - **Real-time**: Supabase real-time subscriptions - **API**: Auto-generated REST API + TypeScript types - **File Storage**: Supabase Storage (S3-compatible)

**Payment Processing:** - **Provider**: Stripe (UK-configured) - **Model**: Annual subscriptions with pro-rating - **Webhooks**: Stripe webhooks for subscription lifecycle - **Integration**: Custom Stripe + Supabase webhook handling

**Cloud Hosting:** - **Frontend**: Vercel (seamless Next.js deployment) - **Database/Backend**: Supabase (global CDN, automatic scaling) - **CDN**: Vercel Edge Network + Supabase CDN - **SSL**: Automatic HTTPS with both platforms

**Testing Frameworks:** - **Unit Tests**: Jest + React Testing Library - **Integration Tests**: Supabase test client for database testing - **E2E Tests**: Playwright with Supabase test environment - **Performance**: Lighthouse CI

### System Architecture

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐  
│ Vercel │◄──►│ Next.js App │◄──►│ Supabase │  
│ (Frontend) │ │ (Client-side) │ │ (Backend/DB/Auth)│  
└─────────────────┘ └──────────────────┘ └─────────────────┘  
 │ │ │  
 │ │ ├─ PostgreSQL  
 │ │ ├─ Authentication  
 │ │ ├─ Real-time API  
 │ │ ├─ File Storage  
 │ │ └─ Edge Functions  
 │ │  
 │ ▼  
 │ ┌──────────────────┐  
 └──────────────►│ Stripe │  
 │ (Payments) │  
 └──────────────────┘

**Architecture Approach:** - **Supabase-centric** (all backend functionality in one platform) - **Serverless by default** (automatic scaling, no infrastructure management) - **Real-time first** (live progress updates, instant quiz results) - **Edge-optimized** (global CDN distribution for fast loading)

**Security Considerations:** - Domain-restricted authentication (@daylewis.net only) via Supabase Auth - Row Level Security (RLS) for role-based access control - Built-in API rate limiting and DDoS protection - Automatic data encryption at rest and in transit - Supabase security best practices (SOC 2 Type 2 compliant)

## 3. CORE FEATURES & REQUIREMENTS

### Phase 1 (MVP - September 17th)

**Authentication & User Management:** - [ ] Email/password registration restricted to @daylewis.net domains - [ ] User role assignment (Student/Leader) - [ ] Profile management and password reset - [ ] Session management with secure logout

**Content Management:** - [ ] Display training modules (converted from existing HTML) - [ ] Module navigation and progress tracking - [ ] Responsive design for mobile/tablet access - [ ] Search and filtering capabilities

**Assessment System:** - [ ] Interactive quiz functionality with multiple question types - [ ] Case study presentation and submission - [ ] Answer storage with timestamp tracking - [ ] Immediate feedback for quizzes

**Leader Dashboard:** - [ ] View all student submissions by module - [ ] Student progress overview - [ ] Individual student performance tracking - [ ] Basic reporting and export functionality

**Progress Tracking:** - [ ] Module completion status - [ ] Quiz scores and attempt history - [ ] Learning path visualization - [ ] Time spent tracking

### Phase 2 (Post-MVP - Q4 2024)

**Payment Integration:** - [ ] Stripe subscription setup (annual billing) - [ ] Paywall implementation for premium modules - [ ] Subscription management dashboard - [ ] Pro-rating for mid-cycle changes

**Advanced Features:** - [ ] Automated grading with detailed feedback - [ ] Certificate generation upon completion - [ ] Leader-to-student feedback system - [ ] Advanced analytics dashboard

**Content Enhancement:** - [ ] Rich text editor for content updates - [ ] Version control for training materials - [ ] Bulk content operations - [ ] Content approval workflow

### Future Considerations

**Advanced Learning Features:** - [ ] Video content integration with progress tracking - [ ] Interactive clinical simulations - [ ] Downloadable PDFs and resources - [ ] Discussion forums and peer interaction

**Analytics & Insights:** - [ ] Learning analytics and patterns - [ ] Predictive performance modeling - [ ] Detailed reporting suite - [ ] Integration with pharmacy management systems

## 4. COMPLIANCE & STANDARDS

### Healthcare Education Standards

* **GPhC Standards**: Alignment with General Pharmaceutical Council requirements
* **NICE Guidelines**: Integration of NICE clinical guidelines in content
* **CPD Requirements**: Continuing Professional Development tracking
* **Clinical Governance**: Audit trails for educational compliance

### Data Protection & Privacy

* **UK GDPR Compliance**: Data processing lawful basis documentation
* **Data Retention**: Automated deletion policies for inactive accounts
* **Consent Management**: Granular permissions for data usage
* **Right to Erasure**: Automated data deletion workflows

### Accessibility Standards

* **WCAG 2.1 AA**: Web Content Accessibility Guidelines compliance
* **Screen Reader Support**: Semantic HTML and ARIA labels
* **Keyboard Navigation**: Full keyboard accessibility
* **Color Contrast**: High contrast mode support

### Security Standards

* **OWASP Top 10**: Regular security assessment against OWASP guidelines
* **ISO 27001**: Information security management alignment
* **Penetration Testing**: Quarterly security assessments
* **Incident Response**: Documented breach response procedures

## 5. DATABASE SCHEMA

### Core Tables

-- Supabase Auth handles users table automatically  
-- We create a profiles table to extend user data  
CREATE TABLE profiles (  
 id UUID REFERENCES auth.users(id) PRIMARY KEY,  
 email VARCHAR(255) NOT NULL,  
 role user\_role NOT NULL DEFAULT 'student',  
 first\_name VARCHAR(100) NOT NULL,  
 last\_name VARCHAR(100) NOT NULL,  
 is\_active BOOLEAN DEFAULT true,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Enable RLS (Row Level Security)  
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;  
  
-- Modules table  
CREATE TABLE modules (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 title VARCHAR(255) NOT NULL,  
 description TEXT,  
 content\_url VARCHAR(500),  
 is\_premium BOOLEAN DEFAULT false,  
 order\_index INTEGER,  
 estimated\_duration INTEGER, -- minutes  
 category VARCHAR(100),  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Quizzes table  
CREATE TABLE quizzes (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 module\_id UUID REFERENCES modules(id) ON DELETE CASCADE,  
 title VARCHAR(255) NOT NULL,  
 description TEXT,  
 passing\_score INTEGER DEFAULT 70,  
 time\_limit INTEGER, -- minutes  
 max\_attempts INTEGER DEFAULT 3,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Questions table  
CREATE TABLE questions (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 quiz\_id UUID REFERENCES quizzes(id) ON DELETE CASCADE,  
 question\_text TEXT NOT NULL,  
 question\_type question\_type NOT NULL DEFAULT 'multiple\_choice',  
 options JSONB, -- For multiple choice options  
 correct\_answer TEXT NOT NULL,  
 explanation TEXT,  
 points INTEGER DEFAULT 1,  
 order\_index INTEGER,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Quiz Attempts table  
CREATE TABLE quiz\_attempts (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES users(id) ON DELETE CASCADE,  
 quiz\_id UUID REFERENCES quizzes(id) ON DELETE CASCADE,  
 answers JSONB NOT NULL,  
 score INTEGER,  
 started\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 completed\_at TIMESTAMP,  
 time\_spent INTEGER -- seconds  
);  
  
-- Module Progress table  
CREATE TABLE module\_progress (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES users(id) ON DELETE CASCADE,  
 module\_id UUID REFERENCES modules(id) ON DELETE CASCADE,  
 status progress\_status DEFAULT 'not\_started',  
 started\_at TIMESTAMP,  
 completed\_at TIMESTAMP,  
 time\_spent INTEGER DEFAULT 0, -- seconds  
 UNIQUE(user\_id, module\_id)  
);  
  
-- Subscriptions table  
CREATE TABLE subscriptions (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES users(id) ON DELETE CASCADE,  
 stripe\_subscription\_id VARCHAR(255) UNIQUE,  
 status subscription\_status DEFAULT 'inactive',  
 current\_period\_start TIMESTAMP,  
 current\_period\_end TIMESTAMP,  
 cancel\_at\_period\_end BOOLEAN DEFAULT false,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

### Enums

CREATE TYPE user\_role AS ENUM ('student', 'leader', 'admin');  
CREATE TYPE question\_type AS ENUM ('multiple\_choice', 'true\_false', 'short\_answer', 'case\_study');  
CREATE TYPE progress\_status AS ENUM ('not\_started', 'in\_progress', 'completed');  
CREATE TYPE subscription\_status AS ENUM ('active', 'inactive', 'past\_due', 'canceled', 'unpaid');

## 6. API ENDPOINTS

### Authentication (Supabase Auth)

// Built-in Supabase Auth methods (no custom endpoints needed)  
supabase.auth.signUp() // Register with email validation  
supabase.auth.signInWithPassword() // User login  
supabase.auth.signOut() // User logout  
supabase.auth.resetPasswordForEmail() // Password reset  
supabase.auth.getUser() // Get current user  
  
// Custom validation for @daylewis.net domain in client-side

### Database Operations (Supabase Client)

// User Management (with RLS)  
const { data: users } = await supabase  
 .from('profiles')  
 .select('\*') // Leaders can see all, students only themselves  
  
// Module Management  
const { data: modules } = await supabase  
 .from('modules')  
 .select('\*')  
 .eq('is\_premium', false) // Filter based on subscription status  
  
// Quiz Operations  
const { data: quiz } = await supabase  
 .from('quizzes')  
 .select(`  
 \*,  
 questions(\*)  
 `)  
 .eq('module\_id', moduleId)  
  
// Submit Quiz Attempt  
const { data: attempt } = await supabase  
 .from('quiz\_attempts')  
 .insert({  
 user\_id: user.id,  
 quiz\_id: quizId,  
 answers: answers,  
 score: calculatedScore  
 })  
  
// Real-time Progress Updates  
supabase  
 .channel('progress')  
 .on('postgres\_changes', {  
 event: 'INSERT',  
 schema: 'public',  
 table: 'quiz\_attempts'  
 }, handleNewSubmission)  
 .subscribe()

### Payment Processing

POST /api/subscriptions/create // Create Stripe subscription  
POST /api/subscriptions/cancel // Cancel subscription  
GET /api/subscriptions/status // Get subscription status  
POST /api/webhooks/stripe // Stripe webhook handler

## 7. DEVELOPMENT ROADMAP

### Sprint 1 (August 26 - September 8): Foundation

**Week 1:** - [ ] Set up Supabase project and configure database schema - [ ] Initialize Next.js project with Supabase integration - [ ] Configure Supabase Auth with @daylewis.net domain validation - [ ] Set up Row Level Security (RLS) policies for user roles - [ ] Create basic user registration and login flows

**Week 2:** - [ ] Build basic UI components and layout structure - [ ] Implement role-based access with RLS policies - [ ] Create module listing with Supabase real-time subscriptions - [ ] Set up file storage with Supabase Storage - [ ] Convert first 5 HTML modules to new system structure

### Sprint 2 (September 9 - September 17): MVP Completion

**Week 1:** - [ ] Complete quiz functionality with multiple question types - [ ] Implement answer submission and storage - [ ] Build leader dashboard for student overview - [ ] Add progress tracking for module completion - [ ] Create basic reporting features

**Week 2:** - [ ] Implement remaining HTML module conversions - [ ] Add comprehensive error handling and validation - [ ] Perform thorough testing (unit, integration, E2E) - [ ] Set up production deployment pipeline - [ ] Deploy MVP to production environment

### Sprint 3 (September 18 - October 1): Payment Integration

* ☐ Integrate Stripe for UK-based subscriptions
* ☐ Implement paywall logic for premium modules
* ☐ Create subscription management dashboard
* ☐ Add webhook handling for payment events
* ☐ Test payment flows thoroughly

### Sprint 4 (October 2 - October 15): Enhancement & Polish

* ☐ Advanced analytics and reporting features
* ☐ Automated grading system implementation
* ☐ Performance optimization and caching
* ☐ Security audit and penetration testing
* ☐ User acceptance testing and feedback incorporation

## 8. FILE STRUCTURE

clinical-pharm-platform/  
├── src/  
│ ├── app/ # Next.js App Router pages  
│ │ ├── (auth)/ # Auth-related pages  
│ │ ├── dashboard/ # User dashboard  
│ │ ├── modules/ # Training modules  
│ │ └── api/ # API routes (mainly for Stripe)  
│ ├── components/ # Reusable UI components  
│ │ ├── ui/ # Base UI components  
│ │ ├── auth/ # Authentication components  
│ │ ├── modules/ # Module-specific components  
│ │ └── dashboard/ # Dashboard components  
│ ├── lib/ # Utility functions  
│ │ ├── supabase/ # Supabase client configuration  
│ │ ├── stripe/ # Stripe integration  
│ │ ├── validation/ # Zod schemas  
│ │ └── utils/ # General utilities  
│ └── types/ # TypeScript type definitions  
├── supabase/ # Supabase configuration  
│ ├── migrations/ # Database migrations  
│ ├── functions/ # Edge functions  
│ └── config.toml # Supabase config  
├── public/ # Static assets  
├── docs/ # Project documentation  
├── .env.example # Environment variables template  
├── package.json # Dependencies  
├── supabase-types.ts # Auto-generated Supabase types  
└── PROJECT\_OVERVIEW.md # This file

## 9. DEPLOYMENT STRATEGY

### Development Environment

# Local development setup  
npm install  
supabase start # Start local Supabase instance  
npm run dev # Start Next.js development server

### Staging Environment

* **Frontend**: Vercel preview deployments
* **Backend**: Supabase staging project
* **Domain**: Auto-generated Vercel preview URLs
* **Purpose**: UAT and client demos

### Production Deployment

* **Frontend**: Vercel production deployment
* **Backend**: Supabase production project
* **CDN**: Vercel Edge Network + Supabase CDN
* **Monitoring**: Vercel Analytics + Supabase Dashboard
* **Domain**: learning.daylewis.com

### CI/CD Pipeline

# GitHub Actions workflow  
name: Deploy Pipeline  
on: [push to main]  
jobs:  
 - test: Run Jest, Playwright tests  
 - build: Build Next.js application  
 - deploy-staging: Auto-deploy to Vercel preview  
 - supabase-deploy: Deploy database migrations  
 - integration-tests: Run E2E tests against staging  
 - deploy-production: Deploy to Vercel production (manual approval)  
 - post-deploy: Run smoke tests, notify team

## 10. MIGRATION PLAN

### Content Migration Strategy

**Phase 1: Structure Analysis** 1. Analyze existing HTML modules for content patterns 2. Extract quiz questions and case studies 3. Identify media assets (images, documents) 4. Map content relationships and dependencies

**Phase 2: Data Extraction**

# Custom script to parse HTML content  
npm run migrate:extract-content  
# Outputs: JSON files with structured content data

**Phase 3: Database Population** 1. Create migration scripts for bulk content import 2. Validate content integrity and formatting 3. Set up content approval workflow 4. Migrate user accounts and historical data

**Phase 4: Asset Migration** 1. Upload media files to S3 with proper naming convention 2. Update content references to use CDN URLs 3. Implement lazy loading for performance 4. Set up automated backup procedures

### Migration Commands

npm run migrate:backup-html # Backup existing HTML files  
npm run migrate:extract # Extract structured data  
npm run migrate:validate # Validate extracted content  
npm run migrate:import # Import to database  
npm run migrate:verify # Verify migration success

## 11. TESTING STRATEGY

### Unit Testing

* **Framework**: Jest + React Testing Library
* **Coverage Target**: 80% minimum
* **Focus Areas**: Components, utilities, API endpoints
* **Mocking**: Database queries, external APIs

### Integration Testing

* **Framework**: Supertest for API testing
* **Database**: Test database with Docker
* **Authentication**: Mock NextAuth.js sessions
* **File Uploads**: Mock AWS S3 operations

### End-to-End Testing

* **Framework**: Playwright
* **Test Scenarios**:
  + Complete user registration and login flow
  + Module completion and quiz submission
  + Leader dashboard functionality
  + Payment flow (using Stripe test mode)

### Performance Testing

* **Tools**: Lighthouse CI, Artillery
* **Targets**:
  + Page load times < 2 seconds
  + Time to Interactive < 3 seconds
  + 95th percentile response time < 500ms

### User Acceptance Testing

* **Participants**: 10-15 Day Lewis pharmacy students and leaders
* **Duration**: 2 weeks before MVP launch
* **Feedback Collection**: In-app feedback widget, user interviews

## 12. SECURITY CONSIDERATIONS

### Authentication & Authorization

* **Password Policy**: Minimum 8 characters, complexity requirements
* **Session Management**: JWT with short expiration, refresh tokens
* **MFA**: TOTP for leader accounts, optional for students
* **Domain Restriction**: Strict @daylewis.net email validation

### Data Protection

* **Encryption at Rest**: AES-256 for database and file storage
* **Encryption in Transit**: TLS 1.3 for all communications
* **PII Handling**: Minimal collection, encrypted storage, secure deletion
* **Backup Security**: Encrypted backups with access logging

### Application Security

* **Input Validation**: Zod schemas for all API inputs
* **SQL Injection Prevention**: Parameterized queries with Prisma
* **XSS Protection**: Content Security Policy headers
* **CSRF Protection**: SameSite cookies and CSRF tokens

### Infrastructure Security

* **Network Security**: VPC with private subnets, security groups
* **Access Control**: IAM roles with least privilege principle
* **Monitoring**: Real-time security alerts, access logging
* **Compliance**: Regular security audits and penetration testing

## 13. MONITORING & MAINTENANCE

### Error Tracking

* **Service**: Sentry for error monitoring
* **Alerts**: Real-time notifications for critical errors
* **Logging**: Structured logging with request tracing
* **Dashboards**: Error trends and resolution tracking

### Performance Monitoring

* **APM**: New Relic or DataDog for application performance
* **Database**: Query performance monitoring
* **Infrastructure**: AWS CloudWatch for system metrics
* **User Experience**: Real User Monitoring (RUM)

### Backup Strategies

* **Database**: Daily automated backups with 30-day retention
* **File Storage**: S3 cross-region replication
* **Code**: Git repository with multiple remote backups
* **Configuration**: Infrastructure as Code with Terraform

### Update Procedures

* **Dependencies**: Automated security updates with Dependabot
* **System Updates**: Monthly maintenance windows
* **Content Updates**: Versioned content with rollback capability
* **Database Migrations**: Automated with Prisma, tested in staging

## 14. QUESTIONS & DECISIONS PENDING

### Technical Decisions

* ☐ **Caching Strategy**: Redis vs in-memory caching for quiz data
* ☐ **File Storage**: Direct S3 uploads vs server-side processing
* ☐ **Search Implementation**: PostgreSQL full-text vs Elasticsearch
* ☐ **Real-time Features**: WebSockets vs Server-Sent Events

### Business Logic Clarifications

* ☐ **Subscription Tiers**: Single annual plan vs multiple tiers
* ☐ **Content Access**: Immediate paywall vs freemium model
* ☐ **Quiz Attempts**: Unlimited retakes vs limited attempts
* ☐ **Certification**: Automated certificates vs manual approval

### Compliance & Legal

* ☐ **Data Retention**: How long to keep user data after account deletion
* ☐ **International Access**: Restrictions for non-UK users
* ☐ **Content Licensing**: Rights management for medical content
* ☐ **GDPR Compliance**: Data processing agreement with Day Lewis

### Integration Requirements

* ☐ **Existing Systems**: Integration with Day Lewis HR/training systems
* ☐ **Third-party Tools**: Learning management system compatibility
* ☐ **Analytics**: Google Analytics vs privacy-focused alternatives
* ☐ **Communication**: Email notifications vs in-app messaging

## 15. REFERENCE COMMANDS

### Development Setup

# Initial setup  
git clone <repository>  
cd clinical-pharm-platform  
npm install  
cp .env.example .env  
  
# Start development environment  
supabase login # Login to Supabase CLI  
supabase start # Start local Supabase instance  
supabase db reset # Reset database with fresh migrations  
npm run dev # Start Next.js development server  
  
# Database operations  
supabase db studio # Open Supabase Studio (database GUI)  
supabase db reset # Reset database with fresh migrations  
supabase gen types typescript --local # Generate TypeScript types

### Testing Commands

# Run all tests  
npm run test # Unit tests  
npm run test:e2e # End-to-end tests  
npm run test:coverage # Coverage report  
  
# Specific test suites  
npm run test:api # API integration tests  
npm run test:frontend # Frontend component tests  
npm run test:db # Database tests

### Build & Deployment

# Build for production  
npm run build # Build Next.js application  
npm run type-check # TypeScript type checking  
npm run lint # Lint code  
  
# Supabase operations  
supabase db push # Push database changes to remote  
supabase functions deploy # Deploy edge functions  
supabase gen types typescript # Generate types from remote DB  
  
# Deployment (automatic with Git)  
git push origin main # Deploys to Vercel automatically  
supabase db push --linked # Deploy DB changes to production

### Content Management

# Content migration  
npm run migrate:extract # Extract content from HTML  
npm run migrate:import # Import content to Supabase  
npm run migrate:validate # Validate migrated content  
  
# Content operations  
supabase db dump # Backup all data  
npm run content:sync # Sync content with Supabase Storage  
npm run content:verify # Verify content integrity

## 16. DOCUMENT HISTORY

### Version 1.0 - August 22, 2024

* **Author**: AI Assistant
* **Changes**: Initial comprehensive project overview created
* **Scope**: Complete technical architecture and project planning document
* **Next Review**: After MVP completion (September 17, 2024)

### Version 1.1 - August 23, 2024

* **Author**: AI Assistant
* **Changes**: Updated architecture to use Supabase instead of Express + PostgreSQL
* **Rationale**: Simplified development since data is educational (not healthcare), faster MVP delivery
* **Key Updates**:
  + Replaced Express + PostgreSQL + NextAuth with Supabase
  + Updated deployment strategy to Vercel + Supabase
  + Simplified file structure and development commands
  + Reduced estimated development time by ~60%

### Version 1.2 - [Date TBD]

* **Author**: [Team Member]
* **Changes**: [To be documented]
* **Scope**: [To be documented]

*This document serves as the master reference for the Day Lewis Clinical Pharmacist Training Platform project. It should be updated regularly to reflect current project status, architectural decisions, and implementation progress.*