CMS Project Implementation Details

Frontend Implementation (client/)

public/

- Store static assets including fonts, images, favicon, and robots.txt
- Include a basic index.html template for the SPA
- Set up manifest.json for PWA capabilities
- Add placeholder images for templates and UI elements

src/components/

core/

- **Button.tsx**: Implement customizable button with variants (primary, secondary, ghost)
- **Input.tsx**: Form input components with validation support
- **Modal.tsx**: Reusable modal dialog with customizable header/footer
- **Card.tsx**: Content container with consistent styling
- **Navigation.tsx**: Site navigation components (navbar, sidebar)
- **Typography.tsx**: Text components (headings, paragraphs, lists)
- **Dropdown.tsx**: Dropdown/select components

editor/

- **UIEditor/**
  - **Canvas.tsx**: Main drag-and-drop interface for visual editing
  - **ComponentPalette.tsx**: Available components for drag-and-drop
  - **PropertyPanel.tsx**: Edit component properties
  - **LayerNavigator.tsx**: Hierarchical view of components
  - **ActionToolbar.tsx**: Undo/redo and editing actions
- **CodeEditor/**
  - **MonacoWrapper.tsx**: Integration with Monaco Editor
  - **Linter.tsx**: Real-time code validation
  - **AutoComplete.tsx**: Context-aware code suggestions
  - **ThemeSelector.tsx**: Editor theme customization
- **SyncEngine/**
  - **DiffGenerator.tsx**: Compare visual/code changes

    - **ConflictResolver.tsx**: Handle sync conflicts
    - **SyncControls.tsx**: Toggle auto-sync options
- **Templates/**
    - **TemplateCard.tsx**: Template preview with selection
    - **TemplateImporter.tsx**: Import custom templates
    - **CategoryFilter.tsx**: Filter templates by category

preview/

- **DevicePreview.tsx**: Responsive previews (mobile, tablet, desktop)
- **PreviewNavigation.tsx**: Navigate between pages in preview
- **PreviewControls.tsx**: Refresh, share preview links
- **InteractivePreview.tsx**: Click-to-edit from preview

ui/

- Implement shadcn/ui components customized for the CMS theme
- Include dark/light mode variants
- Ensure accessibility compliance (ARIA attributes, keyboard navigation)

pages/

- **Dashboard.tsx**: Project overview and metrics
- **Editor.tsx**: Main editing interface combining UI/code editors
- **Projects.tsx**: Project listing and management
- **Settings.tsx**: User and system settings
- **Templates.tsx**: Template browsing/selection
- **Preview.tsx**: Full-page preview with responsive tools
- **Deploy.tsx**: Deployment options and history

hooks/

- **useAutosave.ts**: Auto-save edits with configurable intervals
- **useMediaQuery.ts**: Responsive design helpers
- **useDragAndDrop.ts**: Drag-and-drop functionality
- **useCodeValidation.ts**: Validate code in real-time
- **useComponentSync.ts**: Sync between visual/code editors
- **useHistory.ts**: Manage edit history and undo/redo

store/

- **slices/**
  - **projectSlice.ts**: Project metadata and settings
  - **editorSlice.ts**: Editor state and preferences
  - **componentsSlice.ts**: Component tree and properties
  - **codeSlice.ts**: Code representation of the project
  - **userSlice.ts**: User preferences and authentication
  - **previewSlice.ts**: Preview settings and state
- **middleware/**
  - **syncMiddleware.ts**: Keep visual and code editors in sync
  - **validationMiddleware.ts**: Validate changes before save
  - **persistenceMiddleware.ts**: Handle auto-saving to backend
  - **undoRedoMiddleware.ts**: Track changes for history

services/

- **api.ts**: Base API client configuration
- **projectService.ts**: Project CRUD operations
- **authService.ts**: Authentication and user management
- **templateService.ts**: Template retrieval and management
- **deployService.ts**: Deployment API integration
- **exportService.ts**: Export projects to different formats

utils/

- **componentMapper.ts**: Map visual components to code
- **codeParser.ts**: Parse code into component structure
- **cssUtils.ts**: CSS manipulation helpers
- **validation.ts**: Input and code validation
- **responsiveUtils.ts**: Responsive design helpers
- **storageUtils.ts**: Local storage management

types/

- **component.ts**: Component type definitions
- **project.ts**: Project structure types
- **user.ts**: User and authentication types

- **editor.ts**: Editor state types
- **api.ts**: API response and request types

styles/

- **global.css**: Base CSS and resets
- **variables.css**: CSS variables for theming
- **animations.css**: Common animations
- **editor.css**: Editor-specific styles

Backend Implementation (server/)

src/controllers/

- **authController.ts**: User authentication endpoints
- **projectController.ts**: Project management endpoints
- **templateController.ts**: Template management endpoints
- **deployController.ts**: Deployment control endpoints
- **renderController.ts**: Preview rendering endpoints

src/models/

- **User.ts**: User account schema
- **Project.ts**: Project data schema
- **Template.ts**: Template schema
- **Deployment.ts**: Deployment history schema
- **Asset.ts**: Media assets schema

src/services/

auth/

- **authentication.ts**: User authentication logic
- **authorization.ts**: Permission checking
- **userManagement.ts**: User CRUD operations
- **tokenService.ts**: JWT token generation/validation

sites/

- **projectManager.ts**: Project CRUD operations
- **assetManager.ts**: Media asset handling
- **versionControl.ts**: Project version history

- **exportManager.ts**: Export to different formats

codegen/

- **htmlGenerator.ts**: Generate HTML from component tree
- **cssGenerator.ts**: Generate CSS from style properties
- **jsGenerator.ts**: Generate JS for interactive elements
- **validatorService.ts**: Validate generated code

renderer/

- **previewRenderer.ts**: Generate live previews
- **seoAnalyzer.ts**: SEO recommendations
- **performanceAnalyzer.ts**: Page performance metrics
- **responsiveTester.ts**: Test different screen sizes

deploy/

- **staticDeployer.ts**: Deploy to static hosting
- **domainManager.ts**: Custom domain configuration
- **buildProcess.ts**: Build optimization process
- **deploymentMonitor.ts**: Monitor deployment status

src/routes/

- **authRoutes.ts**: Authentication endpoints
- **projectRoutes.ts**: Project management endpoints
- **templateRoutes.ts**: Template endpoints
- **deployRoutes.ts**: Deployment endpoints
- **assetRoutes.ts**: Asset management endpoints

src/middleware/

- **authentication.ts**: Verify user authentication
- **rateLimiter.ts**: API rate limiting
- **errorHandler.ts**: Consistent error responses
- **logger.ts**: Request logging
- **validator.ts**: Request validation

src/utils/

- **encryption.ts**: Data encryption utilities

- **fileSystem.ts**: File management utilities
- **stringManipulation.ts**: String processing utilities
- **configLoader.ts**: Load configuration based on environment
- **dateTime.ts**: Date and time utilities

src/db/

- **connection.ts**: Database connection setup
- **migrations/**: Database schema migrations
- **seeders/**: Initial data seeders
- **queryBuilders/**: Complex query helpers

Integration Points

1. **Visual-to-Code Synchronization:**
   - Frontend SyncEngine connects components to real-time code
   - Changes in visual editor update code editor (and vice versa)
   - Conflict resolution when both editors have changes
2. **Preview System:**
   - Frontend sends current state to backend renderer
   - Renderer service generates preview HTML
   - Preview components display rendered output with device frames
3. **Deployment Pipeline:**
   - Frontend collects deployment settings
   - Backend build process optimizes assets
   - Deployment service pushes to selected hosting platform
   - Status updates sent back to frontend
4. **Template System:**
   - Templates stored in database with metadata
   - Frontend loads and displays template previews
   - Selected template instantiated in editor
   - Custom templates can be saved from projects
5. **Authentication Flow:**
   - Frontend auth service manages tokens
   - Backend validates tokens on protected routes
   - Role-based access controls project editing/viewing

Technical Requirements

1. **Performance Considerations:**
   - Code editor must handle large files without lag
   - Visual editor needs efficient DOM operations
   - Renderer service should cache when possible
   - State management optimized for frequent updates
2. **Security Requirements:**
   - Sanitize all user inputs
   - Validate generated code for security issues
   - Implement CSRF protection
   - Secure API endpoints with proper authentication
3. **Accessibility:**
   - All components meet WCAG 2.1 AA standards
   - Keyboard navigation throughout the application
   - Screen reader compatibility
   - Color contrast requirements met
4. **Responsive Design:**
   - All interfaces work on devices from 320px up
   - Editor has adaptive layout for smaller screens
   - Preview tool shows different device sizes
5. **Browser Compatibility:**
   - Support latest two versions of major browsers
   - Graceful degradation for older browsers
   - Polyfills for essential features