## Supplementary Information for ReAlign MVP AI Coding Agent

This document provides additional configurations, schemas, and content snippets derived from the project documentation to aid in the development of ReAlign MVP.

---

### 1. Asset Information

\*\*Logo:\*\*

\* The specific ReAlign logo file (SVG/PNG) needs to be provided separately. The documents refer to "ReAlign logo"[cite: 446].

\*\*Icons (from lucide-react):\*\*

Based on the `TransactionView.tsx` scaffold and Brand Kit, common icons to be potentially preloaded or frequently used include[cite: 360, 459]:

\* `UploadCloud` (for uploads) [cite: 360, 459]

\* `Send` (for sending messages) [cite: 360]

\* `FilePlus2` (for adding files/documents) [cite: 360]

\* `Clock` (for status, due dates, or time-related info) [cite: 360, 459]

\* `Eye` (for visibility toggles or viewing) [cite: 360]

\* `MessageCircle` (for messages or communication features) [cite: 360, 459]

\* `CheckCircle` (for complete status, success) [cite: 360, 459]

\* `AlertCircle` (for warnings, pending status, errors) [cite: 360, 459]

\* `ArrowRight`, `ChevronDown` (for navigation) [cite: 459]

---

### 2. Configuration Files (Examples)

\*\*ESLint Configuration (`.eslintrc.js` - Basic Example):\*\*

```javascript

// .eslintrc.js

module.exports = {

env: {

browser: true,

es2021: true,

node: true,

},

extends: [

'eslint:recommended',

'plugin:react/recommended',

'plugin:@typescript-eslint/recommended', // If using TypeScript

'prettier', // Add Prettier last to override other formatting rules

],

parser: '@typescript-eslint/parser', // If using TypeScript

parserOptions: {

ecmaFeatures: {

jsx: true,

},

ecmaVersion: 'latest',

sourceType: 'module',

},

plugins: [

'react',

'@typescript-eslint/eslint-plugin', // If using TypeScript

'prettier',

],

rules: {

'prettier/prettier': 'error',

'react/react-in-jsx-scope': 'off', // For React 17+ with new JSX transform

'react/prop-types': 'off', // Useful in TypeScript projects

// Add any project-specific rules here

},

settings: {

react: {

version: 'detect', // Automatically detects React version

},

},

};

*Rationale: This basic ESLint setup includes React and TypeScript support (as inferred from .tsx files like TransactionView.tsx and the use of React ) and integrates Prettier as specified.*

**Prettier Configuration (.prettierrc.js - Basic Example):**

JavaScript

// .prettierrc.js

module.exports = {

semi: true,

singleQuote: false, // Or true, based on preference

tabWidth: 2,

trailingComma: 'es5', // Or 'all'

printWidth: 100, // Or 80

arrowParens: 'always',

};

*Rationale: A common Prettier setup. The Technical Architecture mentions Prettier for code style.*

**TailwindCSS Theme Configuration (tailwind.config.js - Based on Brand Kit):**

JavaScript

// tailwind.config.js

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

content: [

'./src/pages/\*\*/\*.{js,ts,jsx,tsx,mdx}',

'./src/components/\*\*/\*.{js,ts,jsx,tsx,mdx}',

'./src/app/\*\*/\*.{js,ts,jsx,tsx,mdx}',

// For shadcn/ui, if used in the typical way

// './components/\*\*/\*.{ts,tsx}',

// './app/\*\*/\*.{ts,tsx}',

// './src/\*\*/\*.{ts,tsx}',

],

theme: {

extend: {

colors: {

brand: {

primary: '#263b75', // Primary Blue [cite: 456]

accent: '#f2c94c', // Accent Yellow [cite: 456]

},

background: {

light: '#f9f9f9', // Light Background [cite: 456]

},

text: {

dark: '#1c1c1e', // Dark Text [cite: 456]

},

status: {

success: '#2ecc71', // Success [cite: 456]

warning: '#f39c12', // Warning [cite: 456]

danger: '#e74c3c', // Danger [cite: 456]

},

// shadcn/ui preset (example, adjust if using shadcn/ui theme generation)

border: 'hsl(var(--border))',

input: 'hsl(var(--input))',

ring: 'hsl(var(--ring))',

// background: 'hsl(var(--background))', // ReAlign has its own background.light

foreground: 'hsl(var(--foreground))',

primary: {

DEFAULT: 'hsl(var(--primary))', // This would map to ReAlign's Primary Blue

foreground: 'hsl(var(--primary-foreground))',

},

secondary: {

DEFAULT: 'hsl(var(--secondary))',

foreground: 'hsl(var(--secondary-foreground))',

},

destructive: {

DEFAULT: 'hsl(var(--destructive))', // Maps to ReAlign's Danger color

foreground: 'hsl(var(--destructive-foreground))',

},

muted: {

DEFAULT: 'hsl(var(--muted))',

foreground: 'hsl(var(--muted-foreground))',

},

accent: {

DEFAULT: 'hsl(var(--accent))', // Maps to ReAlign's Accent Yellow

foreground: 'hsl(var(--accent-foreground))',

},

popover: {

DEFAULT: 'hsl(var(--popover))',

foreground: 'hsl(var(--popover-foreground))',

},

card: {

DEFAULT: 'hsl(var(--card))',

foreground: 'hsl(var(--card-foreground))',

},

},

fontFamily: {

sans: ['Inter', '-apple-system', 'BlinkMacSystemFont', 'Segoe UI', 'Roboto', 'Helvetica Neue', 'Arial', 'sans-serif'], // Primary Font: Inter [cite: 456]

},

borderRadius: {

'2xl': '12px', // As per Brand Kit button styles [cite: 456]

// shadcn/ui uses logical names like lg, md, sm

lg: 'var(--radius)',

md: 'calc(var(--radius) - 2px)',

sm: 'calc(var(--radius) - 4px)',

},

// Spacing, keyframes for animations (framer-motion [cite: 22]) can be added here if needed

// Keyframes for shadcn/ui animations

keyframes: {

"accordion-down": {

from: { height: "0" },

to: { height: "var(--radix-accordion-content-height)" },

},

"accordion-up": {

from: { height: "var(--radix-accordion-content-height)" },

to: { height: "0" },

},

},

animation: {

"accordion-down": "accordion-down 0.2s ease-out",

"accordion-up": "accordion-up 0.2s ease-out",

},

},

},

plugins: [

require('tailwindcss-animate') // For shadcn/ui animations

],

};

*Rationale: This configuration directly translates the Brand Kit's color palette, typography (Inter font), and relevant button styles (border-radius) into a Tailwind theme. It also includes basic setup for shadcn/ui theming and animations, as shadcn/ui is a core part of the frontend stack.*

### **3. Detailed Data Validation Schemas (Zod Examples)**

**Auth Schemas (based on API Routes ):**

TypeScript

// src/lib/validators/auth.ts (example path)

import { z } from 'zod';

export const LoginSchema = z.object({

email: z.string().email('Invalid email address.'),

password: z.string().min(1, 'Password is required.'), // Consider adding min length, e.g., .min(8, 'Password must be at least 8 characters long.')

});

export const MagicLinkRequestSchema = z.object({

email: z.string().email('Invalid email address.').optional(),

phone: z.string().optional(), // Add more specific phone validation if needed

}).refine(data => data.email || data.phone, {

message: 'Either email or phone must be provided.',

path: ['email'], // Or path: ['phone'] or a general path

});

**Transaction Creation Schema (based on API Routes and Database Schema ):**

TypeScript

// src/lib/validators/transaction.ts (example path)

import { z } from 'zod';

const PartySchema = z.object({

name: z.string().min(1, 'Party name is required.'),

email: z.string().email('Invalid email address for party.'),

phone: z.string().optional(), // Consider more specific validation

role: z.enum(['seller', 'buyer', 'listing\_agent', 'buyers\_agent', 'escrow']), // [cite: 270]

});

const PHASES = [ // [cite: 379, 272]

"Transaction Initiated", "Property Listed", "Initial Document Collection",

"Offer Received", "Offer Submitted", "Lender Review",

"BPO Ordered", "Approval Received", "In Closing"

] as const;

export const CreateTransactionSchema = z.object({

title: z.string().min(1, 'Transaction title is required.'),

property\_address: z.string().min(1, 'Property address is required.'), // [cite: 272]

parties: z.array(PartySchema).min(1, 'At least one party must be added.'),

initialPhase: z.enum(PHASES), // [cite: 272]

initialMessage: z.string().optional(),

});

export const UpdateTransactionSchema = z.object({

title: z.string().min(1, 'Transaction title is required.').optional(),

property\_address: z.string().min(1, 'Property address is required.').optional(),

currentPhase: z.enum(PHASES).optional(),

});

**Document Request Schema (based on API Routes and Database Schema ):**

TypeScript

// src/lib/validators/documentRequest.ts (example path)

import { z } from 'zod';

const DOC\_REQUEST\_STATUSES = ['requested', 'pending', 'complete', 'overdue'] as const; // [cite: 283]

export const CreateDocumentRequestSchema = z.object({

docType: z.string().min(1, 'Document type is required.'), // From hardcoded preset list [cite: 285, 121]

assignedToUserId: z.string().uuid('Invalid user ID format.'),

dueDate: z.string().refine((date) => !isNaN(Date.parse(date)), { message: "Invalid date format, expected YYYY-MM-DD" }).optional(), // [cite: 345]

});

export const UpdateDocumentRequestSchema = z.object({

status: z.enum(DOC\_REQUEST\_STATUSES).optional(),

revisionNote: z.string().optional(), // [cite: 286, 350]

dueDate: z.string().refine((date) => !isNaN(Date.parse(date)), { message: "Invalid date format, expected YYYY-MM-DD" }).optional(),

});

**File Upload Metadata Schema (based on API Routes and Component Spec ):**

TypeScript

// src/lib/validators/upload.ts (example path)

import { z } from 'zod';

export const FileUploadMetadataSchema = z.object({

// 'file' field handled by multipart/form-data, backend validates file itself

docType: z.string().min(1, 'Document type is required.'), // [cite: 353]

visibility: z.enum(['private', 'shared']), // [cite: 353, 289]

// transactionId is part of the route, not body for this example

});

*Rationale: These Zod schemas are directly derived from the API request body structures specified in the API Routes document and field constraints in the Database Schema. They include basic checks like non-empty strings, email format, and enum validation where appropriate.*

### **4. Content & Copy**

**Example HTML Email Template (Magic Link Delivery):**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Your ReAlign Access Link</title>

<style>

body { font-family: 'Inter', Arial, sans-serif; margin: 0; padding: 20px; background-color: #f9f9f9; color: #1c1c1e; }

.container { background-color: #ffffff; padding: 30px; border-radius: 12px; max-width: 600px; margin: auto; box-shadow: 0 4px 6px rgba(0,0,0,0.1); }

.header { text-align: center; margin-bottom: 20px; }

.header img { max-width: 150px; } /\* Placeholder for ReAlign Logo \*/

.content p { font-size: 16px; line-height: 1.6; }

.button { display: inline-block; background-color: #263b75; color: #ffffff; padding: 12px 25px; text-decoration: none; border-radius: 12px; font-weight: 500; margin-top: 20px; text-align: center; }

.footer { margin-top: 30px; text-align: center; font-size: 14px; color: #666; }

</style>

</head>

<body>

<div class="container">

<div class="header">

<h2>ReAlign Access</h2>

</div>

<div class="content">

<p>Hello,</p>

<p>Your access link to ReAlign is below. It expires in 24 hours or after first use: [MagicLinkURL]</p> <p style="text-align: center;">

<a href="[MagicLinkURL]" class="button">Access ReAlign</a>

</p>

<p>If you did not request this link, please ignore this email.</p>

</div>

<div class="footer">

<p>&copy; {{currentYear}} ReAlign. All rights reserved.</p>

</div>

</div>

</body>

</html>

*Rationale: This HTML template structures the magic link delivery message using styling inspired by the Brand Kit (colors, fonts). The [MagicLinkURL] placeholder would be replaced by the actual generated link.*

**Consolidated UI Text Snippets (Examples):**

* **Magic Link Screen:**
  + Button: "Send Access Link"
  + Subtext: "We’ll email or text you a one-time link. No password required."
  + Expired Token Message: "This link has expired. Request a new one below."
  + Max Resends Toast: 'You have exceeded the maximum number of resend attempts. Please try again later.'
* **Upload Widget:**
  + Error Message (on fail): 'Upload failed. Please try again.' or 'Upload failed. File too large. Please try again.'
  + Retry Button: "Retry"
  + Informational Subtext: 'Uploads are private to you and the negotiator by default.'
  + Upload Button Text (example): "Upload" (often used with an icon like UploadCloud)
* **Transaction View (Negotiator):**
  + Phase Dropdown Label: "Current Phase:"
  + Phase Dropdown Placeholder: "Select phase"
  + Message Input Placeholder: "Type your message..."
  + Document Request Placeholder: "Request a new document..."
  + Revision Note Placeholder: "Reason for revision..." or "Missing signature on page 2"
* **Session Expiry (Negotiator):**
  + Message: "Your session has expired. Please log in again."
* **General:**
  + Loading Message: "Loading authentication details..."
  + Not Authenticated: "Not authenticated. Redirecting..."
  + Error Dismiss Button: "Dismiss"

*Rationale: These text snippets are directly extracted or inferred from the Figma Wireframe Spec, Component Specification, User Onboarding Flow, Notification Rules, and the TransactionView.tsx scaffold.*