Credit & Funding Platform

Complete Technical Specification

Credit & Funding Platform - Complete Specification (Part 2)

Continuation from Part 1

4. Tech Stack (Continued)

4.2 Package Installation Commands

Frontend (apps/web)

```
# Navigate to web app
cd apps/web

# Core Next.js
pnpm add next@latest react react-dom
pnpm add -D typescript @types/react @types/node @types/react-dom

# Styling
pnpm add tailwindcss postcss autoprefixer
pnpm add -D @tailwindcss/forms
npx tailwindcss init -p

# shadcn/ui (install CLI then add components)
npx shadcn-ui@latest init
npx shadcn-ui@latest add button card dialog dropdown-menu
npx shadcn-ui@latest add form input label select checkbox radio-group
npx shadcn-ui@latest add table tabs toast badge avatar progress
npx shadcn-ui@latest add command popover sheet tooltip accordion alert-dialog

# Icons
pnpm add lucide-react

# Forms & Validation
pnpm add react-hook-form zod @hookform/resolvers

# Data Fetching
pnpm add @tanstack/react-query axios

# Charts
```

```
pnpm add recharts

# Dates
pnpm add date-fns

# State (if needed)
pnpm add zustand

# Dev Dependencies
pnpm add -D @tanstack/eslint-plugin-query
```

Backend (apps/api)

```
# Navigate to API app
cd apps/api

# NestJS Core
pnpm add @nestjs/common @nestjs/core @nestjs/platform-express
pnpm add reflect-metadata rxjs

# CLI (for generating modules)
pnpm add -g @nestjs/cli

# Configuration
pnpm add @nestjs/config

# Database
pnpm add @prisma/client
pnpm add -D prisma

# Validation
pnpm add class-validator class-transformer

# Authentication
pnpm add @nestjs/jwt @nestjs/passport passport passport-jwt
pnpm add bcrypt
pnpm add -D @types/bcrypt @types/passport-jwt

# Queue & Jobs
pnpm add @nestjs/bull bullmq ioredis
pnpm add -D @types/ioredis

# HTTP Client
pnpm add axios

# Utilities
pnpm add dayjs

# Email (optional)
pnpm add @nestjs-modules/mailer nodemailer
pnpm add -D @types/nodemailer

# PDF Processing (for upload fallback)
pnpm add pdf-parse
# Encryption
```

```
pnpm add @nestjs/config

# Rate Limiting
pnpm add @nestjs/throttler
```

Shared Packages (packages/db)

```
# Navigate to db package
cd packages/db

# Prisma
pnpm add @prisma/client
pnpm add -D prisma

# Initialize
npx prisma init
```

4.3 Environment Variables

Backend (.env)

```
# ====================================
# DATABASE
# ====================================
DATABASE_URL="postgresql://user:password@localhost:5432/credit_platform"

# ====================================
# JWT Authentication
# ====================================
JWT_SECRET="your-super-secret-jwt-key-min-32-characters-change-in-production"
JWT_EXPIRES_IN="15m"
JWT_REFRESH_SECRET="your-refresh-secret-also-change-this"
JWT_REFRESH_EXPIRES_IN="7d"

# ====================================
# REDIS
# ====================================
REDIS_URL="redis://localhost:6379"
REDIS_PASSWORD="" # If using managed Redis

# ====================================
# CREDIT VENDOR (iSoftPull)
# ====================================
CREDIT_VENDOR="ISOFTPULL"
ISOFTPULL_API_KEY="your-isoftpull-api-key"
ISOFTPULL_API_URL="https://api.isoftpull.com/v1"
ISOFTPULL_API_SECRET="your-api-secret"

# ====================================
# AI PROVIDERS
# ====================================
ANTHROPIC_API_KEY="sk-ant-api03-..."
```

```
OPENAI_API_KEY="sk-..."

# ====================================
# EMAIL (SendGrid Example)
# ====================================
SMTP_HOST="smtp.sendgrid.net"
SMTP_PORT="587"
SMTP_USER="apikey"
SMTP_PASS="SG.your-sendgrid-api-key"
EMAIL_FROM="noreply@yourcreditplatform.com"
EMAIL_FROM_NAME="Credit Platform"

# ====================================
# APPLICATION
# ====================================
NODE_ENV="development" # or "production"
PORT="3001"
FRONTEND_URL="http://localhost:3000"
API_URL="http://localhost:3001"
```

Frontend (.env.local)

```
# ====================================
# API
# ====================================
NEXT_PUBLIC_API_URL="http://localhost:3001"

# ====================================
# AUTH (if using NextAuth)
# ====================================
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="your-nextauth-secret-32-chars-minimum"

# ====================================
# PUBLIC CONFIGS
# ====================================
NEXT_PUBLIC_APP_NAME="Credit & Funding Platform"
NEXT_PUBLIC_BRAND_COLOR="#8faa76"
NEXT_PUBLIC_SUPPORT_EMAIL="support@yourcreditplatform.com"

# ====================================
# ANALYTICS
# ====================================
NEXT_PUBLIC_POSTHOG_KEY="phc_..."
NEXT_PUBLIC_POSTHOG_HOST="https://app.posthog.com"

# ====================================
# SENTRY (optional)
# ====================================
NEXT_PUBLIC_SENTRY_DSN="https://...@sentry.io/..."
```

5. Branding and UI

5.1 Brand Identity

Color System

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        // Primary Brand Color (#8faa76)
        primary: {
          50: '#f5f8f3',
          100: '#e8f0e3',
          200: '#d1e2c7',
          300: '#adcf9d',
          400: '#8faa76',  // Main brand color - USE THIS
          500: '#6b8c54',
          600: '#527040',
          700: '#415834',
          800: '#36472d',
          900: '#2d3b26',
        },

        // Neutral Grays
        gray: {
          50: '#fafafa',
          100: '#f5f5f5',
          200: '#e5e5e5',
          300: '#d4d4d4',
          400: '#a3a3a3',
          500: '#737373',
          600: '#525252',
          700: '#404040',
          800: '#262626',
          900: '#171717',
        },

        // Success (green - funding approved, checks passed)
        success: {
          50: '#f0fdf4',
          100: '#dcfce7',
          200: '#bbf7d0',
          500: '#22c55e',
          600: '#16a34a',
          700: '#15803d',
        },

        // Warning (amber - needs attention)
        warning: {
          50: '#fffbeb',
          100: '#fef3c7',
          200: '#fde68a',
          500: '#f59e0b',
          600: '#d97706',
```

Typography Scale

```
// Font Configuration
{
  fontFamily: {
    sans: ['Inter var', 'system-ui', 'sans-serif'],
    mono: ['JetBrains Mono', 'Monaco', 'Courier New', 'monospace']
  },

  fontSize: {
    xs: ['0.75rem', { lineHeight: '1rem' }],      // 12px - captions, helper text
    sm: ['0.875rem', { lineHeight: '1.25rem' }],  // 14px - secondary text
    base: ['1rem', { lineHeight: '1.5rem' }],     // 16px - body text
    lg: ['1.125rem', { lineHeight: '1.75rem' }],  // 18px - large body, subtitles
    xl: ['1.25rem', { lineHeight: '1.75rem' }],   // 20px - small headings
    '2xl': ['1.5rem', { lineHeight: '2rem' }],    // 24px - section headings
    '3xl': ['1.875rem', { lineHeight: '2.25rem' }], // 30px - page titles
    '4xl': ['2.25rem', { lineHeight: '2.5rem' }],   // 36px - hero text
    '5xl': ['3rem', { lineHeight: '1' }],           // 48px - large metrics
    '6xl': ['3.75rem', { lineHeight: '1' }],        // 60px - dashboard numbers
    '7xl': ['4.5rem', { lineHeight: '1' }],         // 72px - hero metrics
    '8xl': ['6rem', { lineHeight: '1' }],           // 96px - extra large
  },

  fontWeight: {
    light: '300',
    normal: '400',
    medium: '500',
    semibold: '600',
    bold: '700',
    extrabold: '800',
  }
}
```

Typography Usage Guide:

- **Body text:** `base` (16px) or `lg` (18px) for readability

- **Labels:** `sm` (14px) for form labels, badges

- **Helper text:** `xs` (12px) for captions, timestamps

- **Headings:** `2xl` to `4xl` (24px-36px) for sections

- **Metrics:** `5xl` to `7xl` (48px-72px) for big dashboard numbers

- **Hero text:** `6xl`+ (60px+) for landing pages

Spacing System

```
// Consistent 4px base unit
{
  spacing: {
    px: '1px',
    0: '0',
    0.5: '0.125rem',  // 2px
    1: '0.25rem',     // 4px
    1.5: '0.375rem',  // 6px
    2: '0.5rem',      // 8px
```

```
      2.5: '0.625rem',  // 10px
      3: '0.75rem',     // 12px
      3.5: '0.875rem',  // 14px
      4: '1rem',        // 16px - standard
      5: '1.25rem',     // 20px
      6: '1.5rem',      // 24px - card padding
      7: '1.75rem',     // 28px
      8: '2rem',        // 32px - section spacing
      9: '2.25rem',     // 36px
      10: '2.5rem',     // 40px
      11: '2.75rem',    // 44px
      12: '3rem',       // 48px
      14: '3.5rem',     // 56px
      16: '4rem',       // 64px
      20: '5rem',       // 80px
      24: '6rem',       // 96px
    }
  }
```
Spacing Patterns:

- **Card padding:** `p-6` (24px)

- **Section spacing:** `space-y-8` or `space-y-12` (32px-48px)

- **Button padding:** `px-4 py-2` (16px horizontal, 8px vertical)

- **Input padding:** `px-3 py-2` (12px horizontal, 8px vertical)

- **Container max width:** `max-w-7xl` (1280px)

Shadow & Elevation

```
  {
    boxShadow: {
      sm: '0 1px 2px 0 rgb(0 0 0 / 0.05)',           // Subtle
      DEFAULT: '0 1px 3px 0 rgb(0 0 0 / 0.1)',       // Cards
      md: '0 4px 6px -1px rgb(0 0 0 / 0.1)',         // Elevated cards
      lg: '0 10px 15px -3px rgb(0 0 0 / 0.1)',       // Modals
      xl: '0 20px 25px -5px rgb(0 0 0 / 0.1)',       // Large modals
      '2xl': '0 25px 50px -12px rgb(0 0 0 / 0.25)',  // Dramatic
      inner: 'inset 0 2px 4px 0 rgb(0 0 0 / 0.05)',  // Inputs
    },

    borderRadius: {
      none: '0',
      sm: '0.125rem',     // 2px
      DEFAULT: '0.25rem', // 4px
      md: '0.375rem',     // 6px - buttons, inputs
      lg: '0.5rem',       // 8px - cards
      xl: '0.75rem',      // 12px - large cards
      '2xl': '1rem',      // 16px
      '3xl': '1.5rem',    // 24px
      full: '9999px',     // circles, pills
    }
  }
```

5.2 Design Principles

1. Visual Hierarchy Through Size

```
// Good example - clear hierarchy
<div className="space-y-2">
  {/* Most important - biggest */}
  <h1 className="text-6xl font-bold text-gray-900">
    $125,000
  </h1>

  {/* Supporting context - medium */}
  <p className="text-lg text-gray-600">
    Total funded this month
  </p>

  {/* Additional detail - smallest */}
  <p className="text-sm text-gray-500">
    Across 23 clients • 12% increase from last month
  </p>
</div>
```

2. White Space is Content

```
// Bad - cramped
<div className="p-2 space-y-1">
  <h2>Title</h2>
  <p>Content</p>
</div>

// Good - breathing room
<div className="p-6 space-y-4">
  <h2 className="text-2xl font-semibold">Title</h2>
  <p className="text-base leading-relaxed">
    Content with proper spacing
  </p>
</div>
```

3. Color with Purpose

Each color has a specific job:

- **Primary green (#8faa76)** - Brand, CTAs, links

- **Success green** - Positive states, approved, passed checks

- **Warning amber** - Needs attention, in progress

- **Danger red** - Errors, failed, declined

- **Gray** - Neutral content, borders, backgrounds

- **Info blue** - Informational messages

4. Consistent Component Patterns

Reuse the same patterns throughout the app for consistency.

5.3 Key UI Components

Status Badge Component

```
// components/ui/status-badge.tsx
import { cva, type VariantProps } from "class-variance-authority";

const statusBadgeVariants = cva(
  "inline-flex items-center gap-1.5 px-2.5 py-0.5 text-xs font-medium rounded-md border",
  {
    variants: {
      status: {
        success: "bg-success-50 text-success-700 border-success-200",
        warning: "bg-warning-50 text-warning-700 border-warning-200",
        danger: "bg-danger-50 text-danger-700 border-danger-200",
        info: "bg-info-50 text-info-700 border-info-200",
        neutral: "bg-gray-50 text-gray-700 border-gray-200",
      }
    },
    defaultVariants: {
      status: "neutral"
    }
  }
);

export function StatusBadge({ status, children }: StatusBadgeProps) {
  return (
    <span className={statusBadgeVariants({ status })}>
      {children}
    </span>
  );
}

// Usage
<StatusBadge status="success">Fundable</StatusBadge>
<StatusBadge status="danger">Not Ready</StatusBadge>
<StatusBadge status="warning">In Progress</StatusBadge>
```

Metric Card Component

```
// components/dashboard/metric-card.tsx
export function MetricCard({
  title,
  value,
  subtitle,
  trend,
  icon: Icon
}: MetricCardProps) {
  return (
    <div className="bg-white rounded-lg border border-gray-200 p-6">
```

```
      {/* Header with icon */}
      <div className="flex items-center justify-between">
        <p className="text-sm font-medium text-gray-600">
          {title}
        </p>
        {Icon && <Icon className="w-5 h-5 text-gray-400" />}
      </div>

      {/* Main value */}
      <p className="mt-2 text-3xl font-semibold text-gray-900">
        {value}
      </p>

      {/* Subtitle */}
      {subtitle && (
        <p className="mt-1 text-sm text-gray-500">
          {subtitle}
        </p>
      )}

      {/* Trend indicator */}
      {trend && (
        <div className="mt-2 flex items-center gap-1 text-sm">
          {trend > 0 ? (
            <>
              <TrendingUp className="w-4 h-4 text-success-500" />
              <span className="text-success-700">+{trend}%</span>
            </>
          ) : (
            <>
              <TrendingDown className="w-4 h-4 text-danger-500" />
              <span className="text-danger-700">{trend}%</span>
            </>
          )}
          <span className="text-gray-500">vs last month</span>
        </div>
      )}
    </div>
  );
}
```

Data Table Component

```
// components/ui/data-table.tsx
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table";

export function DataTable({ columns, data }) {
  return (
    <div className="rounded-lg border border-gray-200 overflow-hidden">
      <Table>
        <TableHeader>
          <TableRow className="bg-gray-50">
            {columns.map((column) => (
              <TableHead key={column.key} className="font-semibold">
                {column.label}
              </TableHead>
            ))}
          </TableRow>
        </TableHeader>
        <TableBody>
          {data.length === 0 ? (
            <TableRow>
              <TableCell
                colSpan={columns.length}
                className="text-center text-gray-500 py-8"
              >
                No data available
              </TableCell>
            </TableRow>
          ) : (
            data.map((row, rowIndex) => (
              <TableRow
                key={rowIndex}
                className="hover:bg-gray-50 transition-colors"
              >
                {columns.map((column) => (
                  <TableCell key={column.key}>
                    {column.render
                      ? column.render(row[column.key], row)
                      : row[column.key]}
                  </TableCell>
                ))}
              </TableRow>
            ))
          )}
        </TableBody>
      </Table>
    </div>
  );
}
```

Empty State Component

```
// components/ui/empty-state.tsx
export function EmptyState({
  icon: Icon,
  title,
  description,
  action
}: EmptyStateProps) {
  return (
    <div className="text-center py-12">
      {Icon && (
        <Icon className="mx-auto h-12 w-12 text-gray-400" />
      )}
      <h3 className="mt-2 text-sm font-semibold text-gray-900">
        {title}
      </h3>
      <p className="mt-1 text-sm text-gray-500">
        {description}
      </p>
      {action && (
        <div className="mt-6">
          {action}
        </div>
      )}
    </div>
  );
}
```

5.4 Layout Components

App Shell Structure

```
// components/layout/app-shell.tsx
export function AppShell({ children }: { children: React.ReactNode }) {
  return (
    <div className="min-h-screen bg-gray-50">
      {/* Sidebar */}
      <Sidebar />

      {/* Main Content Area */}
      <div className="lg:pl-64">
        {/* Top Bar */}
        <TopBar />

        {/* Page Content */}
        <main className="py-8 px-4 sm:px-6 lg:px-8">
          <div className="mx-auto max-w-7xl">
            {children}
          </div>
        </main>
      </div>
    </div>
  );
}
```

Sidebar Navigation

```
// components/layout/sidebar.tsx
const navigation = [
  { name: 'Dashboard', href: '/dashboard', icon: LayoutDashboard },
  { name: 'Clients', href: '/dashboard/clients', icon: Users },
  { name: 'Funding', href: '/dashboard/funding', icon: DollarSign },
  { name: 'Referrals', href: '/dashboard/referrals', icon: UserPlus },
  { name: 'Training', href: '/dashboard/training', icon: GraduationCap },
  { name: 'Events', href: '/dashboard/events', icon: Calendar },
  { name: 'Settings', href: '/dashboard/settings', icon: Settings },
];

export function Sidebar() {
  const pathname = usePathname();
```

```
    return (
      <div className="hidden lg:fixed lg:inset-y-0 lg:flex lg:w-64 lg:flex-col">
        <div className="flex flex-col flex-grow bg-white border-r border-gray-200">
          {/* Logo */}
          <div className="flex items-center h-16 px-6 border-b border-gray-200">
            <img src="/logo.svg" alt="Logo" className="h-8" />
          </div>

          {/* Navigation */}
          <nav className="flex-1 px-3 py-4 space-y-1">
            {navigation.map((item) => {
              const isActive = pathname === item.href;

              return (
                <Link
                  key={item.name}
                  href={item.href}
                  className={cn(
                    "flex items-center gap-3 px-3 py-2 text-sm font-medium rounded-md transition-colors",
                    isActive
                      ? "bg-primary-50 text-primary-700"
                      : "text-gray-700 hover:bg-gray-50 hover:text-primary-600"
                  )}
                >
                  <item.icon className="w-5 h-5" />
                  {item.name}
                </Link>
              );
            })}
          </nav>
        </div>
      </div>
    );
  }
```

6. Functional Modules - Detailed

6.1 Credit Intake & Analysis Module

Purpose

- Pull 3-bureau credit reports via API
- Parse and normalize credit data
- Analyze creditworthiness
- Calculate readiness score
- Route clients to appropriate path

Key Features

1. Consent management and FCRA compliance
2. Integration with iSoftPull (or similar vendors)
3. Fallback PDF upload + AI parsing
4. AI-powered credit analysis
5. Readiness scoring algorithm
6. Automatic path routing

API Endpoints

```
// Credit Endpoints
POST   /api/clients/:id/credit/start        // Initiate credit pull
POST   /api/clients/:id/credit/upload       // Upload PDF (fallback)
GET    /api/clients/:id/credit/latest       // Get latest report
GET    /api/clients/:id/credit/history      // Get all reports
POST   /api/clients/:id/credit/recheck      // Re-run analysis
```

Implementation Notes

Module Structure:

```
apps/api/src/credit/
■■■ credit.module.ts
■■■ credit.controller.ts
■■■ credit.service.ts
■■■ dto/
■   ■■■ start-credit-pull.dto.ts
■   ■■■ upload-pdf.dto.ts
■   ■■■ credit-response.dto.ts
■■■ jobs/
■   ■■■ credit-pull.job.ts
■   ■■■ credit-parse.job.ts
■   ■■■ credit-analysis.job.ts
■■■ vendors/
■   ■■■ credit-vendor.interface.ts
■   ■■■ isoftpull.adapter.ts
■   ■■■ fake-vendor.adapter.ts (for testing)
■■■ utils/
    ■■■ credit-parser.ts
    ■■■ utilization-calculator.ts
    ■■■ readiness-scorer.ts
```

Key Service Methods:

The credit service handles:

- Initiating credit pulls with proper consent
- Managing vendor integrations
- Processing raw credit data
- Running analysis algorithms
- Calculating readiness scores
- Determining client paths (CREDIT_REPAIR vs LENDING)

Frontend Components:

- Credit consent form

- Credit report viewer
- Readiness checklist
- PDF upload interface

6.2 Credit Repair Routing Module

Purpose

- Identify clients who need credit improvement
- Route them to Herman's credit repair team
- Track improvement progress
- Trigger re-analysis when ready

Key Features

1. Automatic routing based on readiness score
2. Webhook integration with external CRM
3. Progress tracking
4. Re-analysis triggers

API Endpoints

```
// Credit Repair Endpoints
POST   /api/clients/:id/referrals/credit-repair    // Create referral
GET    /api/referrals/credit-repair                // List all
PATCH  /api/referrals/:id/update-status           // Update progress
POST   /webhooks/credit-repair-update             // Webhook handler
```

Implementation Notes

When a client scores below fundable threshold:
1. System creates referral record
2. Sends webhook to Herman's CRM with client info
3. Updates client status to "IN_CREDIT_REPAIR"
4. Displays improvement plan in client portal
5. Tracks progress via webhook updates
6. Triggers re-analysis when score improves

6.3 Lending Marketplace Module (Hotwire-Style)

Purpose

- Generate opaque funding offers
- Manage offer selection and fee agreement
- Reveal lender after commitment
- Track applications
- Record funding outcomes

Key Features

1. Hotwire-style opaque offers
2. Fee agreement modal
3. Lender reveal after commitment
4. Application tracking
5. Funding outcome recording

API Endpoints

```
// Offer Endpoints
POST   /api/clients/:id/offers/generate     // Generate offers
GET    /api/clients/:id/offers              // List offers
POST   /api/offers/:id/select               // Select & commit
POST   /api/offers/:id/track-application    // Track app status
POST   /api/offers/:id/record-funding       // Record outcome
GET    /api/offers/:id                      // Get single offer
```

Critical Flow

1. **Generate Offers** - Create 3-5 opaque offers based on credit tier
2. **Display Offers** - Show without lender names
3. **Client Selects** - User clicks "Select This Option"
4. **Fee Agreement** - Modal shows 10% fee, requires explicit agreement
5. **Reveal Lender** - ONLY after agreement, show lender name and application URL
6. **Track Application** - Monitor progress
7. **Record Funding** - Agent enters final outcome and actual loan amount
8. **Calculate Revenue** - Automatic fee calculation (10% platform + 2% lender)

6.4 Admin Dashboard Module

Purpose

- Overview of organization metrics
- Revenue tracking
- Client pipeline visibility
- Transaction management
- Referral oversight

Key Features

1. Real-time metrics dashboard
2. Revenue breakdowns
3. Client funnel visualization
4. Transaction history
5. Export capabilities

API Endpoints

```
// Dashboard Endpoints
GET    /api/admin/summary              // Dashboard overview
GET    /api/admin/revenue              // Revenue breakdown
GET    /api/admin/clients/stats        // Client statistics
GET    /api/admin/transactions         // Transaction list
GET    /api/admin/referrals            // Referral list
GET    /api/admin/export               // Export data
```

Dashboard Metrics

Key Metrics to Display:

- Total clients
- Fundable percentage
- Total funded amount
- Platform fee revenue
- Lender fee revenue
- Pipeline status (analyzing, viewing offers, pending, funded)
- Month-over-month growth
- Average deal size

6.5 White Label Module

Purpose

- Support multiple partner organizations
- Custom branding per partner
- Revenue sharing automation
- Isolated data access

Key Features

1. Partner onboarding
2. Custom branding (logo, colors)
3. Subdomain/custom domain

4. Revenue split configuration
5. Partner dashboard

API Endpoints

```
// Organization Endpoints
POST   /api/organizations              // Create partner org
GET    /api/organizations              // List organizations
GET    /api/organizations/:id          // Get single org
PATCH  /api/organizations/:id          // Update org
GET    /api/organizations/:id/revenue  // Partner revenue
```

Partner Onboarding Flow

1. SUPER_ADMIN creates new organization
2. Sets branding (logo, colors)
3. Configures fee split (e.g., 60/40)
4. Sets up subdomain
5. Creates partner admin user
6. Partner logs in and starts adding clients
7. Revenue automatically split based on configuration

6.6 Referral Program Module

Purpose

- Track affiliate referrals
- Calculate rewards ($400 per funded client)
- Manage payouts
- Generate referral links

Key Features

1. Unique referral links
2. Conversion tracking
3. Payout management
4. Affiliate dashboard

API Endpoints

```
// Referral Endpoints
POST   /api/referrals/generate-link    // Create referral link
GET    /api/referrals/my-stats         // Affiliate stats
GET    /api/referrals/payouts          // Payout history
POST   /api/referrals/:id/mark-paid    // Mark as paid
```

Referral Flow

1. Affiliate gets unique link: `/r/affiliate123`
2. Client signs up through link
3. Referral record created with status "PENDING"
4. Client goes through credit → funding process
5. When loan funds, referral status changes to "PAYABLE"
6. Admin marks referral as "PAID" after payout
7. Affiliate can track their conversions and earnings

6.7 Training & Content Module

Purpose

- Provide educational content
- Promote main course/program
- Increase engagement
- Build authority

Key Features

1. Training content library
2. Video embedding
3. Content categorization
4. Progress tracking (optional)

API Endpoints

```
// Training Endpoints
GET    /api/training/content           // List all content
GET    /api/training/content/:id       // Get single content
POST   /api/training/content           // Create content (admin)
```

Content Types

- Short lessons (text + video)
- Downloadable resources
- Course promotion banners
- Best practices guides

6.8 Events & Webinars Module

Purpose

- Manage events and webinars
- Track registrations
- Promote upcoming events
- Build community

Key Features

1. Event creation and management
2. Registration tracking
3. Calendar view
4. Event types (FREE, PAID)

API Endpoints

```
// Event Endpoints
GET    /api/events                     // List events
GET    /api/events/:id                 // Get single event
POST   /api/events                     // Create event (admin)
PATCH  /api/events/:id                 // Update event
POST   /api/events/:id/register        // Register for event
```

Event Information Stored

- Title and description
- Start/end time
- Event type (FREE/PAID)
- Registration URL (external)
- Max attendees
- Location/Zoom link

6.9 Activity Logging Module

Purpose

- Track all system activities
- Audit trail
- Troubleshooting
- Analytics

Key Features

1. Automatic activity logging

2. Searchable logs
3. Filtered by entity type
4. Organization-scoped

What Gets Logged

- Credit pulls completed
- Offers generated
- Offers selected
- Applications started
- Loans funded
- Referrals created
- User actions

API Endpoints

```
// Activity Endpoints
GET    /api/activity                   // List activities
GET    /api/activity/client/:id        // Client-specific
GET    /api/activity/user/:id          // User-specific
```

END OF PART 2

Part 3 will contain:

- Complete Prisma Schema
- API Integration Requirements
- AI Integration Details
- Frontend Build Plan
- Backend Architecture
- Security & Compliance
- Deployment
- Testing
- Implementation Phases
- FAQ

Credit & Funding Platform - Complete Specification (Part 3 - Final)

Continuation from Part 2

7. Data Model - Complete Prisma Schema

```
// This is your Prisma schema file
```

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// ==========================================
// ORGANIZATIONS & USERS
// ==========================================

model Organization {
  id          String   @id @default(cuid())
  name        String
  slug        String   @unique
  type        OrganizationType @default(PRIMARY)
  parentOrgId String?
  parentOrg   Organization? @relation("WhiteLabelParent", fields: [parentOrgId], references: [id])
  childOrgs   Organization[] @relation("WhiteLabelParent")

  isWhiteLabel Boolean @default(false)
  logoUrl      String?
  primaryColor String? @default("#8faa76")
  feeSplitPct  Float?  // Partner share (e.g., 0.60 = 60%)

  subdomain     String?  @unique
  customDomain  String?  @unique

  // Relationships
  users         User[]
  clients       Client[]
  offers        Offer[]
  transactions  Transaction[]
  referrals     Referral[]
  events        Event[]
  activityLogs  ActivityLog[]
  config        OrganizationConfig?

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([parentOrgId])
  @@index([slug])
}

enum OrganizationType {
  PRIMARY
```

8. API Integration Requirements

8.1 Credit Bureau Integration (iSoftPull)

What You Need

1. **iSoftPull Account**

- Sign up at https://www.isoftpull.com/

- Choose pricing tier (typically starts $5-15 per pull)

- Get API credentials

2. **Required Credentials:**

```
ISOFTPULL_API_KEY="your-api-key"
ISOFTPULL_API_SECRET="your-api-secret"
ISOFTPULL_API_URL="https://api.isoftpull.com/v1"
```

3. **API Integration Example:**

```
// vendors/isoftpull.adapter.ts
export class ISoftPullAdapter implements CreditVendorAdapter {
  private baseUrl: string;
  private apiKey: string;
  private apiSecret: string;

  constructor() {
    this.baseUrl = process.env.ISOFTPULL_API_URL;
    this.apiKey = process.env.ISOFTPULL_API_KEY;
    this.apiSecret = process.env.ISOFTPULL_API_SECRET;
  }

  async pullReport(input: CreditPullInput): Promise<CreditReportRaw> {
    const response = await axios.post(
      `${this.baseUrl}/credit/pull`,
      {
        consumer: {
          firstName: input.firstName,
          lastName: input.lastName,
          ssn: input.ssn,
          dateOfBirth: input.dob,
          address: {
            street: input.address.line1,
            city: input.address.city,
            state: input.address.state,
            zip: input.address.postalCode
          }
        },
        bureaus: ['experian', 'equifax', 'transunion'],
        reportType: 'soft'
      },
      {
        headers: {
          'X-API-Key': this.apiKey,
          'X-API-Secret': this.apiSecret,
          'Content-Type': 'application/json'
        }
      }
    );
```

```
      return this.normalizeResponse(response.data);
    }
  }
```
Alternative Vendors:

- **SoftPullSolutions** - Similar to iSoftPull
- **SmartCredit** - Consumer-focused but has API
- **Equifax API** - Direct bureau integration (complex)

8.2 Lender Integration

What You Need

For each lender partner:

1. **Signed Referral Agreement**

- 2% referral fee
- Terms of partnership
- Payment schedule

2. **Application URL with Tracking**

```
https://lender.com/apply?ref=YOUR_ORG_ID&client=CLIENT_ID
```

3. **Optional: Webhook for Status Updates**

```
// Lender sends webhook when application status changes
POST /webhooks/lender-update
{
  referenceId: "client_456",
  status: "APPROVED", // or "DECLINED", "PENDING"
  loanAmount: 50000,
  apr: 8.5,
  term: 60
}
```

Recommended Lenders

Business Lending:

- **Lendio** - Marketplace aggregator
- **Funding Circle** - SMB term loans
- **OnDeck** - Lines of credit
- **BlueVine** - Working capital

Personal Lending:

- **LendingClub** - Personal loans
- **Prosper** - Personal loans
- **Upgrade** - Personal loans & cards

8.3 Email Service (SendGrid)

```
# Install
pnpm add @sendgrid/mail

# Configure
SENDGRID_API_KEY="SG.your-key"
EMAIL_FROM="noreply@yourplatform.com"
```

Email Templates Needed:

1. Welcome email (client signup)
2. Credit analysis complete
3. Funding options available
4. Application status update
5. Loan funded confirmation
6. Credit repair referral

8.4 File Storage (AWS S3 or Cloudflare R2)

For storing:

- Uploaded credit report PDFs
- Loan documents
- User avatars

```
# Install AWS SDK
pnpm add @aws-sdk/client-s3

# Configure
AWS_ACCESS_KEY_ID="your-key"
AWS_SECRET_ACCESS_KEY="your-secret"
AWS_S3_BUCKET="credit-platform-documents"
AWS_S3_REGION="us-east-1"
```

9. AI Integration Details

9.1 Credit Report Parsing with Claude

Implementation

```
// ai/credit-parser.service.ts
export class CreditParserService {
  private anthropic: Anthropic;

  constructor() {
    this.anthropic = new Anthropic({
      apiKey: process.env.ANTHROPIC_API_KEY
    });
  }

  async parseCreditReport(rawText: string): Promise<ParsedCreditData> {
    const message = await this.anthropic.messages.create({
      model: 'claude-3-5-sonnet-20241022',
      max_tokens: 4096,
      messages: [
        {
          role: 'user',
          content: this.buildParsingPrompt(rawText)
        }
      ]
    });

    const responseText = message.content[0].text;
    const jsonMatch = responseText.match(/\{[\s\S]*\}/);

    if (!jsonMatch) {
      throw new Error('Failed to parse credit report');
    }

    return JSON.parse(jsonMatch[0]);
  }

  private buildParsingPrompt(rawText: string): string {
    return `Extract credit information from this report and return as JSON.

Required format:
{
  "scores": { "experian": 705, "equifax": 710, "transunion": 700 },
  "accounts": [...],
  "inquiries": [...],
  "publicRecords": [],
  "collections": []
}

Report: ${rawText}

Return ONLY valid JSON.`;
  }
}
```

9.2 Generating Recommendations

```
// ai/recommendation.service.ts
export class RecommendationService {
  async generateRecommendations(params: {
    checks: ReadinessChecks;
    metrics: CreditMetrics;
    path: string;
  }): Promise<Recommendations> {
    const prompt = `Based on this credit analysis, provide recommendations:

Score: ${params.metrics.scores.average}
Failed Checks: ${Object.entries(params.checks)
  .filter(([_, c]) => !c.pass)
  .map(([k, c]) => c.message)
```

```
    .join(', ')}

  Provide JSON with: summary, priorities (array), timeline, accountActions`;

      const response = await this.anthropic.messages.create({
        model: 'claude-3-5-sonnet-20241022',
        max_tokens: 2048,
        messages: [{ role: 'user', content: prompt }]
      });

      return this.parseRecommendations(response.content[0].text);
    }
  }
```

10. Frontend UI Build Plan

10.1 Screen Inventory

Authentication (4 screens)

1. `/login` - Login page
2. `/register` - Registration
3. `/forgot-password` - Password reset request
4. `/reset-password/:token` - Password reset form

Agent/Admin Dashboard (13 screens)

5. `/dashboard` - Main dashboard with metrics
6. `/dashboard/clients` - Client list with filters
7. `/dashboard/clients/new` - New client intake form
8. `/dashboard/clients/:id` - Client detail page
9. `/dashboard/clients/:id/credit` - Credit report view
10. `/dashboard/clients/:id/offers` - Funding offers
11. `/dashboard/clients/:id/improvement` - Improvement plan
12. `/dashboard/funding` - Funding pipeline overview
13. `/dashboard/transactions` - Transaction history
14. `/dashboard/referrals` - Referral management
15. `/dashboard/training` - Training content
16. `/dashboard/events` - Events calendar
17. `/dashboard/settings` - Organization settings

Client Portal (7 screens)

18. `/portal` - Client dashboard
19. `/portal/credit` - My credit status
20. `/portal/offers` - My funding options
21. `/portal/improvement` - My improvement plan
22. `/portal/training` - Training resources
23. `/portal/events` - Upcoming events

24. `/portal/profile` - Profile settings

Super Admin (3 screens)

25. `/admin/organizations` - Organization management
26. `/admin/partners` - Partner overview
27. `/admin/analytics` - Platform analytics

10.2 Implementation Order

Phase 1: Foundation (Week 1-2)

- [ ] Setup Next.js + Tailwind + shadcn/ui
- [ ] Create app shell (layout, sidebar, topbar)
- [ ] Build auth pages
- [ ] Setup API client + React Query
- [ ] Create base components

Phase 2: Core Dashboard (Week 3-4)

- [ ] Dashboard home with metrics
- [ ] Client list with table
- [ ] New client form
- [ ] Client detail structure
- [ ] Basic data flow

Phase 3: Credit Features (Week 5-6)

- [ ] Credit consent form
- [ ] Credit report display
- [ ] Readiness checklist
- [ ] Improvement plan view
- [ ] PDF upload

Phase 4: Lending Features (Week 7-8)

- [ ] Funding type selector
- [ ] Offer cards (Hotwire-style)
- [ ] Fee agreement modal
- [ ] Lender reveal
- [ ] Application tracking

Phase 5: Admin & Management (Week 9-10)

- [ ] Transaction management

- [ ] Referral dashboard
- [ ] Revenue analytics
- [ ] Export functionality

Phase 6: Client Portal (Week 11-12)

- [ ] Client dashboard
- [ ] Client views (simplified)
- [ ] Training & events
- [ ] Profile management

10.3 Component Library Structure

```
components/
■■■ ui/                      # shadcn/ui base components
■■■ layout/                  # Layout components
■■■ dashboard/               # Dashboard widgets
■■■ clients/                 # Client management
■■■ credit/                  # Credit features
■■■ funding/                 # Lending features
■■■ common/                  # Reusable components
■■■ forms/                   # Form components
```

11. Backend Architecture

11.1 NestJS Module Structure

```
apps/api/src/
■■■ main.ts
■■■ app.module.ts
■■■ auth/                    # Authentication
■■■ organizations/           # Organization management
■■■ users/                   # User management
■■■ clients/                 # Client management
■■■ credit/                  # Credit reporting
■■■ offers/                  # Lending marketplace
■■■ transactions/            # Revenue tracking
■■■ referrals/               # Referral program
■■■ events/                  # Events & webinars
■■■ admin/                   # Admin endpoints
■■■ webhooks/                # External webhooks
■■■ common/                  # Shared utilities
■■■ config/                  # Configuration
```

11.2 Key Middleware & Guards

Organization Scoping Middleware

```
@Injectable()
export class OrgScopeMiddleware implements NestMiddleware {
  use(req: Request, res: Response, next: NextFunction) {
    const user = req.user;

    if (!user) return next();

    if (user.role === 'SUPER_ADMIN') {
      req.orgScope = {};
    } else {
      req.orgScope = { organizationId: user.organizationId };
    }

    next();
  }
}
```

Roles Guard

```
@Injectable()
export class RolesGuard implements CanActivate {
  constructor(private reflector: Reflector) {}

  canActivate(context: ExecutionContext): boolean {
    const requiredRoles = this.reflector.getAllAndOverride<UserRole[]>(
      'roles',
      [context.getHandler(), context.getClass()]
    );

    if (!requiredRoles) return true;

    const { user } = context.switchToHttp().getRequest();
    return requiredRoles.some((role) => user.role === role);
  }
}
```

12. Security & Compliance

12.1 FCRA Compliance

Key Requirements

1. **Consent** - Explicit consent before pulling credit
2. **Purpose** - Permissible purpose only
3. **Disclosure** - Inform it's a soft pull
4. **Data Security** - Encrypt credit data

5. **Dispute Rights** - Inform consumers

Implementation

```
const consentLanguage = `
By checking this box, I authorize [Company Name] to obtain my consumer
credit report from one or more consumer reporting agencies. I understand
this is a "soft inquiry" and will not affect my credit score.
`;

await prisma.creditReport.create({
  data: {
    consentGivenAt: new Date(),
    consentText: consentLanguage,
  }
});
```

12.2 Data Security

Encryption

```
export class EncryptionService {
  private algorithm = 'aes-256-gcm';
  private key = Buffer.from(process.env.ENCRYPTION_KEY, 'hex');

  encrypt(text: string): string {
    const iv = crypto.randomBytes(16);
    const cipher = crypto.createCipheriv(this.algorithm, this.key, iv);
    let encrypted = cipher.update(text, 'utf8', 'hex');
    encrypted += cipher.final('hex');
    const authTag = cipher.getAuthTag();
    return `${iv.toString('hex')}:${authTag.toString('hex')}:${encrypted}`;
  }

  decrypt(encrypted: string): string {
    const [ivHex, authTagHex, encryptedText] = encrypted.split(':');
    const iv = Buffer.from(ivHex, 'hex');
    const authTag = Buffer.from(authTagHex, 'hex');
    const decipher = crypto.createDecipheriv(this.algorithm, this.key, iv);
    decipher.setAuthTag(authTag);
    let decrypted = decipher.update(encryptedText, 'hex', 'utf8');
    decrypted += decipher.final('utf8');
    return decrypted;
  }
}
```

Password Hashing

```
import * as bcrypt from 'bcrypt';

// Hash
```

```
const hash = await bcrypt.hash(password, 10);

// Verify
const isValid = await bcrypt.compare(password, hash);
```

12.3 Rate Limiting

```
@Module({
  imports: [
    ThrottlerModule.forRoot({
      ttl: 60,
      limit: 10,
    }),
  ],
})
export class AppModule {}

// On sensitive endpoints
@UseGuards(ThrottlerGuard)
@Throttle(3, 60) // 3 requests per minute
async startCreditPull() {}
```

13. Deployment & Infrastructure

13.1 VPS Setup

Initial Server Setup

```
# SSH into server
ssh root@your-server-ip

# Update system
apt update && apt upgrade -y

# Install Node.js 20
curl -fsSL https://deb.nodesource.com/setup_20.x | bash -
apt install -y nodejs

# Install pnpm
npm install -g pnpm

# Install PostgreSQL
apt install -y postgresql postgresql-contrib

# Install Redis
apt install -y redis-server

# Install Nginx
apt install -y nginx
```

```
# Install PM2
npm install -g pm2
```

PostgreSQL Setup

```
sudo -u postgres psql

CREATE DATABASE credit_platform;
CREATE USER platform_user WITH ENCRYPTED PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE credit_platform TO platform_user;
```

Deploy Application

```
# Create directory
mkdir -p /var/www/credit-platform
cd /var/www/credit-platform

# Clone repo
git clone your-repo-url .

# Install & build
pnpm install
pnpm run build

# Setup env
cp .env.example .env
nano .env

# Run migrations
pnpm prisma migrate deploy

# Start with PM2
pm2 start ecosystem.config.js
pm2 save
pm2 startup
```

Nginx Configuration

```
server {
    listen 80;
    server_name api.yourplatform.com;

    location / {
        proxy_pass http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

server {
    listen 80;
    server_name yourplatform.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

SSL Setup

```
certbot --nginx -d yourplatform.com -d api.yourplatform.com
```

13.2 CI/CD with GitHub Actions

```
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Setup Node
        uses: actions/setup-node@v3
        with:
          node-version: '20'

      - name: Install pnpm
        run: npm install -g pnpm

      - name: Install & Build
        run: |
          pnpm install
          pnpm build

      - name: Deploy
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.SERVER_HOST }}
          username: ${{ secrets.SERVER_USER }}
          key: ${{ secrets.SSH_PRIVATE_KEY }}
          script: |
            cd /var/www/credit-platform
            git pull
            pnpm install
            pnpm build
            pm2 restart all
```

14. Testing Strategy

14.1 Backend Testing

```
describe('CreditService', () => {
  let service: CreditService;

  beforeEach(async () => {
    const module = await Test.createTestingModule({
      providers: [CreditService, PrismaService],
    }).compile();

    service = module.get<CreditService>(CreditService);
  });

  it('should create pending credit report', async () => {
    const result = await service.startCreditPull(clientId, {
      ssn: '123-45-6789',
      dob: '1990-01-01',
      consent: true
    });

    expect(result.status).toBe('PENDING');
  });
});
```

14.2 Frontend Testing

```
import { render, screen } from '@testing-library/react';
import { OfferCard } from './offer-card';

describe('OfferCard', () => {
  it('renders offer details', () => {
    render(<OfferCard offer={mockOffer} onSelect={jest.fn()} />);

    expect(screen.getByText('Best Overall')).toBeInTheDocument();
    expect(screen.getByText('$100,000')).toBeInTheDocument();
  });
});
```

15. Implementation Phases

Phase 1: Foundation (Weeks 1-2)

- [ ] Setup monorepo
- [ ] Configure database
- [ ] Build auth system
- [ ] Create app shell
- [ ] Setup deployment

Phase 2: Core Features (Weeks 3-6)

- [ ] Client management
- [ ] Credit integration
- [ ] AI analysis
- [ ] Basic dashboard

Phase 3: Lending Marketplace (Weeks 7-10)

- [ ] Offer generation
- [ ] Hotwire UI
- [ ] Fee agreement
- [ ] Application tracking

Phase 4: Revenue & Admin (Weeks 11-12)

- [ ] Transaction tracking
- [ ] Revenue dashboards
- [ ] Export functionality

Phase 5: White Label (Weeks 13-14)

- [ ] Partner onboarding
- [ ] Custom branding
- [ ] Revenue splits

Phase 6: Polish & Launch (Weeks 15-16)

- [ ] Client portal
- [ ] Training content
- [ ] Testing
- [ ] Production deploy

16. FAQ & Troubleshooting

Q: What if iSoftPull is too expensive?

A: Start with PDF upload + AI parsing. Add iSoftPull later.

Q: How do I handle failed credit pulls?

A: Implement retry logic with exponential backoff. Log and notify agents.

Q: What if lenders don't pay the 2% fee?

A: Track in database, send invoices. Build relationships first.

Q: How do I prevent platform bypass?

A: The Hotwire model solves this - they commit to 10% BEFORE seeing lender.

Q: Can I use this for credit repair only?

A: Yes! Disable lending module, focus on repair path.

Q: How do I onboard first white label partner?

A: Use SUPER_ADMIN panel to create org, set fee split, provide credentials.

END OF COMPLETE SPECIFICATION

This 3-part document contains everything your developers need to build the platform. All sections are covered in detail with code examples, implementation notes, and best practices.