Product Requirements Document: SAINTRIX Beta

1. Introduction

This Product Requirements Document (PRD) outlines the vision, scope, and functional requirements for **SAINTRIX Beta**, a cutting-edge credit repair software designed to empower individuals to take control of their financial health by efficiently identifying, disputing, and resolving inaccuracies on their credit reports. Leveraging a modern technology stack comprising Supabase for the backend, Vercel for deployment, and Cursor AI for accelerated development, SAINTRIX Beta aims to provide a seamless, secure, and highly effective platform for both end-users (clients) and credit repair professionals (administrators).

Credit repair is a critical service that helps consumers improve their credit scores, which in turn unlocks better financial opportunities, including lower interest rates on loans, easier access to housing, and improved financial stability. The current landscape often involves manual, time-consuming processes, and a lack of transparency. SAINTRIX Beta seeks to revolutionize this by automating key aspects of the dispute process, providing clear insights, and fostering efficient communication between clients and administrators.

This document will detail the core functionalities, unique selling propositions, user experience flows, and technical considerations necessary for the successful development and deployment of SAINTRIX Beta. It will serve as the foundational guide for the development team, ensuring all stakeholders are aligned on the product's objectives and features.

2. Product Vision

SAINTRIX Beta envisions a future where credit repair is accessible, transparent, and efficient for everyone. Our vision is to be the leading platform that simplifies the complex credit dispute process, enabling individuals to achieve optimal credit health with minimal effort. We aim to build a trusted ecosystem where clients feel empowered and administrators have all the tools necessary to deliver exceptional service. The software will not only facilitate disputes but also educate users on credit best practices, fostering long-term financial literacy.

3. Target Audience

SAINTRIX Beta serves two primary user groups:

- **Individual Clients:** Consumers with inaccurate or negative items on their credit reports who are looking to improve their credit scores. This includes individuals seeking to qualify for better loans, mortgages, or simply achieve financial stability.
- **Credit Repair Professionals/Agencies:** Businesses or individuals who offer credit repair services to clients. These users require robust tools to manage multiple client cases, automate dispute processes, track progress, and communicate effectively with their clients.

4. Scope

This initial Beta release of SAINTRIX Beta will focus on establishing a strong foundation for core credit repair functionalities, emphasizing security, usability, and automation. Future iterations will expand upon these features, incorporating more advanced analytics, deeper integrations, and broader educational resources.

In-Scope for Beta:

* Secure Client and Admin Authentication (Sign-up, Login, Password Management, MFA).
* Integration with a single major credit data provider for credit report import.
* Client Dashboard for credit score overview and negative item summary.
* Dispute item identification and selection by clients.
* Automated generation of dispute letters (FCRA compliant).
* Secure document upload and management for clients.
* Basic notification system for dispute status updates.
* Admin Dashboard for client management, dispute overview, and letter template management.
* Robust security measures for data protection and privacy.

Out-of-Scope for Beta (Potential Future Enhancements):

* Integration with multiple credit data providers simultaneously.
* Advanced credit score simulation tools.
* Financial literacy modules and educational content.
* Payment processing integration for credit repair services.
* CRM functionalities beyond basic client management.
* Mobile application development.
* Advanced reporting and analytics for administrators.

* Direct communication channels (chat/messaging) between clients and admins within the platform.

5. Functional Requirements

5.1. Core System Requirements

- **Scalability:** The system must be designed to handle a growing number of users and data points without significant performance degradation.
- **Security:** All data, especially sensitive personal and financial information, must be encrypted at rest and in transit. The system must adhere to industry best practices for data privacy and security (e.g., SOC 2, HIPAA, GDPR compliance considerations).
- **Reliability:** The system must be highly available with minimal downtime.
- **Performance:** User interfaces and backend processes must be responsive and efficient.
- **Auditability:** All significant actions (e.g., dispute letter generation, status changes) must be logged for auditing purposes.

5.2. Client Portal Features

5.2.1. User Authentication & Onboarding

- **Secure Registration:** Clients can create an account using email/password. Email verification required.
- **Multi-Factor Authentication (MFA):** Optional but highly recommended MFA setup (e.g., via authenticator app or SMS) for enhanced security.
- **Login/Logout:** Standard secure login and logout functionality.
- **Password Management:** Forgot password and change password functionalities.
- **Terms of Service & Privacy Policy Acceptance:** Mandatory acceptance during registration.

5.2.2. Credit Report Integration & Analysis

- **Credit Data Provider Integration:** Clients can securely link their credit profiles via a chosen third-party API (e.g., Plaid, Finicity, or a specialized credit API). This will be a critical, secure server-side integration.
- **Automated Report Import:** Once linked, the system automatically imports and parses credit reports from Equifax, Experian, and TransUnion.
- **Credit Score Display:** A clear, prominent display of the client's current credit score (e.g., FICO or VantageScore, depending on provider data).

- **Negative Item Identification:** The system automatically highlights and categorizes negative items (e.g., late payments, collections, charge-offs, bankruptcies, inquiries) from the imported reports.
- **Item Details View:** Clients can click on any item to view detailed information (account name, balance, date, creditor, etc.).

5.2.3. Advanced Dispute Management

- **Intuitive Dispute Selection:** Clients can easily select specific negative items they wish to dispute from a categorized list.
- **Dispute Rationale Input:** For each selected item, clients can provide a brief rationale or supporting notes for the dispute (e.g.,

item was paid, identity theft, incorrect balance). This input will inform the dispute letter.
* **Evidence Upload:** Clients can securely upload supporting documents (e.g., bank statements, payment receipts, police reports) directly linked to specific dispute items. These documents will be stored in Supabase Storage with strict access controls.
* **Dispute Letter Preview:** Before generation, clients can preview the auto-generated dispute letter to ensure accuracy and completeness.
* **Dispute Letter Generation & Download:** The system generates FCRA-compliant dispute letters (e.g., PDF format) based on selected items and templates. Clients can download these letters for mailing.
* **Dispute Status Tracking:** A dedicated section where clients can view the real-time status of each dispute (e.g., `Pending Submission`, `Sent to Bureau`, `Response Received`, `Resolved`, `Unresolved`). This status will be updated by administrators.
* **Communication Log:** A chronological log of all actions related to a dispute, including letter generation, status changes, and administrator notes (visible to client).

5.2.4. Advanced Client Portal Features (Unique)

- **Credit Health Gamification & Education:**
  - **Progress Milestones:** Clients earn badges or unlock achievements for actions like importing reports, submitting disputes, or achieving credit score improvements. This encourages engagement.
  - **Personalized Credit Tips:** Based on their credit report analysis, the system provides tailored, actionable advice on how to improve their credit health (e.g., "Consider paying down your credit card balance to below 30% utilization").
  - **

Credit Score Predictor (Basic): A simple tool where clients can input hypothetical actions (e.g., pay off a $500 debt, open a new credit card) and get a directional estimate of how it

might impact their score (with clear disclaimers that this is an estimate).
* **Automated Follow-up Reminders:** The system automatically schedules and sends reminders to clients for follow-up actions, such as mailing dispute letters, checking for responses, or providing additional documentation.
* **Secure Messaging (One-Way from Admin):** Admins can send secure, in-app messages to clients regarding their case, which clients can view but not directly reply to (replies would be via external email/phone for Beta).

5.3. Admin Dashboard Features

5.3.1. Secure Authentication & Access Control

- **Admin Login:** Secure login for administrators.
- **Role-Based Access Control (RBAC):** For future scalability, the system should support different admin roles (e.g., Super Admin, Case Manager) with varying levels of access to client data and functionalities. For Beta, a single 'Admin' role is sufficient.

5.3.2. Client Management

- **Client List:** A searchable and filterable list of all registered clients.
- **Client Profile View:** Detailed view of each client, including contact information, credit report summary, and a history of all interactions and disputes.
- **Client Status Management:** Admins can set and update the status of a client (e.g., `Active`, `On Hold`, `Completed`).

5.3.3. Comprehensive Dispute Management (Centralized Control)

- **Global Dispute Overview:** A dashboard showing all active disputes across all clients, with filters for status, client, and dispute type.
- **Dispute Item Review & Action:** Admins can review client-selected dispute items, add internal notes, and approve/reject items for dispute.
- **Dispute Letter Generation & Mailing Status:** Admins can generate dispute letters on behalf of clients, mark letters as `Mailed`, and track response deadlines.
- **Response Tracking:** Admins can log responses received from credit bureaus or creditors, upload response documents, and update the dispute status accordingly (e.g., `Resolved`, `Unresolved`, `Requires Further Action`).
- **Document Management:** Admins can view and manage all documents uploaded by clients, and upload internal documents related to a client's case.
- **Internal Notes:** Admins can add private, internal notes to client profiles or specific dispute items, visible only to other administrators.

5.3.4. Advanced Admin Portal Features (Unique)

- **Automated Dispute Workflow Engine:**
  - **Configurable Dispute Stages:** Admins can define custom stages for the dispute process (e.g., `Initial Review`, `Letter Sent`, `Bureau Response`, `Follow-up`).
  - **Automated Task Assignment:** Based on dispute status changes, the system can automatically assign follow-up tasks to specific administrators (e.g.,

Review Bureau Response").
  * **Automated Reminders for Admins:** System sends reminders to admins for overdue tasks or upcoming deadlines related to disputes.
* **Smart Letter Template Management:**
  * **Dynamic Placeholders:** Templates support dynamic placeholders (e.g., `{{client_name}}`, `{{dispute_item_account_number}}`) that are automatically populated from the database.
  * **Version Control for Templates:** Admins can create, edit, and version dispute letter templates, ensuring consistency and allowing for A/B testing of different letter strategies.
  * **Conditional Logic:** Basic conditional logic within templates (e.g., if dispute type is 'late payment', include specific paragraph).
* **Integrated Communication Hub (Admin-to-Client):**
  * **Bulk Client Messaging:** Admins can send targeted messages to groups of clients based on their status or dispute progress.
  * **Pre-defined Message Templates:** Create and use templates for common client communications (e.g.,

welcome messages, dispute updates).
* **Audit Trail & Activity Log:** A detailed, immutable log of all actions performed by administrators, including changes to client data, dispute status updates, and letter generations. This is crucial for compliance and accountability.

6. Non-Functional Requirements

6.1. Performance

- **Response Time:** All user-facing actions (page loads, form submissions) should complete within 2 seconds under normal load.
- **Data Processing:** Credit report parsing and dispute letter generation should be near-instantaneous.

6.2. Security

- **Data Encryption:** All sensitive data (PII, credit report data, documents) must be encrypted at rest (Supabase database and storage) and in transit (SSL/TLS).
- **Access Control:** Strict role-based access control (RBAC) for the Admin Portal. Row-Level Security (RLS) in Supabase will be heavily utilized to ensure clients can only access their own data.
- **Vulnerability Management:** Regular security audits and penetration testing (after Beta) to identify and mitigate vulnerabilities.
- **Input Validation:** All user inputs must be rigorously validated to prevent injection attacks (SQL, XSS).
- **Rate Limiting:** Implement rate limiting on API endpoints to prevent abuse and brute-force attacks.

6.3. Scalability

- **Horizontal Scalability:** The architecture should support adding more resources (e.g., Vercel serverless functions, Supabase compute) to handle increased user load.
- **Database Scalability:** Supabase PostgreSQL should be configured for optimal performance and scalability, with proper indexing and query optimization.

6.4. Usability (UX/UI)

- **Intuitive Interface:** Both Client and Admin portals must have clean, intuitive, and easy-to-navigate interfaces.
- **Responsive Design:** The application must be fully responsive and accessible on various devices (desktop, tablet, mobile).
- **Accessibility:** Adherence to WCAG 2.1 AA standards where feasible.

6.5. Reliability & Maintainability

- **Error Handling:** Robust error handling and logging mechanisms for both frontend and backend.
- **Monitoring:** Comprehensive monitoring of application performance, errors, and security events.
- **Code Quality:** Adherence to coding standards, best practices, and thorough code reviews.
- **Automated Testing:** Implementation of unit, integration, and end-to-end tests.

7. Technical Architecture (High-Level)

Frontend: Next.js (React) deployed on Vercel.
Backend: Supabase (PostgreSQL, Auth, Storage, Edge Functions).
Development Environment: Cursor AI.

```
graph TD
  A[Client Browser/Device] -->|HTTPS| B(Vercel Hosted Next.js App)
  B -->|API Calls (Supabase JS SDK)| C(Supabase API Gateway)
  C -->|Auth| D(Supabase Auth)
  C -->|Database Operations| E(Supabase PostgreSQL)
  C -->|File Storage| F(Supabase Storage)
  C -->|Custom Logic| G(Supabase Edge Functions)
  G -->|Secure API Call| H[Third-Party Credit Data API]
  H -->|Data| G
  G -->|Data| E

  I[Admin Browser/Device] -->|HTTPS| J(Vercel Hosted Next.js Admin App)
  J -->|API Calls (Supabase JS SDK)| C

  K[Cursor AI] -->|Development| B
  K -->|Development| G
  K -->|Database Schema Design/Migrations| E
```

8. Development Guide and Implementation Roadmap

This section outlines a phased approach to developing SAINTRIX Beta, leveraging the strengths of Supabase, Vercel, and Cursor AI. Each phase builds upon the previous one, ensuring a stable and functional product at every step.

Phase 1: Foundation & Authentication (Weeks 1-2)

Goal: Establish the core project structure, secure authentication, and basic user profiles.

Key Activities:

1. **Project Setup:**
   * Initialize a new Supabase project. Configure database region and enable extensions as needed.
   * Create a new Next.js project (`create-next-app`) and link it to a Git repository (e.g., GitHub).
   * Connect the Git repository to Vercel for automatic deployments.
   * Configure Supabase environment variables in Vercel (SUPABASE_URL, SUPABASE_ANON_KEY).
   * Set up Cursor AI as the primary development environment.

2. **Supabase Database Schema (Initial):**
   * Define `profiles` table (linked to `auth.users`) for additional user metadata (e.g., `first_name`, `last_name`, `phone_number`).
   * Implement Row-Level Security (RLS) policies for the `profiles` table to ensure users can only access/update their own profile.
3. **Supabase Authentication:**
   * Configure email/password authentication in Supabase.
   * Implement sign-up, login, and password reset flows in the Next.js application using the Supabase JavaScript client library.
   * Add email verification flow.
   * Implement Multi-Factor Authentication (MFA) setup (optional for Beta, but good to plan for).
4. **Client Portal - Basic Dashboard:**
   * Create a protected route in Next.js that only authenticated users can access.
   * Display a welcome message and basic user profile information fetched from Supabase.

Cursor AI Focus:
* Use Cursor AI to generate boilerplate code for Next.js pages, components, and Supabase client interactions.
* Ask Cursor AI for help with Supabase RLS policies and SQL table definitions.
* Utilize Cursor AI for debugging authentication flows.

Phase 2: Credit Report Integration & Display (Weeks 3-5)

Goal: Securely integrate with a third-party credit data provider and display parsed credit report data to the client.

Key Activities:
1. **Third-Party Credit API Research & Selection:**
   * Thoroughly research and select a credit data provider (e.g., Plaid, Finicity, or a specialized credit API like Credit Bureau Connection, CoreLogic Credco).
This is a critical decision and requires careful consideration of cost, data coverage, compliance, and ease of integration.
2. **Supabase Edge Function for Credit API Integration:**
   * Create a Supabase Edge Function (Deno/TypeScript) that acts as a secure proxy to the chosen third-party credit API.
   * This function will receive a request from the frontend (via a Next.js API route), make the authenticated call to the third-party API using your secret API keys (stored securely in Supabase secrets or Vercel environment variables), and return the raw credit report data.

    * Implement robust error handling and logging within the Edge Function.
3. **Database Schema Update:**
    * Define `credit_reports` table to store the imported raw credit report data (consider JSONB column for flexibility).
    * Define `credit_accounts`, `credit_inquiries`, `public_records` tables to store parsed, structured data from the credit report, linked to the `credit_reports` table and `profiles` table.
4. **Next.js API Route for Edge Function Interaction:**
    * Create a Next.js API route that the client-side application will call to initiate the credit report import process. This route will then call the Supabase Edge Function.
5. **Client Portal - Credit Report Import UI:**
    * Develop a user interface for clients to initiate the credit report import (e.g., a button to connect their credit profile).
    * Display loading states and success/error messages.
6. **Client Portal - Credit Report Display:**
    * Parse the imported credit report data and display key information on the client dashboard:
        * Current credit score.
        * Summary of positive accounts.
        * Categorized list of negative items (late payments, collections, etc.).
    * Implement a detailed view for each credit item.

Cursor AI Focus:
* Use Cursor AI to help write the Deno/TypeScript code for the Supabase Edge Function, especially for API request/response handling.
* Ask Cursor AI to generate SQL for parsing and structuring credit report data into your Supabase tables.
* Leverage Cursor AI for building complex React components for displaying credit report data.

Phase 3: Dispute Management & Letter Generation (Weeks 6-9)

Goal: Enable clients to select dispute items, generate FCRA-compliant dispute letters, and track their status.

Key Activities:
1. **Database Schema Update:**
    * Define `dispute_items` table (linked to `credit_accounts` or `credit_inquiries`) to track individual items selected for dispute.
    * Define `dispute_letters` table to store metadata about generated letters (e.g., `user_id`, `dispute_item_id`, `template_id`, `status`, `generated_at`, `sent_at`).

    * Define `documents` table for storing metadata about uploaded evidence, linked to `dispute_items`.
2. **Client Portal - Dispute Selection UI:**
    * Allow clients to select specific negative items from their credit report display for dispute.
    * Provide an input field for clients to add a rationale/notes for each dispute.
    * Implement a file upload component for clients to attach supporting evidence (upload to Supabase Storage).
3. **Supabase Edge Function - Dispute Letter Generation:**
    * Create an Edge Function that takes `dispute_item_ids` and generates a dispute letter.
    * This function will fetch client data, dispute item details, and a letter template from Supabase.
    * Use a library within the Edge Function (e.g., `html-pdf` or similar Deno-compatible PDF generation library) to create a PDF letter.
    * Store the generated PDF in Supabase Storage and save its path in the `dispute_letters` table.
4. **Client Portal - Letter Preview & Download:**
    * Implement a feature to preview the generated letter before finalization.
    * Allow clients to download the generated PDF letter.
5. **Client Portal - Dispute Status Tracking:**
    * Create a dedicated section to display the status of each dispute (e.g., `Pending`, `Generated`, `Sent`, `Resolved`).
    * Display a communication log for each dispute item.

Cursor AI Focus:
* Ask Cursor AI to help with the logic for dispute item selection and state management in React.
* Utilize Cursor AI for writing the PDF generation logic within the Supabase Edge Function.
* Get assistance from Cursor AI for implementing secure file uploads to Supabase Storage.

Phase 4: Admin Dashboard & Workflow Automation (Weeks 10-14)

Goal: Build a comprehensive Admin Dashboard for managing clients, disputes, and automating workflows.

Key Activities:
1. **Admin Authentication & Authorization:**
    * Implement a separate login flow for administrators.
    * Implement basic role-based access control (RBAC) using Supabase RLS or a custom

`is_admin` flag on the `profiles` table.

2. **Admin Dashboard - Client Management:**
    * Create a table view of all clients with search and filter capabilities.
    * Implement a detailed client profile view, showing all their credit reports, dispute items, and uploaded documents.
    * Allow admins to update client status.
3. **Admin Dashboard - Global Dispute Management:**
    * Create a centralized view of all disputes across all clients.
    * Allow admins to review client-submitted dispute items, add internal notes, and approve/reject.
    * Implement functionality for admins to mark letters as `Mailed` and log responses from bureaus.
    * Allow admins to upload internal documents related to a client's case.
4. **Admin Dashboard - Letter Template Management:**
    * Create a UI for admins to create, edit, and manage dispute letter templates.
    * Implement dynamic placeholder support and basic conditional logic for templates.
    * Store templates in a dedicated Supabase table.
5. **Automated Dispute Workflow Engine:**
    * Define `dispute_stages` and `admin_tasks` tables in Supabase.
    * Implement Supabase Database Triggers or Edge Functions to automate task assignment and reminders based on dispute status changes.
    * Create an admin UI to view and manage assigned tasks.
6. **Integrated Communication Hub (Admin-to-Client):**
    * Implement a feature for admins to send one-way secure messages to clients (stored in Supabase, displayed in client portal).
    * Create a UI for managing pre-defined message templates.
7. **Audit Trail:**
    * Implement Supabase Database Triggers or Edge Functions to log all significant admin actions to an `audit_log` table.

Cursor AI Focus:

* Utilize Cursor AI for building complex data tables and forms in the Admin Dashboard.
* Ask Cursor AI for assistance with Supabase Database Triggers and advanced RLS policies for RBAC.
* Leverage Cursor AI for implementing the workflow automation logic within Supabase Edge Functions.

Phase 5: Testing, Refinement & Deployment (Weeks 15-16)

Goal: Ensure the application is stable, secure, and ready for Beta users.

Key Activities:

1. **Comprehensive Testing:**
   * **Unit Tests:** For critical frontend components and Supabase Edge Functions.
   * **Integration Tests:** To verify interactions between frontend, Next.js API routes, and Supabase.
   * **End-to-End Tests:** Simulate user flows for both client and admin portals.
   * **Security Testing:** Basic penetration testing and vulnerability scanning.
2. **Performance Optimization:**
   * Optimize Supabase queries and database indexing.
   * Optimize frontend performance (e.g., lazy loading, image optimization).
3. **Error Handling & Logging:**
   * Refine error messages for clarity.
   * Ensure all critical errors are logged and monitored.
4. **Documentation:**
   * Create user guides for both Client and Admin portals.
   * Document API endpoints and database schema.
5. **Beta Launch Preparation:**
   * Set up analytics (e.g., Vercel Analytics, Google Analytics).
   * Prepare for user feedback collection.

Cursor AI Focus:

* Use Cursor AI to help write test cases and identify performance bottlenecks.
* Ask Cursor AI for suggestions on error handling best practices.

9. Future Enhancements (Post-Beta)

- Integration with additional credit data providers.
- Advanced credit score simulation and forecasting.
- AI-powered credit report analysis for deeper insights and dispute recommendations.
- In-app chat/messaging between clients and administrators.
- Payment processing for credit repair services.
- Mobile application development (React Native).
- Enhanced reporting and analytics for administrators (e.g., dispute success rates, average time to resolution).
- Integration with CRM systems.

10. Glossary

- **API:** Application Programming Interface

- **BaaS:** Backend-as-a-Service
- **FCRA:** Fair Credit Reporting Act
- **MFA:** Multi-Factor Authentication
- **Next.js:** A React framework for building web applications.
- **PRD:** Product Requirements Document
- **RLS:** Row-Level Security
- **Supabase:** An open-source Firebase alternative providing database, authentication, storage, and edge functions.
- **Vercel:** A platform for frontend developers, providing hosting and serverless functions.

11. References

[1] Fair Credit Reporting Act (FCRA): https://www.ftc.gov/legal-library/statutes/fair-credit-reporting-act
[2] Supabase Documentation: https://supabase.com/docs
[3] Vercel Documentation: https://vercel.com/docs
[4] Plaid: https://plaid.com/
[5] Finicity: https://www.finicity.com/

Author: Manus AI
Date: July 31, 2025
Version: 1.0 Beta PRD