Beta Testing & Analytics Suite Guide

For the Replit Development Agent

Version: 1.0
Date: 2025-09-10
Author: Replit Coach Too
Status: Draft

1. Objective

This document provides a detailed technical guide for the Replit Agent to build a comprehensive suite of tools specifically for the **paid beta testing phase**. These features will enable the Avallen Solutions team to efficiently gather user feedback, debug issues, monitor platform health, and collect crucial user analytics.

2. Part 1: In-App Feedback Widget

Objective: To allow users to easily report bugs and suggest features from anywhere within the application.

2.1. Database Enhancement

- New Table: feedback_submissions

Column Name	Data Type	Constraints	Description
id	UUID	PRIMARY KEY	Unique identifier for the submission.
company_id	UUID	FOREIGN KEY (companies.id)	Links to the company providing feedback.
feedback_type	VARCHAR(50)	NOT NULL	'Bug Report' or 'Feature Suggestion'.
message	TEXT	NOT NULL	The user's written feedback.
page_url	VARCHAR(255)		The URL the user was on when they submitted.
submitted_at	TIMESTAMPTZ	DEFAULT now()	Timestamp of the submission.
status	VARCHAR(50)	DEFAULT 'new'	'new', 'in_progress', 'resolved'.

2.2. Backend Implementation

- **New API Endpoint:** POST /api/feedback
    - **Logic:** This endpoint will accept the feedback type, message, and current URL from the user, create a new record in the feedback_submissions table, and send an email notification to the admin team.

2.3. Frontend Implementation

- **UI Component:** FeedbackWidget
    - A persistent, floating button (e.g., with a "Feedback" or "?" icon) visible on all pages for logged-in users.

    - Clicking the button opens a modal with a simple form:
        - Radio buttons for "Report a Bug" or "Suggest a Feature".
        - A text area for the message.
    - **Logic:** On submit, the component automatically captures the current window.location.href and sends all data to the POST /api/feedback endpoint.

3. Part 2: User Impersonation (Super Admin Feature)

Objective: To allow admins to securely log in as a specific user to reproduce bugs and provide better support.

3.1. Backend Implementation

- **New API Endpoint:** POST /api/admin/users/<company_id>/impersonate
    - **Security:** This endpoint must be strictly protected and only accessible to users with the 'admin' role.
    - **Logic:**
        1. The admin requests to impersonate a user.
        2. The backend validates the admin's privileges.
        3. It generates a new, short-lived (e.g., 5-minute) JSON Web Token (JWT) that contains the user_id of the *target user* but also includes an impersonator_id claim with the admin's own user_id.
        4. It returns this temporary token to the admin's browser.

3.2. Frontend Implementation

- **Admin Dashboard (/admin/users):**
    - In the user list table, add a new "Impersonate" button for each user.
    - **Logic:** Clicking this button calls the POST /api/admin/users/<company_id>/impersonate endpoint. On receiving the temporary token, the frontend stores it in local storage and performs a full page reload. The application's authentication logic must be updated to prioritize this temporary token if it exists.
- **Impersonation Banner:**
    - Create a persistent banner component that appears at the top of the screen *only* during an impersonation session.
    - **Content:** "You are currently viewing as [User's Company Name]. Click here to end session."
    - **Logic:** Clicking the banner clears the temporary token from local storage and reloads the page, returning the admin to their own session.

4. Part 3: LCA Calculation Log (Super Admin Feature)

Objective: To provide admins with a real-time view of the status of all LCA calculation jobs.

4.1. Database Enhancement

- New Table: lca_jobs

Column Name	Data Type	Constraints	Description
id	UUID	PRIMARY KEY	Unique identifier for the job.
report_id	UUID	FOREIGN KEY (reports.id)	Links to the report being generated.
status	VARCHAR(50)	NOT NULL	'queued', 'running', 'success', 'failed'.
started_at	TIMESTAMPTZ		Timestamp when the job started processing.
completed_at	TIMESTAMPTZ		Timestamp when the job finished.
duration_seconds	INTEGER		Total time taken for the calculation.
error_message	TEXT	NULLABLE	Stores any error message if the job failed.

4.2. Backend Implementation

- **Update to LCA Calculation Service (Celery Worker):**
  - The Celery worker must be modified to create and update a record in the lca_jobs table throughout its lifecycle:
    1. When the task is first queued, create a record with status: 'queued'.
    2. When the task begins, update the status to 'running' and set started_at.
    3. On successful completion, update the status to 'success', set completed_at, and calculate duration_seconds.
    4. If an error occurs, update the status to 'failed' and save the exception details to error_message.
- **New API Endpoint:** GET /api/admin/lca-jobs
  - Fetches the latest 100 records from the lca_jobs table, ordered by started_at.

4.3. Frontend Implementation

- **New Admin Page (/admin/job-monitor):**
  - A new page in the admin dashboard.
  - **UI:** A table that displays the data from the GET /api/admin/lca-jobs endpoint.
  - **Features:** The table should auto-refresh every 30 seconds and clearly color-code the status of each job (e.g., green for success, red for failed).

5. Part 4: Third-Party Service Integrations

5.1. Error Tracking Service (Sentry)

- **Action:**
  1. Create a new project in Sentry.
  2. Add the Sentry SDK to the Python backend (sentry-sdk[flask]). Configure it to capture all unhandled exceptions.
  3. Add the Sentry SDK to the React frontend (@sentry/react). Configure it to capture all frontend errors.
  4. The Sentry DSN (Data Source Name) key **must** be stored in Replit Secrets.

5.2. Event-Based Analytics (Mixpanel)

- **Action:**
  1. Create a new project in Mixpanel.
  2. Add the Mixpanel SDK to the React frontend (mixpanel-browser).
  3. The Mixpanel Project Token **must** be stored in Replit Secrets.
  4. Implement mixpanel.track() calls for the following key user events:
     - Onboarding Started
     - Onboarding Step Completed (with a property for the step name)
     - Product LCA Created
     - Supplier Invited
     - Report Generated
     - Goal Set
     - Feedback Submitted