Back End Structure

Drinks Sustainability Tool

Version: 1.0
Date: 2025-07-18
Author: Replit Coach Too
Status: Draft

1. Introduction

This document outlines the backend architecture for the 'Drinks Sustainability Tool'. It details the API endpoints, data models, and service logic required to support the features defined in the PRD and the user flows. The backend will be built using Python with the Flask framework.

2. Core Architecture

The backend will be designed using a standard three-tier architecture:

- **API/Controller Layer:** Handles incoming HTTP requests, validates input, and routes them to the appropriate service.
- **Service Layer:** Contains the core business logic. It orchestrates data from the models and calls external services (like OpenLCA or Stripe).
- **Data Access/Model Layer:** Interacts directly with the PostgreSQL database, defining the structure of our data and handling all create, read, update, and delete (CRUD) operations.

3. Data Models (PostgreSQL Schema)

This section defines the core tables in our database.

- **users**
  - id (Primary Key, UUID)
  - replit_user_id (String, Unique) - From Replit Auth
  - email (String, Unique)
  - name (String)
  - created_at (Timestamp)
- **companies**
  - id (Primary Key, UUID)
  - user_id (Foreign Key to users.id)
  - company_name (String)
  - reporting_period_start (Date)
  - reporting_period_end (Date)
  - stripe_customer_id (String, Unique) - For billing

    - subscription_tier (String, e.g., 'free', 'growth')
- **company_data** (Operational Footprint)
    - id (Primary Key, UUID)
    - company_id (Foreign Key to companies.id)
    - data_type (String, e.g., 'electricity', 'diesel', 'waste_landfill')
    - value (Numeric)
    - unit (String, e.g., 'kWh', 'litres', 'kg')
    - reporting_period (Date Range)
- **products** (SKUs)
    - id (Primary Key, UUID)
    - company_id (Foreign Key to companies.id)
    - sku_name (String)
    - product_type (String, e.g., 'Spirit', 'Beer')
- **product_inputs** (Data for a specific product's LCA)
    - id (Primary Key, UUID)
    - product_id (Foreign Key to products.id)
    - input_type (String, e.g., 'raw_material', 'packaging', 'transport')
    - description (String, e.g., 'PET', 'Malted Barley', 'Road Freight')
    - value (Numeric)
    - unit (String, e.g., 'kg', 'litres', 'tonne-km')
- **suppliers**
    - id (Primary Key, UUID)
    - company_id (Foreign Key to companies.id) - The client who invited them
    - supplier_name (String)
    - supplier_email (String, Unique)
    - data_submission_status (String, e.g., 'pending', 'complete')
- **reports** (LCA Reports)
    - id (Primary Key, UUID)
    - company_id (Foreign Key to companies.id)
    - product_id (Foreign Key to products.id, Nullable for company-wide reports)
    - status (String, e.g., 'draft', 'pending_review', 'approved')
    - generated_at (Timestamp)
    - report_data_json (JSONB) - Stores the calculated results from OpenLCA
    - report_pdf_url (String) - Link to the final downloadable PDF

4. API Endpoints (RESTful API)

All endpoints will require authentication via a token from Replit Auth.

Authentication (/auth)

- POST /auth/login: Handled by Replit Auth.
- GET /auth/me: Returns the profile of the currently logged-in user.

Onboarding (/api/onboarding)

- POST /: Creates the initial company profile and operational data.
- PUT /: Updates onboarding data.

Products & LCA Data (/api/products)

- POST /: Creates a new product (SKU).
- POST /<product_id>/inputs: Adds LCA input data for a specific product.
- GET /<product_id>: Retrieves all data for a specific product.

Reports (/api/reports)

- POST /generate: **Asynchronous.** Kicks off the LCA calculation for a company/product. This will add a job to the Celery queue.
  - *Request Body:* { "company_id": "...", "product_id": "..." }
  - *Response:* { "job_id": "...", "message": "Report generation started." }
- GET /<report_id>: Retrieves a specific report's data and status.
- POST /<report_id>/request-review: Changes the report status to pending_review and notifies the internal team.

Suppliers (/api/suppliers)

- POST /invite: Sends an invitation email to a new supplier.
- GET /: Lists all suppliers for the user's company and their status.
- POST /portal/submit: **Public Endpoint.** A dedicated endpoint for the supplier portal to submit data without full authentication (uses a unique, secure token from the invite link).

Billing (/api/billing)

- POST /create-checkout-session: Creates a Stripe Checkout session for a user to subscribe.
- POST /webhook: A public webhook to receive events from Stripe (e.g., payment_succeeded, subscription_updated).

5. Service Logic & Integrations

- **LCA Calculation Service:**
  - This service will be triggered by a Celery background worker.
  - It will fetch all required input data from the database for a given report.
  - It will format the data and make an API call to the server-hosted **OpenLCA** instance.

    - Upon receiving the results from OpenLCA, it will save the calculated footprint to the reports.report_data_json field and update the report's status to draft.
- **Stripe Integration Service:**
    - This service will manage all interactions with the Stripe API, including creating customers, managing subscriptions, and handling webhook events to update a user's subscription status in our database.
- **Notification Service:**
    - A simple service for sending emails (e.g., report ready, supplier invites, review status updates).