Security Checklist

Drinks Sustainability Tool

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

1. Introduction

This document provides a comprehensive security checklist for the 'Drinks Sustainability Tool'. Its purpose is to ensure that security best practices are integrated throughout the development lifecycle, from initial design to deployment and ongoing maintenance. Adherence to this checklist is mandatory to protect user data, maintain platform integrity, and build trust with our clients.

2. Guiding Principles

- **Defense in Depth:** We will implement multiple layers of security controls, so if one layer fails, others are in place to protect the system.
- **Principle of Least Privilege:** Users and system components will only be granted the minimum level of access required to perform their functions.
- **Secure by Default:** The platform will be configured with secure settings out-of-the-box.

3. Pre-Deployment Checklist

This checklist must be completed before any new feature or version is deployed to production.

Category	**Check**	**Status**
Authentication & Access	All user authentication is handled exclusively through **Replit Auth**. No custom password management logic is implemented.	[ ] Done
	API endpoints are protected and require a valid token from Replit Auth, except for designated public endpoints (Stripe webhook, supplier portal submission).	[ ] Done

	The Supplier Portal uses a unique, single-use, and time-limited token in the invitation link to prevent unauthorized access.	[ ] Done
Data Management	All sensitive credentials (**Stripe keys, Database URL, API secrets**) are stored exclusively in **Replit Secrets**. They are NEVER hard-coded.	[ ] Done
	All data is encrypted in transit using HTTPS. Replit Deployments handle this by default.	[ ] Done
	All user and company data is encrypted at rest in the PostgreSQL database.	[ ] Done
	The backend sanitizes all user-provided input (from forms, API calls) to prevent SQL Injection and Cross-Site Scripting (XSS) attacks.	[ ] Done
API & Backend Security	The backend implements Cross-Origin Resource Sharing (CORS) policies to only allow requests from the approved frontend domain.	[ ] Done
	The backend implements rate limiting on sensitive endpoints (e.g., login, report generation) to prevent abuse.	[ ] Done
	The Stripe webhook endpoint verifies the signature of incoming requests to ensure they are genuinely from Stripe.	[ ] Done
Dependency Management	All third-party libraries (both Python and npm) are scanned for known vulnerabilities using	[ ] Done

	tools like pip-audit and npm audit.	
	All identified critical and high-severity vulnerabilities in dependencies are patched before deployment.	[ ] Done

4. Post-Deployment & Ongoing Checks

These checks should be performed regularly on the live production environment.

Category	**Check**	**Frequency**
Monitoring & Logging	Comprehensive logs are generated for all significant events (e.g., user login, report generation, failed API calls, payments).	Continuous
	An alerting system is in place to notify the development team of critical errors or suspicious activity (e.g., multiple failed logins).	Continuous
Data & Backups	The PostgreSQL database is backed up automatically on a regular schedule (e.g., daily).	Daily
	A documented data recovery plan is tested periodically to ensure backups are viable.	Quarterly
User & Access Review	User access levels are reviewed periodically to ensure the principle of least privilege is maintained.	Quarterly
Security Audits	Regular security scans of the live application are performed to identify new vulnerabilities.	Monthly
	A full penetration test by a third-party security firm is	Annually

	scheduled to be performed.	