**Product Requirements Document: Project "SkillnetLabs Label Viewer"**

**Version:** 1.0  
**Date:** October 26, 2023  
**Author:** [Your Name/Team]  
**Status:** Draft

1. Introduction & Overview

**1.1. Product Purpose**  
The purpose of this project, codenamed "Zephyr Viewer," is to create a standalone, self-hosted web application that clones the core functionality of Labelary.com. It will allow developers, system administrators, and end-users to visualize Zebra Programming Language (ZPL) code as a simulated label image without requiring physical Zebra printers. This tool is critical for debugging, testing, and validating ZPL scripts during development and integration processes.

**1.2. Problem Statement**  
Developers working with ZPL often lack access to physical printers for quick tests. Existing online tools like Labelary are excellent but cannot be used in secure, air-gapped, or proprietary environments where external internet access is restricted. There is a need for an internal, controllable tool that provides the same reliable service.

**1.3. Goals & Objectives**

* **Primary Goal:** To perfectly replicate the user experience and core functionality of Labelary.com.
* **User Experience:** Provide a clean, intuitive, and fast interface for converting ZPL to an image.
* **Reliability:** Ensure accurate rendering of ZPL codes as per the specified label dimensions and DPI.
* **Security:** Offer a solution that can be deployed on internal networks, keeping sensitive label data (which may contain barcodes with internal information) within the organization's firewall.

**1.4. Non-Goals**

* To become a full-featured ZPL editor or IDE.
* To support printer management or direct printing (beyond the browser's native print function).
* To implement a commercial SaaS model; this is intended as an internal tool clone.

2. User Personas & Stories

**2.1. User Personas**

* **DevOps Developer (David):** Integrates ZPL generation into backend services. Needs to quickly test and verify the ZPL output from his code before deployment.
* **Warehouse Manager (Wendy):** Not a developer. Receives ZPL code from a system vendor and needs to check what the label will look like before going live.
* **Software Tester (Tina):** Tests applications that generate labels. Needs a reliable way to confirm that the correct ZPL is being produced for various test cases.

**2.2. User Stories**

* **As David,** I want to paste my ZPL code and see an instant preview so that I can debug it quickly.
* **As Wendy,** I want to adjust the label width, height, and DPI using simple inputs to match my physical label stock.
* **As Tina,** I want to download the generated label image as a PNG so that I can attach it to my test reports.
* **As David,** I want a REST API endpoint so that I can integrate the ZPL-to-image conversion directly into my CI/CD pipeline.
* **As All Users,** I want the interface to be simple and uncluttered so that I can focus on the label preview without distractions.

3. Functional Requirements

The application must mirror Labelary.com's feature set.

**3.1. Core ZPL Conversion & Display (MVP)**

* **FR1: ZPL Input Textarea**
  + A large, monospaced font textarea for users to paste or type ZPL code.
  + The textarea must be clearly labeled (e.g., "ZPL Code").
* **FR2: Label Parameter Controls**
  + **Width:** Numeric input field (default: 4 inches) with a dropdown to select units (inches or mm).
  + **Height:** Numeric input field (default: 6 inches) with a dropdown to select units (inches or mm).
  + **DPI (Dots Per Inch):** Dropdown with common DPIs (e.g., 152, 203, 300, 600). Default: 203.
  + **Label Index:** For ZPL with multiple labels, an input to select which label to preview (default: 0).
* **FR3: "View Image" Action**
  + A prominent button (e.g., "View Image") to trigger the conversion.
  + The conversion must happen dynamically without a full page reload (using AJAX/fetch).
* **FR4: Image Preview Panel**
  + A dedicated area to display the generated label image.
  + Must handle images larger than the viewport gracefully (e.g., with scrollbars).
  + Display a loading indicator (spinner) while the image is being generated.
* **FR5: Error Handling**
  + If the ZPL is invalid or the server returns an error, display a clear, user-friendly error message in the preview panel (e.g., "Error: Invalid ZPL code. Please check for syntax errors.").

**3.2. Image & Output Management**

* **FR6: Image Download**
  + Provide a "Download" button below the preview image to save the generated label as a PNG file.
* **FR7: RESTful API**
  + Provide an API endpoint (e.g., POST /v1/convert) that accepts ZPL and parameters and returns an image (PNG).
  + The API must accept application/x-www-form-urlencoded data.
  + The API must be CORS-enabled for cross-origin requests from other internal web apps.

**3.3. Usability & Interface**

* **FR8: Responsive Design**
  + The layout must be usable on desktop and tablet devices. The primary focus is on desktop.
* **FR9: "Look and Feel" Fidelity**
  + The UI must be a close visual clone of Labelary.com: simple, minimal, and functional.
  + Use a similar layout: controls on the left, large preview pane on the right.
  + Use a similar color scheme (whites, grays, blue primary action buttons).

4. Non-Functional Requirements

* **NF1 (Performance):** Image generation and display should feel instantaneous for standard labels (< 2 seconds).
* **NF2 (Accuracy):** The image rendering must be pixel-perfect compared to a real Zebra printer and Labelary's output for the same ZPL and settings.
* **NF3 (Reliability):** The service must have high uptime. The backend process responsible for conversion should be robust and not crash on malformed input.
* **NF4 (Security):** The application itself is low-risk. However, it must be resilient to basic attacks (e.g., SQL injection is not applicable if no DB, but should handle large input payloads gracefully to prevent DoS).
* **NF5 (Deployment):** Must be easy to deploy via Docker or as a simple service on a Linux server.

5. Technical Specifications & Architecture

**5.1. Frontend (Client-Side)**

* **Technology:** Vanilla HTML, CSS, and JavaScript (to maximize compatibility and minimize dependencies). Potentially a minimal framework like Vue.js or React for reactivity, but simplicity is key.
* **Key Tasks:**
  + Render the UI based on the specified HTML/CSS mockup.
  + Capture user input from the form controls.
  + Send a POST request to the backend API with the form data.
  + Display the returned image or error message.
  + Handle the download functionality (using URL.createObjectURL()).

**5.2. Backend (Server-Side) - The Critical Component**

* **Technology:** The core of Labelary is a .NET wrapper around the actual Zebra printer SDK. Our clone has two potential paths:
  + **Path A (Direct .NET Clone):** Use the ZPLUtility .NET library (which is what Labelary likely uses) in a C# ASP.NET Core application. This is the most accurate method.
  + **Path B (Cross-Platform Alternative):** Use a **Node.js** backend with a native addon or by shelling out to a command-line tool that can convert ZPL (e.g., zpl-to-png which may use liblabelary - a C++ library). This is less "pure" but might be easier for some teams.
* **API Endpoint:**
  + POST /api/v1/convert
  + **Expected Parameters:** zpl, width, height, dpmm (or dpi), index.
  + **Response:** Direct image/png stream on success, or a JSON error object on failure.

**5.3. Deployment**

* A Dockerfile should be created to package the chosen backend and serve the static frontend files (e.g., using Nginx for the frontend and proxying API calls to the backend app).
* Example: A multi-stage Docker build for a .NET backend.

6. UI/UX Design & Mockups

*(Since this is a direct clone, detailed mockups are less critical, but a layout guide is essential.)*

**Layout:**

text

+-----------------------------------+----------------------------+

| [ZPL Code Textarea] | [Image Preview Area] |

| | |

| | |

| | |

| | |

| | |

| [Width] [in/mm] [Height] [in/mm] | |

| [DPI Dropdown] [Index] | |

| | |

| [\*\*\*\* View Image Button \*\*\*\*] | [Loading Spinner / Error] |

| [Download PNG Button] (appears | |

| after successful generation) | |

+-----------------------------------+----------------------------+

**Styling Notes:**

* Font: Prefer a system monospace font (e.g., Consolas, Monaco, 'Courier New', monospace) for the ZPL textarea.
* Buttons: Primary action button should be a prominent blue (#007bff or similar).
* Inputs: Light gray borders, subtle rounding.

7. Success Metrics

* **Adoption:** Number of internal teams deploying the tool within 3 months of release.
* **Performance:** 99% of image generation requests complete in under 2 seconds.
* **Accuracy:** 100% pixel-match rate with Labelary for a standardized set of 50+ test ZPL scripts.
* **Reliability:** 99.9% uptime in production deployment.

8. Also include in the Scope

* Saving ZPL scripts or label to file.
* A "gallery" of example labels.
* Advanced features like comparing two ZPL scripts.
* Printing the Label to a local printer on the users PC