**Subject:** A future-proof, cost-effective strategy for our form infrastructure post-Flowable Enterprise.

**Executive Summary**

This document outlines a strategy to address the upcoming non-renewal of our Flowable Enterprise license. Our current library of forms, defined in Flowable's proprietary JSON format, will become unusable without the licensed software, posing a significant risk to business processes that rely on them.

We propose a one-time migration of our existing form definitions to a modern, open-source standard. The recommended solution is to adopt the **formio.js open-source library**, which provides a complete ecosystem for building, rendering, and managing forms.

This approach will not only ensure **100% backward compatibility** for our existing forms but will also **eliminate licensing costs**, prevent future vendor lock-in, and provide a powerful, highly customizable platform that can be securely hosted within our own intranet. The estimated effort is a one-time development project to script the migration, with significant long-term benefits in cost, control, and flexibility.

**1. Problem Statement**

With the decision to not renew the Flowable Enterprise license, we face a critical issue:

* We possess a valuable asset: a large number of business-critical forms defined as Flowable Page Model JSON files.
* This JSON schema is proprietary and requires the licensed Flowable rendering engine to display, edit, or use.
* Upon license expiration, all existing forms will become inoperable, requiring a complete, manual rebuild of every form, leading to significant cost, time investment, and potential disruption to business operations.

**2. Core Objective**

Our goal is to find a solution that allows us to:

1. **Preserve our investment** by salvaging our existing form definitions.
2. **Ensure backward compatibility** so that forms can still be rendered and submitted.
3. **Establish a future-proof platform** for creating and editing forms.
4. **Eliminate recurring license costs** and avoid vendor lock-in.
5. **Maintain full control and security** by hosting the entire solution on our internal network (intranet).

**3. Analysis of Open-Source Form Solutions**

We analyzed several leading open-source solutions that provide a form schema, a visual editor (builder), and a rendering engine. The formio.js library emerged as the strongest candidate.

**Comparison of Leading Open-Source Form Ecosystems:**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | form.io (formio.js) | SurveyJS | JSON Forms |
| **Schema Type** | Proprietary (but open & well-documented) | Proprietary (but open) | **JSON Schema (Open Standard)** |
| **Visual Editor** | **Excellent & Free (MIT License)** | Excellent, but **Paid Commercial License Required** | No mature, official editor |
| **Renderer License** | Free (MIT License) | Free (MIT License) | Free (Apache 2.0 License) |
| **Key Strength** | **Complete, free ecosystem** (builder, renderer, complex components, logic). A direct replacement for enterprise systems. | Highly polished UI/UX and powerful conditional logic. | Strict adherence to open standards, clean separation of data and UI. |
| **Best For** | Organizations seeking a powerful, all-in-one, no-cost form solution with full ownership. | Projects where a superior UI is critical and the editor license fee is acceptable. | Programmatic form generation where a visual builder is not a primary requirement. |
| **Recommendation** | **Highly Recommended** | **Not Recommended** (due to editor cost) | **Not Recommended** (due to lack of editor) |

**4. Proposed Solution: formio.js**

We recommend adopting the free, MIT-licensed formio.js library. It is crucial to distinguish this from the *paid Form.io commercial platform*. We will only use the free library and build our own backend.

**How the formio.js Free Version Helps Us:**

* **Form Builder:** It provides a powerful, visual, drag-and-drop editor that we can embed in our own internal web application.
* **Form Renderer:** It can take a form JSON and render a fully interactive and functional HTML form.
* **JSON Schema:** It uses a well-defined JSON format that can support all our existing form complexities, including layouts, validation, and conditional logic.

This means we get all the necessary front-end tools completely free of charge. Our only cost is the internal development effort to host it and build the necessary backend APIs.

**5. Implementation and Integration Strategy**

**5.1. The Migration Path: Converting Flowable JSON**

The migration from Flowable JSON to formio.js JSON is a medium-complexity task achievable via a one-time script.

* **Process:** We will develop a script (e.g., in Node.js or Python) that reads each Flowable JSON file, transforms it according to a defined mapping, and saves the new formio.js JSON file.
* **Tools:** Open-source tools like jq, JSONata, or native language features can be used to perform the mapping.
* **Expression Buttons:** Complex logic like Flowable's "outcome" buttons can be cleanly migrated. The standard pattern in formio.js is to use a hidden field to store the outcome ('approved', 'rejected') and have buttons that set the value of this field before a final "Submit" button is clicked.

**5.2. Secure Intranet Deployment**

We can host the formio.js builder and renderer securely within our intranet, ensuring no data is ever exposed to the public internet.

* **Architecture:** The solution consists of two parts:
  1. **Frontend App (on ECS):** A lightweight containerized web application that serves the Form Builder UI. This will be placed in a **private VPC subnet** and accessed via an **internal Application Load Balancer (ALB)**.
  2. **Backend API & DB (Our Control):** Our internal team will manage the API and database to save/load form definitions and submissions. These will also reside in the private VPC.
* **Result:** The entire system is firewalled from the internet. Users access the form builder via an internal URL, and all data communication happens exclusively within our secure network.

**5.3. Integration with Corporate Design System (Custom CSS)**

The solution provides full support for our organization's mandated design system and styling.

* **Requirement:** All form controls (text fields, buttons, etc.) must match our corporate branding.
* **Solution:** formio.js offers multiple ways to apply custom CSS. The recommended and most robust method is using the **customClass property**. We can assign our design system's classes (e.g., .ds-input, .ds-button-primary) directly to form components in the JSON definition, ensuring perfect and maintainable style alignment.

**5.4. Advanced Extensibility (Custom React Components)**

For unique requirements not covered by standard controls, we can integrate our own custom React components.

* **Requirement:** Ability to use specialized, internally-developed React components (e.g., an interactive chart, a custom address lookup) within a form.
* **Solution:** While an advanced feature, this is fully supported. It involves creating a JavaScript "bridge" that allows formio.js to render a placeholder element and then uses ReactDOM to mount our custom React component into that element. This gives us unlimited flexibility to extend the platform to meet any future business need.