# **Basel Notification Application Assistant - Product Requirements Document (PRD)**

**Version:** 1.0  
 **Date:** October 18, 2025  
 **Status:** Ready for Development  
 **Developer:** Solo (Richard)  
 **Development Tools:** Claude Code CLI / Web Interface  
 **Timeline:** 2-3 weeks to MVP  
 **Budget:** $0-12/year (domain optional)

## **📋 Executive Summary**

### **Vision Statement**

Transform the Basel Convention hazardous waste notification process from a 2-year learning curve into a 30-minute guided experience through an intelligent, self-service web application.

### **Product Description**

A free, open-source, browser-based "smart form" that helps facilities personnel complete Basel Convention notifications by providing contextual tooltips, reference links, integrated AI assistance, and automated PDF generation—all without requiring backend infrastructure.

### **Primary Goals**

1. ✅ **Reduce completion time** from 2-4 hours to <30 minutes
2. ✅ **Eliminate errors** through intelligent validation and guidance
3. ✅ **Educate users** as they complete the form
4. ✅ **Generate submission-ready PDFs** automatically
5. ✅ **Prevent data loss** with auto-save functionality

### **Success Metrics**

* Form completion rate: >60%
* User satisfaction: >4.0/5.0
* Time to complete: <30 minutes (vs 2-4 hours baseline)
* Error rate: <5% of submissions
* Return user rate: >40%

## **🎯 Product Context**

### **Problem Statement**

**Current Pain Point:** Basel Convention notification forms are overwhelmingly complex (180+ fields, 22 sections, international regulations, specific codes). Non-technical building facilities personnel struggle with:

* No guidance on what information to enter
* Complex regulatory terminology
* Fear of costly errors (delays, penalties)
* 2-4 hours per form completion
* Need to gather information over multiple sessions

**Impact:** Project owner spent nearly 2 years learning to complete forms correctly. Others face same struggle, leading to consultant costs ($500-5,000 per form) or risky DIY attempts.

### **Target Users**

**Primary:** Building facilities personnel (non-technical)

* Waste management coordinators
* Environmental compliance officers
* Facility managers
* Administrative staff

**User Characteristics:**

* Not technical/developers
* Need to work in multiple sessions (information arrives gradually)
* May need to move between form sections freely
* Require contextual help at field level
* Need assurance they're doing it correctly

### **Market Context**

* ~6,000 Basel Convention notifications filed annually worldwide
* 15,000-20,000 potential users globally
* Current solutions: expensive consultants or error-prone manual completion
* No existing specialized smart form tool

## **🏗️ Technical Foundation**

### **Current Assets (70% Complete)**

**Existing HTML Forms:**

1. basel\_form\_Final\_with\_buttons-v1.html (2,877 lines) - **PRIMARY TEMPLATE**
   * All 22 Basel Convention sections implemented
   * 190+ contextual tooltips already coded
   * AI assistant buttons per section
   * Modern responsive design
   * Progress indicators
   * Sample data structure
2. basel\_form\_Final\_filled\_sample.html (2,845 lines)  
   * Pre-populated with realistic data
   * Useful for testing
   * Reference for data format

**What Already Works:**

* ✅ Complete form structure (all 180+ fields)
* ✅ Tooltip system (190+ help texts)
* ✅ AI assistant button UI (expandable panels)
* ✅ Responsive design (mobile-friendly)
* ✅ Form validation framework
* ✅ Visual progress indicators
* ✅ Conditional field logic (disposal vs recovery)
* ✅ Multi-entity support (multiple carriers, generators)

**What Needs to be Built (30%):**

* ❌ Fillable PDF template creation
* ❌ Field mapping (HTML → PDF)
* ❌ PDF generation engine
* ❌ Draft persistence (save/load)
* ❌ AI helper backend (optional enhancement)
* ❌ Final validation before submission
* ❌ Export/import functionality

### **Reference Materials**

* **vCOP8.pdf** - Official Basel Convention form (15 pages)
* **Status:** Flat PDF with NO fillable fields (must be created)
* **190+ pages of architecture documentation** completed

## **🎨 Core Features**

### **Feature 1: Intelligent Field Assistance (Already 90% Complete)**

**User Story:** *"As a facilities coordinator, I need help understanding what to enter in each field so I can complete the form correctly without consulting experts."*

**Requirements:**

#### **1.1 Contextual Tooltips ✅ ALREADY BUILT**

* **What it is:** Help icon (?) next to each field
* **Behavior:** Hover or click shows explanation
* **Content includes:**
  + Field purpose and requirements
  + Examples of correct entries
  + Common mistakes to avoid
  + Regulatory context when relevant

**Example (existing code):**

<div class="field-group">

<label for="exporter\_reg">Registration No:</label>

<input id="exporter\_reg" name="exporter\_reg" required type="text"/>

<div class="help-icon-container">

<span class="help-icon" data-tooltip="exporter\_reg">?</span>

<div class="tooltip" id="tooltip-exporter\_reg">

Enter the official registration number of the exporting

facility as assigned by your national competent authority

</div>

</div>

</div>

**Status:** ✅ Complete - 190+ tooltips already implemented  
 **Action Required:** None - keep as is

#### **1.2 Reference Weblinks 🔧 NEEDS ENHANCEMENT**

**What it is:** Links to official Basel Convention documentation

**Current Status:** Some links exist in tooltips

**Enhancement Needed:** Add systematic reference links for:

* Basel Convention official codes (Y-codes, H-codes, R-codes, D-codes)
* UN shipping classifications
* Country-specific competent authorities
* Packaging standards
* Transport regulations

**Implementation:**

<div class="field-group">

<label for="h\_code">H-Code:</label>

<select id="h\_code" name="h\_code">

<option value="">Select hazard code...</option>

<option value="H1">H1 - Explosive</option>

<option value="H3">H3 - Flammable liquids</option>

</select>

<div class="help-icon-container">

<span class="help-icon">?</span>

<div class="tooltip">

Select the hazard characteristic code(s) that apply.

<br>

<a href="https://www.basel.int/..." target="\_blank">

📖 View official H-code list

</a>

</div>

</div>

</div>

**Acceptance Criteria:**

* [ ] All code selection fields have links to official references
* [ ] Links open in new tab/window
* [ ] Links are to authoritative sources (basel.int, OECD, UN)
* [ ] Links work and are current

**Time Estimate:** 2-3 hours

#### **1.3 AI Assistant Helper 🤖 NEEDS IMPLEMENTATION**

**Current Status:** UI buttons exist but don't function

**What it does:** Section-level AI assistance that provides:

* Overview of section purpose
* Common scenarios and how to handle them
* Validation tips
* Examples of completed sections

**Two Implementation Options:**

##### **Option A: Smart Static Content (Recommended for MVP)**

**How it works:** Pre-written intelligent responses stored in JavaScript, triggered by button click

**Pros:**

* ✅ Free (no API costs)
* ✅ Works offline
* ✅ Fast (instant response)
* ✅ No backend needed
* ✅ Privacy-friendly (no data sent externally)

**Implementation:**

const aiInsights = {

'exporter\_section': {

title: '💡 Understanding Exporter Information',

content: `The exporter is the person/company legally responsible

for the waste shipment. This is usually:

- Your company if shipping directly

- A licensed waste broker if using intermediary

- Must match your country's registration records

Common mistake: Using facility name instead of legal entity name.

Example: "DexMetal Industries Ltd" not "DexMetal Factory 3"`

},

'waste\_codes': {

title: '💡 Selecting the Right Waste Codes',

content: `Basel codes identify your waste type. Most common:

- B1010: Metal wastes (non-hazardous)

- A1180: Electronic waste

- Y12: Wastes from industrial waste treatment

Not sure? Contact your national authority for guidance.

📖 <a href="...">View complete code list</a>`

}

// ... 22 sections total

};

function toggleAIInsight(sectionId) {

const insight = aiInsights[sectionId];

const panel = document.getElementById(`${sectionId}Insight`);

if (panel.style.display === 'none') {

panel.innerHTML = `

<h4>${insight.title}</h4>

<div>${insight.content}</div>

`;

panel.style.display = 'block';

} else {

panel.style.display = 'none';

}

}

**Content Creation:**

* Write 22 section-specific insights (one per form section)
* 100-200 words each
* Include common scenarios, examples, tips
* Add relevant weblinks

**Time Estimate:** 6-8 hours (writing quality content)  
 **Cost:** $0

##### **Option B: Live AI API Integration (Post-MVP Enhancement)**

**How it works:** Call Claude API or OpenAI API with field context

**Pros:**

* ✅ Dynamic responses
* ✅ Can answer specific user questions
* ✅ Continuously improving

**Cons:**

* ❌ Requires API keys ($)
* ❌ Needs internet connection
* ❌ Privacy considerations (data sent to API)
* ❌ Slower response time
* ❌ More complex to implement

**Implementation (simplified):**

async function getAIHelp(sectionId, userQuestion = null) {

const context = getSectionContext(sectionId);

const response = await fetch('https://api.anthropic.com/v1/messages', {

method: 'POST',

headers: {

'x-api-key': API\_KEY,

'content-type': 'application/json',

'anthropic-version': '2023-06-01'

},

body: JSON.stringify({

model: 'claude-3-haiku-20240307', // Cheapest model

max\_tokens: 300,

messages: [{

role: 'user',

content: `Help with Basel Convention form section: ${context}

User question: ${userQuestion || 'General guidance'}`

}]

})

});

return await response.json();

}

**Cost:** ~$0.25 per 1,000 requests (Claude Haiku)  
 **Time Estimate:** 8-12 hours  
 **Decision:** Defer to Phase 2

**Recommendation:** Use Option A for MVP, consider Option B based on user demand

### **Feature 2: PDF Generation & Download 🔧 NEEDS IMPLEMENTATION**

**User Story:** *"As a user, after completing the form, I need a filled PDF that matches the official Basel format so I can submit it to my competent authority."*

**Requirements:**

#### **2.1 Fillable PDF Template Creation (Pre-requisite)**

**Problem:** vCOP8.pdf has NO fillable fields

**Solution:** Use CommonForms (open-source AI tool)

**Process:**

# Step 1: Install CommonForms

pip install commonforms

# Step 2: Generate fillable template

commonforms /path/to/vCOP8.pdf basel\_template\_fillable.pdf \

--model FFDNet-L \

--confidence 0.3 \

--device cpu

# Step 3: Verify results

python3 << 'EOF'

from PyPDF2 import PdfReader

r = PdfReader('basel\_template\_fillable.pdf')

fields = r.get\_fields()

print(f"✅ {len(fields)} fields detected")

for i, name in enumerate(list(fields.keys())[:20]):

print(f"{i+1}. {name}")

EOF

**Expected Output:**

* Fillable PDF with 150-180 detected fields
* Generic field names (Field\_1, Field\_2, Field\_3, etc.)
* 90-95% accuracy

**Time Estimate:** 30 minutes - 2 hours  
 **Cost:** $0 (open source)

**Acceptance Criteria:**

* [ ] Fillable PDF template created
* [ ] >85% of form fields detected
* [ ] PDF opens correctly in Adobe Reader
* [ ] Fields are editable

**Fallback:** If CommonForms detects <85% of fields, use Fiverr ($75-150, 3-5 days)

#### **2.2 Field Mapping Configuration 🔧 NEEDS CREATION**

**What it is:** JSON file mapping HTML field IDs to PDF field names

**Structure:**

{

"version": "1.0",

"template": "basel\_template\_fillable.pdf",

"last\_updated": "2025-10-18",

"mappings": [

{

"htmlField": "exporter\_reg",

"pdfField": "Field\_1",

"pdfPage": 0,

"fieldType": "text",

"required": true,

"maxLength": 50,

"validation": "alphanumeric",

"codeTranslation": null

},

{

"htmlField": "exporter\_name",

"pdfField": "Field\_2",

"pdfPage": 0,

"fieldType": "text",

"required": true,

"maxLength": 100,

"validation": "text",

"codeTranslation": null

},

{

"htmlField": "packaging\_type",

"pdfField": "Field\_87",

"pdfPage": 1,

"fieldType": "select",

"required": false,

"maxLength": 1,

"validation": "code",

"codeTranslation": "packaging\_types"

}

// ... 180+ total mappings

],

"codeTranslations": {

"packaging\_types": {

"drum": "1",

"wooden\_barrel": "2",

"jerrican": "3",

"box": "4",

"bag": "5",

"composite\_packaging": "6",

"pressure\_receptacle": "7",

"bulk": "8",

"other": "9"

},

"transport\_modes": {

"road": "R",

"rail": "T",

"sea": "S",

"air": "A",

"inland\_waterways": "W"

},

"physical\_characteristics": {

"powdery": "1",

"solid": "2",

"viscous": "3",

"sludgy": "4",

"liquid": "5",

"gaseous": "6",

"other": "7"

}

},

"conditionalFields": {

"disposal\_operation": {

"condition": "operation\_type === 'disposal'",

"fields": ["d\_code", "disposal\_facility\_reg"]

},

"recovery\_operation": {

"condition": "operation\_type === 'recovery'",

"fields": ["r\_code", "recovery\_facility\_reg"]

}

}

}

**Creation Process:**

1. Extract PDF field names (from CommonForms output)
2. Create spreadsheet: HTML Field ID | PDF Field Name | Notes
3. Map all 180+ fields manually
4. Convert to JSON
5. Test with sample data

**Time Estimate:** 6-8 hours  
 **Cost:** $0

**Acceptance Criteria:**

* [ ] All HTML fields mapped to PDF fields
* [ ] Code translations included
* [ ] Conditional logic specified
* [ ] JSON validates correctly
* [ ] Documentation of special cases

#### **2.3 PDF Generation Engine 🔧 NEEDS IMPLEMENTATION**

**Library:** pdf-lib v1.17+ (free, MIT license)

**Implementation:**

**File:** src/js/pdf-mapper.js

/\*\*

\* Basel PDF Generator

\* Fills official Basel Convention PDF with user form data

\*/

class BaselPDFMapper {

constructor() {

this.mappings = null;

this.templateBytes = null;

this.initialized = false;

}

/\*\*

\* Initialize - load configuration and template

\*/

async initialize() {

console.log('🚀 Initializing PDF mapper...');

try {

// Load field mappings

const mappingsResponse = await fetch('src/data/field-mappings.json');

this.mappings = await mappingsResponse.json();

console.log(`✅ Loaded ${this.mappings.mappings.length} field mappings`);

// Load PDF template

const pdfResponse = await fetch('public/basel\_template\_fillable.pdf');

this.templateBytes = await pdfResponse.arrayBuffer();

console.log('✅ PDF template loaded');

this.initialized = true;

return true;

} catch (error) {

console.error('❌ Initialization failed:', error);

throw new Error('Failed to initialize PDF mapper: ' + error.message);

}

}

/\*\*

\* Generate filled PDF

\*/

async generatePDF(formData) {

if (!this.initialized) {

throw new Error('Mapper not initialized. Call initialize() first.');

}

console.log('📄 Generating PDF...');

// Load PDF document

const { PDFDocument } = PDFLib;

const pdfDoc = await PDFDocument.load(this.templateBytes);

const form = pdfDoc.getForm();

let successCount = 0;

let errorCount = 0;

const errors = [];

// Fill each mapped field

for (const mapping of this.mappings.mappings) {

try {

const success = this.fillField(form, formData, mapping);

if (success) successCount++;

} catch (error) {

errorCount++;

errors.push({

field: mapping.htmlField,

error: error.message

});

console.warn(`⚠️ Error filling ${mapping.htmlField}:`, error);

}

}

console.log(`✅ Filled ${successCount} fields`);

if (errorCount > 0) {

console.warn(`⚠️ ${errorCount} fields had errors`);

}

// Handle conditional fields

this.fillConditionalFields(form, formData);

// Generate PDF bytes

const pdfBytes = await pdfDoc.save();

console.log('✅ PDF generation complete');

return {

pdfBytes,

stats: {

total: this.mappings.mappings.length,

success: successCount,

errors: errorCount,

errorDetails: errors

}

};

}

/\*\*

\* Fill individual field

\*/

fillField(form, formData, mapping) {

const value = formData[mapping.htmlField];

// Skip empty optional fields

if (!value && !mapping.required) {

return false;

}

// Error if required field missing

if (!value && mapping.required) {

throw new Error(`Required field ${mapping.htmlField} is empty`);

}

try {

let pdfField;

// Handle different field types

if (mapping.fieldType === 'checkbox') {

pdfField = form.getCheckBox(mapping.pdfField);

if (value === true || value === 'true' || value === 'on') {

pdfField.check();

}

} else {

// Text field

pdfField = form.getTextField(mapping.pdfField);

// Apply code translation if needed

let finalValue = value;

if (mapping.codeTranslation) {

finalValue = this.translateCode(

mapping.codeTranslation,

value

);

}

// Apply max length

if (mapping.maxLength) {

finalValue = this.truncate(finalValue, mapping.maxLength);

}

pdfField.setText(String(finalValue));

}

return true;

} catch (error) {

console.warn(`PDF field not found: ${mapping.pdfField}`);

return false;

}

}

/\*\*

\* Translate HTML value to PDF code

\*/

translateCode(translationKey, value) {

const translations = this.mappings.codeTranslations[translationKey];

if (!translations) {

return value;

}

return translations[value] || value;

}

/\*\*

\* Truncate text to max length

\*/

truncate(text, maxLength) {

if (!maxLength || text.length <= maxLength) {

return text;

}

return text.substring(0, maxLength - 3) + '...';

}

/\*\*

\* Handle conditional fields

\*/

fillConditionalFields(form, formData) {

for (const [name, condition] of Object.entries(this.mappings.conditionalFields)) {

if (this.evaluateCondition(condition.condition, formData)) {

console.log(`✅ Conditional group ${name} is active`);

}

}

}

/\*\*

\* Evaluate condition (simple implementation)

\*/

evaluateCondition(conditionStr, formData) {

try {

// Replace variable names with formData values

const code = conditionStr.replace(/(\w+)/g, (match) => {

return `formData['${match}']`;

});

return eval(code);

} catch (error) {

console.error('Condition evaluation error:', error);

return false;

}

}

}

// Global instance

window.pdfMapper = new BaselPDFMapper();

**Usage in main app:**

// Initialize on page load

document.addEventListener('DOMContentLoaded', async () => {

await window.pdfMapper.initialize();

});

// Generate PDF button

async function generatePDF() {

// Show loading

showLoading('Generating your PDF...');

try {

// Collect form data

const formData = collectFormData();

// Validate

const validation = validateForm(formData);

if (!validation.valid) {

alert('Please fill in all required fields:\n' +

validation.missing.join('\n'));

return;

}

// Generate PDF

const result = await window.pdfMapper.generatePDF(formData);

// Download

downloadPDF(

result.pdfBytes,

`basel\_notification\_${new Date().toISOString().split('T')[0]}.pdf`

);

// Show success

alert('✅ PDF generated successfully!\n' +

`${result.stats.success} fields filled`);

} catch (error) {

console.error('PDF generation failed:', error);

alert('Failed to generate PDF. Please check console for details.');

} finally {

hideLoading();

}

}

function downloadPDF(pdfBytes, fileName) {

const blob = new Blob([pdfBytes], { type: 'application/pdf' });

const url = URL.createObjectURL(blob);

const link = document.createElement('a');

link.href = url;

link.download = fileName;

link.click();

setTimeout(() => URL.revokeObjectURL(url), 100);

}

**Time Estimate:** 8-12 hours  
 **Cost:** $0 (pdf-lib is free)

**Acceptance Criteria:**

* [ ] PDF generates without errors
* [ ] All mapped fields populate correctly
* [ ] Code translations work
* [ ] Conditional fields handled properly
* [ ] Generated PDF opens in Adobe Reader
* [ ] File downloads with correct name
* [ ] Loading indicator shows during generation
* [ ] Success/error messages display

### **Feature 3: Draft Persistence 🔧 NEEDS IMPLEMENTATION**

**User Story:** *"As a user, I need to save my progress and return later because I don't have all information available at once."*

**Requirements:**

#### **3.1 Auto-Save to Browser Storage**

**Implementation:**

**File:** src/js/storage.js

/\*\*

\* Basel Form Storage Manager

\* Handles draft saving/loading using localStorage

\*/

class BaselFormStorage {

constructor() {

this.storageKey = 'basel\_form\_draft\_v1';

this.autoSaveInterval = 30000; // 30 seconds

this.autoSaveTimer = null;

}

/\*\*

\* Save form data

\*/

save(formData, options = {}) {

const draft = {

version: '1.0',

data: formData,

timestamp: new Date().toISOString(),

metadata: {

completionPercentage: this.calculateCompletion(formData),

totalFields: Object.keys(formData).length,

requiredFieldsFilled: this.countRequiredFieldsFilled(formData)

}

};

try {

const dataStr = JSON.stringify(draft);

const sizeKB = new Blob([dataStr]).size / 1024;

console.log(`💾 Saving draft (${sizeKB.toFixed(2)} KB)...`);

localStorage.setItem(this.storageKey, dataStr);

if (!options.silent) {

this.showNotification('Draft saved ✓', 'success');

}

this.updateDraftInfo(draft);

return true;

} catch (error) {

if (error.name === 'QuotaExceededError') {

this.showNotification(

'⚠️ Storage full. Please export your draft.',

'error'

);

} else {

console.error('Save error:', error);

this.showNotification('Failed to save draft', 'error');

}

return false;

}

}

/\*\*

\* Load draft

\*/

load() {

try {

const stored = localStorage.getItem(this.storageKey);

if (!stored) {

console.log('No draft found');

return null;

}

const draft = JSON.parse(stored);

// Version check

if (draft.version !== '1.0') {

if (!confirm('Draft is from different version. Load anyway?')) {

return null;

}

}

console.log(`📂 Loaded draft from ${draft.timestamp}`);

this.updateDraftInfo(draft);

return draft;

} catch (error) {

console.error('Load error:', error);

return null;

}

}

/\*\*

\* Populate form with data

\*/

populateForm(data) {

let populated = 0;

Object.keys(data).forEach(fieldId => {

const field = document.getElementById(fieldId);

if (!field) return;

if (field.type === 'checkbox') {

field.checked = data[fieldId] === true || data[fieldId] === 'true';

} else if (field.type === 'radio') {

if (field.value === data[fieldId]) {

field.checked = true;

}

} else {

field.value = data[fieldId];

}

populated++;

});

console.log(`✅ Populated ${populated} fields`);

this.showNotification(`Loaded ${populated} fields`, 'success');

}

/\*\*

\* Enable auto-save

\*/

enableAutoSave() {

this.disableAutoSave();

this.autoSaveTimer = setInterval(() => {

const data = this.collectFormData();

this.save(data, { silent: true });

console.log('🔄 Auto-saved at', new Date().toLocaleTimeString());

}, this.autoSaveInterval);

console.log('🔄 Auto-save enabled (every 30 seconds)');

}

/\*\*

\* Disable auto-save

\*/

disableAutoSave() {

if (this.autoSaveTimer) {

clearInterval(this.autoSaveTimer);

this.autoSaveTimer = null;

}

}

/\*\*

\* Export draft as JSON file

\*/

exportDraft() {

const draft = this.load();

if (!draft) {

this.showNotification('No draft to export', 'warning');

return;

}

const dataStr = JSON.stringify(draft, null, 2);

const blob = new Blob([dataStr], { type: 'application/json' });

const url = URL.createObjectURL(blob);

const link = document.createElement('a');

link.href = url;

link.download = `basel\_draft\_${new Date().toISOString().split('T')[0]}.json`;

link.click();

URL.revokeObjectURL(url);

this.showNotification('Draft exported ✓', 'success');

}

/\*\*

\* Import draft from JSON file

\*/

importDraft(file) {

const reader = new FileReader();

reader.onload = (e) => {

try {

const draft = JSON.parse(e.target.result);

if (!draft.data || !draft.version) {

throw new Error('Invalid draft file');

}

localStorage.setItem(this.storageKey, JSON.stringify(draft));

this.populateForm(draft.data);

this.showNotification('Draft imported ✓', 'success');

} catch (error) {

console.error('Import error:', error);

this.showNotification('Invalid draft file', 'error');

}

};

reader.readAsText(file);

}

/\*\*

\* Clear saved draft

\*/

clear() {

if (!confirm('Delete saved draft?')) {

return false;

}

localStorage.removeItem(this.storageKey);

this.showNotification('Draft deleted', 'info');

this.updateDraftInfo(null);

return true;

}

/\*\*

\* Collect current form data

\*/

collectFormData() {

const form = document.getElementById('baselForm');

const formData = new FormData(form);

const data = {};

for (const [key, value] of formData.entries()) {

data[key] = value;

}

// Include unchecked checkboxes

const checkboxes = form.querySelectorAll('input[type="checkbox"]');

checkboxes.forEach(cb => {

if (!cb.checked && cb.id) {

data[cb.id] = false;

}

});

return data;

}

/\*\*

\* Calculate completion percentage

\*/

calculateCompletion(formData) {

const requiredFields = this.getRequiredFieldIds();

if (requiredFields.length === 0) return 100;

const filled = requiredFields.filter(fieldId => {

const value = formData[fieldId];

return value && value.toString().trim().length > 0;

}).length;

return Math.round((filled / requiredFields.length) \* 100);

}

/\*\*

\* Count required fields filled

\*/

countRequiredFieldsFilled(formData) {

const requiredFields = this.getRequiredFieldIds();

return requiredFields.filter(fieldId => {

const value = formData[fieldId];

return value && value.toString().trim().length > 0;

}).length;

}

/\*\*

\* Get required field IDs

\*/

getRequiredFieldIds() {

const fields = document.querySelectorAll('[required]');

return Array.from(fields).map(f => f.id).filter(id => id);

}

/\*\*

\* Update draft info display

\*/

updateDraftInfo(draft) {

const infoEl = document.getElementById('draftInfo');

if (!infoEl) return;

if (!draft) {

infoEl.innerHTML = 'No draft saved';

infoEl.className = 'draft-info';

return;

}

const date = new Date(draft.timestamp);

const timeAgo = this.getTimeAgo(date);

infoEl.innerHTML = `

<span class="draft-icon">💾</span>

Saved ${timeAgo}

<span class="completion">${draft.metadata.completionPercentage}% complete</span>

`;

infoEl.className = 'draft-info active';

}

/\*\*

\* Get human-readable time ago

\*/

getTimeAgo(date) {

const seconds = Math.floor((new Date() - date) / 1000);

if (seconds < 60) return 'just now';

if (seconds < 3600) return `${Math.floor(seconds / 60)}min ago`;

if (seconds < 86400) return `${Math.floor(seconds / 3600)}hr ago`;

return date.toLocaleDateString();

}

/\*\*

\* Show notification (simple implementation)

\*/

showNotification(message, type = 'info') {

console.log(`[${type.toUpperCase()}] ${message}`);

// Create toast notification

const toast = document.createElement('div');

toast.className = `toast toast-${type}`;

toast.textContent = message;

document.body.appendChild(toast);

setTimeout(() => toast.classList.add('show'), 10);

setTimeout(() => {

toast.classList.remove('show');

setTimeout(() => toast.remove(), 300);

}, 3000);

}

}

// Global instance

window.storage = new BaselFormStorage();

**CSS for toast notifications:**

.toast {

position: fixed;

bottom: 20px;

right: 20px;

padding: 12px 24px;

border-radius: 8px;

background: #333;

color: white;

font-size: 14px;

opacity: 0;

transform: translateY(20px);

transition: all 0.3s ease;

z-index: 10000;

box-shadow: 0 4px 12px rgba(0,0,0,0.3);

}

.toast.show {

opacity: 1;

transform: translateY(0);

}

.toast-success {

background: #10b981;

}

.toast-error {

background: #ef4444;

}

.toast-warning {

background: #f59e0b;

}

.toast-info {

background: #3b82f6;

}

.draft-info {

text-align: center;

padding: 10px;

background: #f3f4f6;

border-radius: 8px;

margin-bottom: 20px;

color: #6b7280;

}

.draft-info.active {

background: #d1fae5;

color: #065f46;

}

.draft-icon {

margin-right: 8px;

}

.completion {

margin-left: 12px;

font-weight: bold;

color: #059669;

}

**Time Estimate:** 4-6 hours  
 **Cost:** $0

**Acceptance Criteria:**

* [ ] Auto-saves every 30 seconds
* [ ] Manual save button works
* [ ] Draft loads on page refresh
* [ ] Progress percentage displays
* [ ] Export/import functionality works
* [ ] Toast notifications display
* [ ] Works across browser sessions
* [ ] Handles localStorage quota exceeded error

### **Feature 4: Comprehensive Validation 🔧 NEEDS ENHANCEMENT**

**User Story:** *"As a user, I need to know if I've completed everything correctly before generating the final PDF so I don't submit an incomplete or incorrect form."*

**Current Status:** Basic HTML5 validation exists

**Enhancement Needed:**

**File:** src/js/validation.js

/\*\*

\* Basel Form Validation

\* Comprehensive pre-submission validation

\*/

class BaselFormValidator {

constructor() {

this.errors = [];

this.warnings = [];

}

/\*\*

\* Validate complete form

\*/

validate(formData) {

this.errors = [];

this.warnings = [];

// Required fields check

this.validateRequiredFields(formData);

// Business logic validation

this.validateBusinessRules(formData);

// Data format validation

this.validateDataFormats(formData);

// Consistency checks

this.validateConsistency(formData);

return {

valid: this.errors.length === 0,

errors: this.errors,

warnings: this.warnings

};

}

/\*\*

\* Validate required fields

\*/

validateRequiredFields(formData) {

const requiredFields = this.getRequiredFields();

requiredFields.forEach(field => {

const value = formData[field.id];

if (!value || value.toString().trim().length === 0) {

this.errors.push({

field: field.id,

label: field.label,

message: `${field.label} is required`

});

}

});

}

/\*\*

\* Validate business rules

\*/

validateBusinessRules(formData) {

// Disposal XOR Recovery must be selected

const hasDisposal = formData.operation\_type === 'disposal';

const hasRecovery = formData.operation\_type === 'recovery';

if (!hasDisposal && !hasRecovery) {

this.errors.push({

field: 'operation\_type',

message: 'Must select either Disposal or Recovery operation'

});

}

if (hasDisposal && hasRecovery) {

this.errors.push({

field: 'operation\_type',

message: 'Cannot select both Disposal and Recovery'

});

}

// If disposal selected, D-code required

if (hasDisposal && !formData.d\_code) {

this.errors.push({

field: 'd\_code',

message: 'D-code required for disposal operations'

});

}

// If recovery selected, R-code required

if (hasRecovery && !formData.r\_code) {

this.errors.push({

field: 'r\_code',

message: 'R-code required for recovery operations'

});

}

// Date validation

const firstDeparture = new Date(formData.first\_departure);

const lastDeparture = new Date(formData.last\_departure);

if (lastDeparture < firstDeparture) {

this.errors.push({

field: 'last\_departure',

message: 'Last departure date must be after first departure'

});

}

// Quantity validation

if (!formData.quantity\_tonnes && !formData.quantity\_cubic) {

this.errors.push({

field: 'quantity\_tonnes',

message: 'Must provide either tonnage or volume'

});

}

}

/\*\*

\* Validate data formats

\*/

validateDataFormats(formData) {

// Email validation

if (formData.exporter\_email) {

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!emailRegex.test(formData.exporter\_email)) {

this.errors.push({

field: 'exporter\_email',

message: 'Invalid email format'

});

}

}

// Phone validation

if (formData.exporter\_phone) {

const phoneRegex = /^\+?[\d\s\-\(\)]+$/;

if (!phoneRegex.test(formData.exporter\_phone)) {

this.warnings.push({

field: 'exporter\_phone',

message: 'Phone number format may be invalid'

});

}

}

// Registration number format

if (formData.exporter\_reg) {

if (formData.exporter\_reg.length < 5) {

this.warnings.push({

field: 'exporter\_reg',

message: 'Registration number seems unusually short'

});

}

}

}

/\*\*

\* Validate consistency

\*/

validateConsistency(formData) {

// If importer same as facility, names should match

if (formData.importer\_name && formData.recovery\_facility\_name) {

// This is a warning, not error

if (formData.importer\_name !== formData.recovery\_facility\_name) {

this.warnings.push({

field: 'recovery\_facility\_name',

message: 'Facility name differs from importer name. Is this correct?'

});

}

}

}

/\*\*

\* Get required fields with labels

\*/

getRequiredFields() {

const fields = [];

document.querySelectorAll('[required]').forEach(el => {

const label = el.previousElementSibling?.textContent || el.id;

fields.push({

id: el.id,

label: label.replace(':', '').trim()

});

});

return fields;

}

/\*\*

\* Show validation results

\*/

showValidationResults(results) {

if (results.valid) {

alert('✅ All validations passed!\nReady to generate PDF.');

return true;

}

let message = '⚠️ Please fix the following issues:\n\n';

results.errors.forEach((err, i) => {

message += `${i + 1}. ${err.message}\n`;

});

if (results.warnings.length > 0) {

message += '\n⚠️ Warnings:\n\n';

results.warnings.forEach((warn, i) => {

message += `${i + 1}. ${warn.message}\n`;

});

}

alert(message);

// Focus first error field

if (results.errors.length > 0) {

const firstErrorField = document.getElementById(results.errors[0].field);

if (firstErrorField) {

firstErrorField.focus();

firstErrorField.scrollIntoView({ behavior: 'smooth', block: 'center' });

}

}

return false;

}

}

// Global instance

window.validator = new BaselFormValidator();

**Usage:**

async function generatePDF() {

// Collect form data

const formData = window.storage.collectFormData();

// Validate

const validation = window.validator.validate(formData);

// Show results

const canProceed = window.validator.showValidationResults(validation);

if (!canProceed) {

return; // Stop if validation failed

}

// Proceed with PDF generation

// ... (rest of generation code)

}

**Time Estimate:** 3-4 hours  
 **Cost:** $0

**Acceptance Criteria:**

* [ ] All required fields validated
* [ ] Business rules enforced
* [ ] Data formats checked
* [ ] Consistency rules applied
* [ ] Clear error messages displayed
* [ ] Focus moves to first error
* [ ] Warnings don't block submission
* [ ] Validation runs before PDF generation

## **🏗️ System Architecture**

### **Architecture Diagram**

┌────────────────────────────────────────────────────────────┐

│ USER'S BROWSER │

│ │

│ ┌──────────────────────────────────────────────────┐ │

│ │ HTML Form UI (basel\_form\_Final\_with\_buttons.html)│ │

│ │ - 180+ fields │ │

│ │ - 190+ tooltips │ │

│ │ - AI assistant panels │ │

│ │ - Progress indicators │ │

│ └────────────┬──────────────────────┬────────────────┘ │

│ │ │ │

│ ┌────────────▼──────────┐ ┌────────▼─────────────────┐ │

│ │ JavaScript Layer │ │ Data Layer │ │

│ │ ┌─────────────────┐ │ │ ┌──────────────────┐ │ │

│ │ │ app.js │ │ │ │ localStorage │ │ │

│ │ │ - Main logic │ │ │ │ - Draft data │ │ │

│ │ │ - Event handlers│ │ │ │ - Auto-save │ │ │

│ │ └─────────────────┘ │ │ └──────────────────┘ │ │

│ │ ┌─────────────────┐ │ │ │ │

│ │ │ pdf-mapper.js │ │ │ ┌──────────────────┐ │ │

│ │ │ - PDF generation│◄─┼──┼──│ field-mappings │ │ │

│ │ │ - Field mapping │ │ │ │ .json │ │ │

│ │ └─────────────────┘ │ │ └──────────────────┘ │ │

│ │ ┌─────────────────┐ │ │ │ │

│ │ │ storage.js │ │ │ ┌──────────────────┐ │ │

│ │ │ - Save/load │◄─┼──┼──│ basel\_template │ │ │

│ │ │ - Export/import │ │ │ │ \_fillable.pdf │ │ │

│ │ └─────────────────┘ │ │ └──────────────────┘ │ │

│ │ ┌─────────────────┐ │ │ │ │

│ │ │ validation.js │ │ │ │ │

│ │ │ - Form validation│ │ │ │ │

│ │ └─────────────────┘ │ │ │ │

│ └───────────────────────┘ └──────────────────────────┘ │

│ │

│ ┌──────────────────────────────────────────────────┐ │

│ │ External Libraries (CDN) │ │

│ │ - pdf-lib v1.17+ (PDF manipulation) │ │

│ └──────────────────────────────────────────────────┘ │

│ │

│ ┌──────────────────────────────────────────────────┐ │

│ │ Outputs │ │

│ │ - Filled PDF download │ │

│ │ - Draft JSON export │ │

│ └──────────────────────────────────────────────────┘ │

└────────────────────────────────────────────────────────────┘

┌────────────────────────────────────────────────────────────┐

│ HOSTING (GitHub Pages - Free) │

│ - Static files only │

│ - No server required │

│ - HTTPS included │

│ - Custom domain support │

└────────────────────────────────────────────────────────────┘

### **Technology Stack**

| **Layer** | **Technology** | **Purpose** | **Cost** | **License** |
| --- | --- | --- | --- | --- |
| **Frontend** | HTML5/CSS3/JS | UI (existing) | $0 | - |
| **PDF Generation** | pdf-lib v1.17+ | Fill PDFs | $0 | MIT |
| **PDF Template** | CommonForms | Create fillable PDF | $0 | Open Source |
| **Storage** | localStorage API | Save drafts | $0 | Browser native |
| **Hosting** | GitHub Pages | Static hosting | $0 | Free tier |
| **Domain** | Optional | Custom URL | $12/yr | - |
| **Analytics** | Google Analytics | Usage tracking | $0 | Free tier |

**Total Cost:** $0-12/year

### **File Structure**

basel-smart-form/

├── index.html # Main application (existing HTML)

├── public/

│ └── basel\_template\_fillable.pdf # Generated fillable template

├── src/

│ ├── js/

│ │ ├── app.js # Main application logic

│ │ ├── pdf-mapper.js # PDF generation engine

│ │ ├── storage.js # Draft persistence

│ │ └── validation.js # Form validation

│ ├── css/

│ │ └── styles.css # Application styles (existing)

│ └── data/

│ └── field-mappings.json # Field mapping configuration

├── docs/

│ ├── user-guide.md # User documentation

│ ├── faq.md # Frequently asked questions

│ └── vCOP8.pdf # Original Basel form (reference)

├── assets/

│ └── images/ # Any images needed

├── .gitignore

└── README.md # Project documentation

## **📅 Development Timeline**

### **Phase 1: MVP Development (2-3 Weeks)**

#### **Week 1: Foundation Setup**

**Days 1-2: PDF Template Creation (6-8 hours)**

* [ ] Install CommonForms
* [ ] Generate fillable PDF from vCOP8.pdf
* [ ] Extract field names (150-180 expected)
* [ ] Verify quality (>85% detection rate)
* [ ] Document missing fields if any
* [ ] Test fillability in Adobe Reader

**Days 3-4: Field Mapping (8 hours)**

* [ ] Create mapping spreadsheet
* [ ] Map all HTML field IDs to PDF field names
* [ ] Add code translation tables
* [ ] Handle conditional fields
* [ ] Convert to JSON format
* [ ] Validate JSON structure
* [ ] Test with 10 sample fields

**Day 5: Development Environment (4 hours)**

* [ ] Create project structure
* [ ] Set up Git repository
* [ ] Copy existing HTML as starting point
* [ ] Add placeholder JS files
* [ ] Configure pdf-lib (CDN link)
* [ ] Test basic page load

**Deliverables:**

* basel\_template\_fillable.pdf
* field-mappings.json (complete)
* Project scaffolding
* Git repo initialized

#### **Week 2: Core Implementation**

**Days 1-2: PDF Generation (10 hours)**

* [ ] Implement BaselPDFMapper class
* [ ] Load and parse field mappings
* [ ] Implement field filling logic
* [ ] Add code translation
* [ ] Handle conditional fields
* [ ] Add error handling
* [ ] Test with sample data
* [ ] Fix mapping errors

**Days 3-4: Draft Persistence (8 hours)**

* [ ] Implement BaselFormStorage class
* [ ] Add manual save/load
* [ ] Implement auto-save (30 sec intervals)
* [ ] Add draft info display
* [ ] Create export/import functionality
* [ ] Add toast notifications
* [ ] Handle storage quota errors
* [ ] Test across browser sessions

**Day 5: Validation & Polish (6 hours)**

* [ ] Implement BaselFormValidator class
* [ ] Add required field checks
* [ ] Add business rules
* [ ] Add data format validation
* [ ] Add consistency checks
* [ ] Integrate with PDF generation flow
* [ ] Polish UI/UX details
* [ ] Fix any bugs

**Deliverables:**

* Working PDF generation
* Complete draft system
* Validation system
* Bug-free core functionality

#### **Week 3: Testing & Launch**

**Days 1-2: Enhancement & Testing (8 hours)**

* [ ] Write 22 AI assistant content blocks
* [ ] Add reference weblinks
* [ ] Cross-browser testing
* [ ] Mobile responsiveness check
* [ ] Performance optimization
* [ ] Accessibility audit
* [ ] User acceptance testing (5 users)
* [ ] Fix reported issues

**Days 3-4: Documentation (6 hours)**

* [ ] Write user guide
* [ ] Create FAQ
* [ ] Add inline help improvements
* [ ] Write README
* [ ] Document code
* [ ] Create troubleshooting guide
* [ ] Record demo video (optional)

**Day 5: Deployment & Launch (4 hours)**

* [ ] Deploy to GitHub Pages
* [ ] Configure custom domain (optional)
* [ ] Set up Google Analytics
* [ ] Final smoke testing
* [ ] Write launch announcement
* [ ] Share on relevant forums
* [ ] Monitor for issues

**Deliverables:**

* Production-ready application
* Complete documentation
* Deployed and accessible
* Launch announcement published

### **Success Criteria for MVP Launch**

**Must Have:**

* ✅ All 180+ fields mapped correctly
* ✅ PDF generates without errors (>95% success rate)
* ✅ Draft saving/loading works reliably
* ✅ Validation catches all required fields
* ✅ Works on Chrome, Firefox, Safari, Edge
* ✅ Mobile responsive
* ✅ User documentation complete
* ✅ Disclaimer prominently displayed
* ✅ 5+ successful beta tests

**Should Have:**

* ✅ Auto-save every 30 seconds
* ✅ Export/import functionality
* ✅ 22 AI content blocks written
* ✅ Reference links added
* ✅ Progress indicator
* ✅ Toast notifications

**Nice to Have (Post-MVP):**

* ⏭️ Live AI API integration
* ⏭️ Cloud sync
* ⏭️ User accounts
* ⏭️ Submission history
* ⏭️ Multi-language support

## **🎨 User Experience Flow**

### **User Journey: First-Time User**

1. Arrive at site

↓

2. See header: "Basel Convention Smart Form Assistant"

Brief explanation of what it does

↓

3. (Optional) Load draft? [Skip] [Load Previous]

↓

4. Start filling form

- See progress indicator

- Use tooltips for help

- Click AI assistant when stuck

- Move between sections freely

↓

5. Draft auto-saves every 30 seconds

- See "Saved ✓" notification

- See "45% complete" indicator

↓

6. (Optional) Export draft for backup

↓

7. Need to leave?

- Close browser

- Data automatically saved

↓

8. Return later

- Load draft automatically offered

- Continue where left off

↓

9. Complete all sections

- Progress shows "100%"

↓

10. Click "Generate PDF"

- Validation runs

- Any errors highlighted

↓

11. Fix errors (if any)

- Try again

↓

12. PDF generates successfully

- Loading indicator shows progress

- "Success! ✓" message

- PDF downloads automatically

↓

13. Open PDF in Adobe Reader

- Verify all fields filled

- Submit to competent authority

### **User Journey: Returning User**

1. Arrive at site

↓

2. See "Load draft from Oct 15?" prompt

↓

3. Click [Load Draft]

- Form populates instantly

- See "78% complete"

↓

4. Complete remaining fields

↓

5. Generate and download PDF

## **🔒 Security & Privacy**

### **Data Privacy Principles**

1. **No Data Leaves User's Computer**
   * All processing client-side
   * No server communication
   * No external data storage
   * No user tracking (except anonymous analytics)
2. **User Data Ownership**
   * Users own their data
   * Can export anytime
   * Can delete anytime
   * No vendor lock-in
3. **Transparency**
   * Open source code
   * Visible data storage (localStorage)
   * Clear privacy policy
   * No hidden tracking

### **Security Measures**

1. **Input Sanitization**
   * HTML encoding of user inputs
   * XSS prevention
   * No eval() of user data (only config JSON)
2. **Secure Defaults**
   * HTTPS only (enforced by GitHub Pages)
   * No external script loading (except CDN)
   * Content Security Policy headers
3. **Browser Security**
   * localStorage encrypted by browser
   * Same-origin policy enforced
   * Secure cookie handling

### **Disclaimer & Legal**

**Prominent disclaimer on homepage:**

⚠️ IMPORTANT DISCLAIMER

This tool assists with completing Basel Convention notification forms

but does NOT guarantee compliance with all regulations.

- Always verify information with your competent authority

- Review generated PDFs carefully before submission

- This tool does not provide legal or regulatory advice

- Users are responsible for accuracy of submitted information

- Consider professional consultation for complex cases

Use at your own risk. No warranties provided.

## **📊 Success Metrics & Analytics**

### **Key Performance Indicators (KPIs)**

**Technical Metrics:**

* PDF generation success rate: Target >95%
* Average page load time: Target <3 seconds
* Average PDF generation time: Target <5 seconds
* Browser compatibility: Target >98% of users
* Mobile usage: Track percentage

**User Experience Metrics:**

* Form completion rate: Target >60%
* Average time to complete: Target <30 minutes
* Bounce rate: Target <40%
* Return user rate: Target >40%
* Draft save/load usage: Track frequency

**Engagement Metrics:**

* AI assistant button clicks per session
* Tooltip hover/click rate
* Most used features
* Most problematic sections (high abandonment)
* Export/import usage

### **Analytics Implementation**

**Google Analytics 4 Setup:**

<!-- Add to index.html <head> -->

<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXXXXX"></script>

<script>

window.dataLayer = window.dataLayer || [];

function gtag(){dataLayer.push(arguments);}

gtag('js', new Date());

gtag('config', 'G-XXXXXXXXXX', {

'anonymize\_ip': true,

'cookie\_flags': 'SameSite=None;Secure'

});

</script>

**Custom Events to Track:**

// PDF generation

gtag('event', 'pdf\_generated', {

'success': true,

'fields\_filled': 156,

'generation\_time': 3.2

});

// Draft saved

gtag('event', 'draft\_saved', {

'completion\_percentage': 67,

'auto\_save': true

});

// AI assistant used

gtag('event', 'ai\_assistant\_clicked', {

'section': 'exporter\_section'

});

// Validation errors

gtag('event', 'validation\_failed', {

'error\_count': 3

});

### **User Feedback Collection**

**Post-PDF Generation Survey (Optional):**

<div id="feedbackPrompt" style="display:none;">

<h3>How was your experience?</h3>

<p>Rate your experience completing this form:</p>

<div class="rating">

<button onclick="submitFeedback(1)">😞 1</button>

<button onclick="submitFeedback(2)">🙁 2</button>

<button onclick="submitFeedback(3)">😐 3</button>

<button onclick="submitFeedback(4)">🙂 4</button>

<button onclick="submitFeedback(5)">😊 5</button>

</div>

<textarea placeholder="Any comments? (optional)"></textarea>

<button onclick="skipFeedback()">Skip</button>

</div>

## **🚀 Deployment**

### **Hosting on GitHub Pages**

**Setup Process:**

# 1. Create GitHub repository

git init

git remote add origin https://github.com/yourusername/basel-smart-form.git

# 2. Commit all files

git add .

git commit -m "Initial commit: Basel Smart Form MVP"

# 3. Push to GitHub

git push -u origin main

# 4. Enable GitHub Pages

# Go to repository Settings > Pages

# Source: Deploy from branch

# Branch: main

# Folder: / (root)

# Click Save

# 5. Site will be live at:

# https://yourusername.github.io/basel-smart-form/

### **Custom Domain (Optional)**

**If you want custom domain (e.g., baselassistant.com):**

1. **Purchase domain** ($12/year from Namecheap, Google Domains, etc.)

**Configure DNS:** Type: CNAME

Name: www

Value: yourusername.github.io

Type: A

Name: @

Value: 185.199.108.153

185.199.109.153

185.199.110.153

185.199.111.153

**Add CNAME file to repo:** echo "baselassistant.com" > CNAME

git add CNAME

git commit -m "Add custom domain"

git push

1. **Enable HTTPS in GitHub Pages settings**

### **Deployment Checklist**

**Before deployment:**

* [ ] All features tested and working
* [ ] No console errors
* [ ] Mobile responsive
* [ ] Cross-browser tested
* [ ] Analytics configured
* [ ] Disclaimer prominently displayed
* [ ] README complete
* [ ] User documentation ready

**After deployment:**

* [ ] Visit live URL and test
* [ ] Test PDF generation on live site
* [ ] Check analytics tracking
* [ ] Test on mobile device
* [ ] Verify HTTPS certificate
* [ ] Check all external links work

## **🧪 Testing Strategy**

### **Testing Levels**

#### **1. Unit Testing (Manual)**

**Test each JavaScript module individually:**

// Test PDF Mapper

console.log('Testing PDF Mapper...');

const mapper = new BaselPDFMapper();

await mapper.initialize();

console.log('✓ Mapper initialized');

const testData = { exporter\_reg: 'TEST-001' };

const result = await mapper.generatePDF(testData);

console.log('✓ PDF generated:', result.stats);

// Test Storage

console.log('Testing Storage...');

const storage = new BaselFormStorage();

storage.save(testData);

console.log('✓ Data saved');

const loaded = storage.load();

console.log('✓ Data loaded:', loaded);

// Test Validation

console.log('Testing Validation...');

const validator = new BaselFormValidator();

const validation = validator.validate(testData);

console.log('✓ Validation complete:', validation);

#### **2. Integration Testing**

**Test complete workflow:**

**Test Case 1: Happy Path**

1. Fill out all required fields
2. Click "Generate PDF"
3. Verify PDF downloads
4. Open PDF and check fields

**Test Case 2: Draft Save/Load**

1. Fill out 50% of form
2. Close browser
3. Reopen
4. Verify draft loads
5. Complete form
6. Generate PDF

**Test Case 3: Validation Errors**

1. Leave required fields empty
2. Click "Generate PDF"
3. Verify error messages display
4. Fill missing fields
5. Verify PDF generates

**Test Case 4: Export/Import**

1. Fill out form
2. Export draft
3. Clear browser data
4. Import draft
5. Verify data restored

#### **3. User Acceptance Testing (UAT)**

**Recruit 5-10 beta testers:**

**Tester Profile:**

* Building facilities personnel
* Non-technical users
* Have completed Basel forms before
* Willing to provide feedback

**Testing Protocol:**

1. Give testers URL (no instructions)
2. Ask them to complete a sample notification
3. Observe (if possible) or collect feedback after
4. Ask specific questions:
   * Was it clear what to do?
   * Did tooltips help?
   * Did you use AI assistant?
   * Any confusing sections?
   * How long did it take?
   * Would you use this instead of manual PDF?

**Success Criteria:**

* 4/5 testers complete form successfully
* Average time <30 minutes
* Average satisfaction >4/5
* No critical bugs reported
* PDF output accepted by testers

#### **4. Cross-Browser Testing**

**Test on:**

* ✅ Chrome (latest)
* ✅ Firefox (latest)
* ✅ Safari (latest)
* ✅ Edge (latest)

**Test on mobile:**

* ✅ iOS Safari
* ✅ Android Chrome

**What to verify:**

* Form renders correctly
* Tooltips work
* PDF generates
* Downloads work
* localStorage persists
* No console errors

#### **5. Performance Testing**

**Metrics to measure:**

// Page load time

performance.measure('page-load', 'navigationStart', 'domContentLoadedEventEnd');

// PDF generation time

const startTime = performance.now();

await generatePDF(formData);

const endTime = performance.now();

console.log(`PDF generation: ${endTime - startTime}ms`);

// localStorage size

const data = localStorage.getItem('basel\_form\_draft\_v1');

const sizeKB = new Blob([data]).size / 1024;

console.log(`Draft size: ${sizeKB.toFixed(2)} KB`);

**Performance Targets:**

* Page load: <3 seconds
* PDF generation: <5 seconds
* Draft save: <100ms
* localStorage usage: <2MB

## **🐛 Troubleshooting & Support**

### **Common Issues & Solutions**

**Issue 1: PDF not generating**

**Symptoms:** Button clicks but nothing happens

**Debugging:**

// Check console for errors

console.log('Checking PDF mapper...');

console.log('Initialized?', window.pdfMapper.initialized);

// Test with minimal data

const testData = { exporter\_reg: 'TEST' };

await window.pdfMapper.generatePDF(testData);

**Solutions:**

* Verify pdf-lib loaded (check Network tab)
* Verify field-mappings.json loads correctly
* Check for PDF field name mismatches
* Verify template PDF exists at correct path

**Issue 2: Draft not saving**

**Symptoms:** No "Saved ✓" notification

**Debugging:**

// Check localStorage available

console.log('localStorage available?',

typeof(Storage) !== "undefined");

// Check quota

navigator.storage.estimate().then(estimate => {

console.log('Used:', estimate.usage);

console.log('Quota:', estimate.quota);

});

**Solutions:**

* Check browser allows localStorage
* Clear old drafts if quota exceeded
* Use export/import as workaround
* Check browser in private mode

**Issue 3: Fields not mapping correctly**

**Symptoms:** PDF has empty fields

**Debugging:**

// Check field mappings

console.log('Mappings loaded:', window.pdfMapper.mappings);

// Check form data collection

const data = window.storage.collectFormData();

console.log('Form data:', data);

// Check specific field

const mapping = window.pdfMapper.mappings.mappings

.find(m => m.htmlField === 'exporter\_reg');

console.log('Mapping for exporter\_reg:', mapping);

**Solutions:**

* Verify HTML field IDs match mapping JSON
* Check for typos in field names
* Verify PDF field names correct
* Test with single field first

### **User Support Plan**

**Support Channels:**

1. **Documentation First**
   * Comprehensive user guide
   * FAQ section
   * Video tutorials (optional)
2. **GitHub Issues**
   * Users can report bugs
   * Feature requests
   * Questions
3. **Email Support**
   * support@baselassistant.com (if custom domain)
   * Response within 48 hours

**Support Escalation:**

1. Check documentation
2. Check FAQ
3. Search GitHub issues
4. Create new GitHub issue
5. Email support (critical issues only)

## **📖 Documentation Requirements**

### **User Guide (docs/user-guide.md)**

**Contents:**

1. **Getting Started**
   * What is this tool?
   * Who should use it?
   * What you'll need
2. **Using the Form**
   * Understanding sections
   * Using tooltips
   * Using AI assistant
   * Moving between sections
   * Saving drafts
3. **Understanding Codes**
   * Y-codes
   * H-codes
   * R-codes / D-codes
   * UN classifications
4. **Generating PDF**
   * When you're ready
   * Validation process
   * Fixing errors
   * Downloading PDF
5. **Tips & Best Practices**
   * Gather info before starting
   * Use reference links
   * Save frequently
   * Review before submitting

### **FAQ (docs/faq.md)**

**Common Questions:**

Q: Is my data secure?  
 A: Yes. All data stays on your computer. Nothing is sent to any server.

Q: Can I access my draft from another computer?  
 A: Not in MVP. Use Export feature to transfer drafts. Cloud sync coming in Phase 2.

Q: What browsers are supported?  
 A: Chrome, Firefox, Safari, Edge (latest versions).

Q: Is this an official Basel Convention tool?  
 A: No. This is an independent assistance tool. Always verify with official authorities.

Q: Does this guarantee compliance?  
 A: No. This assists with completion but doesn't guarantee compliance. Review carefully.

Q: Can I edit the PDF after generating?  
 A: Yes, open in Adobe Reader and make changes if needed.

Q: What if I find a bug?  
 A: Please report on GitHub Issues or email support.

### **Code Documentation**

**README.md:**

# Basel Convention Smart Form Assistant

Intelligent web-based form assistant for completing Basel Convention

hazardous waste notification forms.

## Features

- 180+ fields with contextual help

- AI-powered assistance

- Automatic PDF generation

- Draft saving

- Free and open source

## Quick Start

1. Visit [live site]

2. Start filling form

3. Use tooltips and AI assistance

4. Generate PDF when complete

## For Developers

See [DEVELOPER.md](docs/DEVELOPER.md) for technical details.

## License

MIT License - free for personal and commercial use

## Disclaimer

This tool assists with form completion but does not guarantee

compliance. Always verify with competent authorities.

**Inline Code Comments:**

/\*\*

\* Generate filled Basel Convention PDF from form data

\*

\* @param {Object} formData - Key-value pairs of form field data

\* @returns {Promise<Object>} - { pdfBytes, stats }

\*

\* @example

\* const formData = { exporter\_reg: 'TT-001', ... };

\* const result = await mapper.generatePDF(formData);

\* downloadPDF(result.pdfBytes, 'output.pdf');

\*/

async generatePDF(formData) {

// Implementation...

}

## **🎓 Development Guidelines for Claude Code**

### **Prompt for Claude Code CLI**

I need you to develop the Basel Convention Smart Form Assistant

based on this PRD.

IMPORTANT CONTEXT:

- We're enhancing EXISTING HTML forms (70% complete)

- Primary file: basel\_form\_Final\_with\_buttons-v1.html

- DO NOT rebuild from scratch - enhance existing code

- Focus on the 30% that needs to be built (see PRD Feature sections)

DEVELOPMENT PRIORITIES (in order):

1. Create fillable PDF template (using CommonForms - instructions in PRD)

2. Create field-mappings.json (map HTML IDs to PDF fields)

3. Implement pdf-mapper.js (PDF generation engine)

4. Implement storage.js (draft persistence)

5. Implement validation.js (form validation)

6. Integrate all modules into existing HTML

7. Write AI assistant content (22 sections)

8. Test and fix bugs

TECHNICAL REQUIREMENTS:

- Pure JavaScript (no frameworks)

- Use pdf-lib v1.17+ from CDN

- localStorage for persistence

- Client-side only (no backend)

- Must work offline after first load

REFER TO:

- PRD Feature sections for detailed requirements

- Code examples provided in PRD

- Architecture diagram for structure

- Success criteria for validation

TESTING:

- Test each module before integration

- Cross-browser compatibility required

- Mobile responsive required

- User acceptance testing with 5 users

OUTPUT:

- Complete working application

- All files in correct directory structure

- README with setup instructions

- User guide documentation

BEGIN with Step 1: PDF Template Creation using CommonForms.

After each major step, ask for verification before proceeding.

### **Iterative Development Approach**

**Session 1: Foundation**

* Create fillable PDF template
* Set up project structure
* Create field mappings JSON
* Verify template quality

**Session 2: Core Implementation**

* Implement PDF generation
* Test with 10 fields
* Expand to all fields
* Fix mapping errors

**Session 3: Features**

* Implement draft persistence
* Add validation
* Write AI content
* Integrate everything

**Session 4: Polish & Test**

* Cross-browser testing
* Mobile responsive fixes
* Performance optimization
* Bug fixes

**Session 5: Documentation & Launch**

* Write user guide
* Create FAQ
* Deploy to GitHub Pages
* Launch

## **✅ Acceptance Criteria**

### **Minimum Viable Product (MVP) Definition**

**The MVP is accepted when:**

#### **Core Functionality**

* [ ] User can fill out all 180+ form fields
* [ ] Tooltips display correctly on all fields
* [ ] AI assistant panels work on all sections
* [ ] PDF generates successfully (>95% success rate)
* [ ] All form data correctly populates PDF
* [ ] Code translations work (packaging, transport, etc.)
* [ ] Conditional fields display/hide correctly

#### **Draft System**

* [ ] Manual save button saves draft
* [ ] Draft loads on page refresh
* [ ] Auto-save runs every 30 seconds
* [ ] Draft info displays (timestamp, % complete)
* [ ] Export/import functionality works
* [ ] Toast notifications display correctly

#### **Validation**

* [ ] Required fields validated before PDF generation
* [ ] Business rules enforced (disposal XOR recovery)
* [ ] Clear error messages display
* [ ] Focus moves to first error field
* [ ] Warnings don't block submission

#### **User Experience**

* [ ] Form loads in <3 seconds
* [ ] PDF generates in <5 seconds
* [ ] Mobile responsive design works
* [ ] Navigation between sections is smooth
* [ ] Progress indicator updates correctly
* [ ] No console errors

#### **Cross-Platform**

* [ ] Works on Chrome (latest)
* [ ] Works on Firefox (latest)
* [ ] Works on Safari (latest)
* [ ] Works on Edge (latest)
* [ ] Works on mobile (iOS/Android)

#### **Documentation**

* [ ] User guide complete
* [ ] FAQ written
* [ ] README clear
* [ ] Code commented
* [ ] Disclaimer prominent

#### **Testing**

* [ ] 5+ successful beta tests completed
* [ ] Average completion time <30 minutes
* [ ] Average satisfaction >4.0/5.0
* [ ] No critical bugs remaining

#### **Deployment**

* [ ] Deployed to GitHub Pages
* [ ] HTTPS enabled
* [ ] Analytics configured
* [ ] All links work
* [ ] Custom domain configured (optional)

## **🔄 Post-MVP Roadmap (Phase 2)**

**Not required for MVP, but planned for future:**

### **Phase 2: Enhanced Features (Months 2-3)**

**New Features:**

* User accounts (Firebase Auth)
* Cloud draft storage (Firebase/Supabase)
* Multi-device sync
* Live AI API integration (Claude/OpenAI)
* Submission history
* Email notifications
* Team collaboration
* Version history

**Technical Changes:**

* Add Node.js backend (Express)
* Add database (PostgreSQL/MongoDB)
* Add authentication layer
* Add API endpoints
* Add file storage (S3)

**Investment:** $1,000-2,500 or 40-60 hours  
 **Monthly Cost:** $10-50/month  
 **Timeline:** 4-6 weeks

### **Phase 3: Full SaaS (Months 4-6)**

**Enterprise Features:**

* Multi-tenancy
* Role-based access
* White-label options
* Payment processing (Stripe)
* Advanced analytics
* API for integrations
* Mobile apps
* Audit logs
* Compliance tracking

**Investment:** $10,000-30,000  
 **Monthly Cost:** $100-500/month  
 **Timeline:** 3-6 months

**Revenue Potential:**

* 50 users @ $29/mo = $17,400/year
* 200 users @ $39/mo = $93,600/year
* 500 users @ $49/mo = $294,000/year

## **📊 Project Risks & Mitigation**

### **Technical Risks**

**Risk 1: CommonForms doesn't detect enough fields**

**Impact:** High - can't create fillable template  
 **Probability:** Medium (expected 90-95% accuracy)  
 **Mitigation:**

* Test CommonForms early (Week 1, Day 1)
* If <85% accuracy, pivot to Fiverr option ($75-150)
* Manual addition of missing fields possible
* Budget 3-5 days for Fiverr if needed

**Risk 2: Field mapping errors**

**Impact:** High - incorrect PDF output  
 **Probability:** Medium  
 **Mitigation:**

* Create mapping spreadsheet first (human-readable)
* Test with 10 fields before mapping all
* Use sample data for testing
* Beta test with real users
* Iterate based on feedback

**Risk 3: Browser compatibility issues**

**Impact:** Medium - some users can't access  
 **Probability:** Low (pdf-lib has good support)  
 **Mitigation:**

* Test on all major browsers early
* Use feature detection, not browser detection
* Provide fallback messages
* Document browser requirements

**Risk 4: localStorage quota exceeded**

**Impact:** Low - draft saving fails  
 **Probability:** Low (forms are small)  
 **Mitigation:**

* Implement quota detection
* Provide export functionality
* Warn users before clearing
* Compress data if needed

### **Business Risks**

**Risk 5: Low user adoption**

**Impact:** Medium - product not used  
 **Probability:** Medium  
 **Mitigation:**

* Beta test with 5-10 real users first
* Gather and implement feedback
* Market to target audience
* Free = low barrier to entry
* Focus on UX excellence

**Risk 6: Regulatory changes**

**Impact:** Medium - form becomes outdated  
 **Probability:** Low (Basel forms stable)  
 **Mitigation:**

* Monitor Basel Convention website
* Version control field mappings
* Easy to update (just JSON)
* Subscribe to official updates

**Risk 7: Liability for errors**

**Impact:** High - legal issues  
 **Probability:** Low  
 **Mitigation:**

* Prominent disclaimer
* "Assistance tool" not "compliance guarantee"
* Recommend professional review
* Open source = community review
* No warranties provided

## **💰 Budget & Resources**

### **MVP Budget Breakdown**

| **Item** | **DIY Cost** | **Outsourced Cost** | **Notes** |
| --- | --- | --- | --- |
| **Development** |  |  |  |
| PDF template creation | $0 | $75-150 (Fiverr) | CommonForms is free |
| Field mapping | $0 | (included) | 6-8 hours your time |
| PDF generation code | $0 | (included) | 8-12 hours your time |
| Draft persistence | $0 | (included) | 4-6 hours your time |
| Validation system | $0 | (included) | 3-4 hours your time |
| AI content writing | $0 | $100-200 | 6-8 hours or outsource |
| Testing | $0 | (included) | 8-10 hours your time |
| Documentation | $0 | (included) | 6-8 hours your time |
| **Hosting** |  |  |  |
| GitHub Pages | $0/yr | $0/yr | Free forever |
| Custom domain | $12/yr | $12/yr | Optional |
| SSL certificate | $0/yr | $0/yr | Included with GitHub |
| **Tools** |  |  |  |
| pdf-lib | $0 | $0 | Open source (MIT) |
| CommonForms | $0 | $0 | Open source |
| Development tools | $0 | $0 | VS Code, Git free |
| **Total** | **$12/yr** | **$187-362 + $12/yr** |  |

### **Time Investment (DIY)**

| **Phase** | **Hours** | **Spread Over** |
| --- | --- | --- |
| Week 1: Foundation | 18-20 hours | 5 days |
| Week 2: Implementation | 24 hours | 5 days |
| Week 3: Testing & Launch | 18 hours | 5 days |
| **Total** | **60-62 hours** | **3 weeks** |

**Part-time schedule:** 3-4 hours/day, 5 days/week

### **Cost-Benefit Analysis**

**Investment:** $12/year + 60 hours your time

**Benefits:**

* Working product in 3 weeks
* Helps 10-100 users in first year
* Saves each user 1-2 hours per form
* Prevents costly errors
* Educational value
* Portfolio piece
* Potential revenue stream later

**Break-even:** If helps 10 users save 1 hour each @ $50/hr = $500 value

## **🎓 Learning Outcomes**

**By completing this project, you will learn:**

**Technical Skills:**

* Advanced JavaScript (async/await, modules, classes)
* PDF manipulation with pdf-lib
* Browser APIs (localStorage, Blob, File)
* Form validation techniques
* Client-side architecture
* Performance optimization
* Cross-browser compatibility
* Mobile-responsive design

**Tools & Workflows:**

* Git & GitHub
* GitHub Pages deployment
* CommonForms (AI for PDFs)
* Google Analytics
* VS Code productivity
* Command-line basics
* JSON data structures

**Product Development:**

* Requirements gathering
* User experience design
* Beta testing
* Documentation writing
* Product launch
* User support

**Value:** Skills worth $40,000-80,000/year in job market

## **📞 Support & Communication**

### **During Development**

**For Technical Questions:**

* Refer to PRD Feature sections
* Check implementation guide (190+ pages docs)
* Search Stack Overflow
* GitHub issues of used libraries

**For Architecture Decisions:**

* Follow PRD recommendations
* Document deviations in README
* Ask for clarification if unclear

**For Scope Questions:**

* Stick to MVP requirements
* Defer enhancements to Phase 2
* Focus on core functionality first

### **After Launch**

**User Support:**

* Documentation-first approach
* GitHub Issues for bugs
* Email for critical issues
* Response within 48 hours

**Maintenance:**

* Monitor GitHub Issues
* Fix critical bugs promptly
* Update documentation as needed
* Plan Phase 2 based on feedback

## **✨ Final Notes**

### **Success Factors**

**This project will succeed because:**

1. ✅ **Real problem** - You personally experienced the pain
2. ✅ **Clear solution** - Smart form with guidance
3. ✅ **Strong foundation** - 70% already built
4. ✅ **Proven technology** - All tools battle-tested
5. ✅ **Manageable scope** - 3 weeks to MVP
6. ✅ **Low cost** - $0-12/year
7. ✅ **High value** - Saves hours per form
8. ✅ **Scalable** - Can grow to Phase 2/3

### **Key Principles**

**Follow these principles:**

1. **Start simple** - MVP first, features later
2. **User-focused** - Solve their problems, not yours
3. **Quality over features** - Better to do few things well
4. **Test early, test often** - Catch issues early
5. **Document everything** - Future you will thank you
6. **Ship it** - Done is better than perfect
7. **Iterate** - Improve based on feedback

### **The Path Forward**

Week 1: Foundation

↓

Week 2: Implementation

↓

Week 3: Testing & Launch

↓

Users benefit immediately

↓

Gather feedback

↓

Plan Phase 2 enhancements

↓

Scale based on demand

## **🎯 Ready to Build?**

**You have everything you need:**

* ✅ Clear requirements
* ✅ Detailed specifications
* ✅ Code examples
* ✅ Architecture design
* ✅ Timeline & budget
* ✅ Success criteria
* ✅ Risk mitigation
* ✅ Support plan

**Your existing assets:**

* ✅ 70% complete HTML forms
* ✅ 190+ tooltips already written
* ✅ UI design finished
* ✅ 190+ pages of documentation

**What you need to build (30%):**

* PDF template creation (2-3 hours)
* Field mapping (6-8 hours)
* PDF generation (8-12 hours)
* Draft persistence (4-6 hours)
* Validation (3-4 hours)
* AI content (6-8 hours)
* Testing (8-10 hours)
* Documentation (6-8 hours)

**Total: 45-60 hours = 3 weeks part-time**

## **📋 PRD Approval**

**Document Status:** ✅ Ready for Development

**Approved By:** Richard (Project Owner)

**Development Start Date:** [To be determined]

**Expected Completion:** 3 weeks from start

**Next Action:**

1. Review this PRD completely
2. Ask any clarification questions
3. Approve PRD
4. Begin Week 1, Day 1: Install CommonForms

**END OF PRODUCT REQUIREMENTS DOCUMENT**

**Version:** 1.0  
 **Date:** October 18, 2025  
 **Pages:** 50+  
 **Status:** Ready for Implementation

**Let's build something amazing! 🚀**