# **Document Extraction Tool - Technical Assessment**

## **Project Overview**

Build a PDF document extraction tool that can accurately parse unstructured agricultural/environmental reports and extract key information with high accuracy. This tool should demonstrate your ability to work with complex document structures, implement intelligent data extraction, and create user-friendly interfaces.

## **The Challenge**

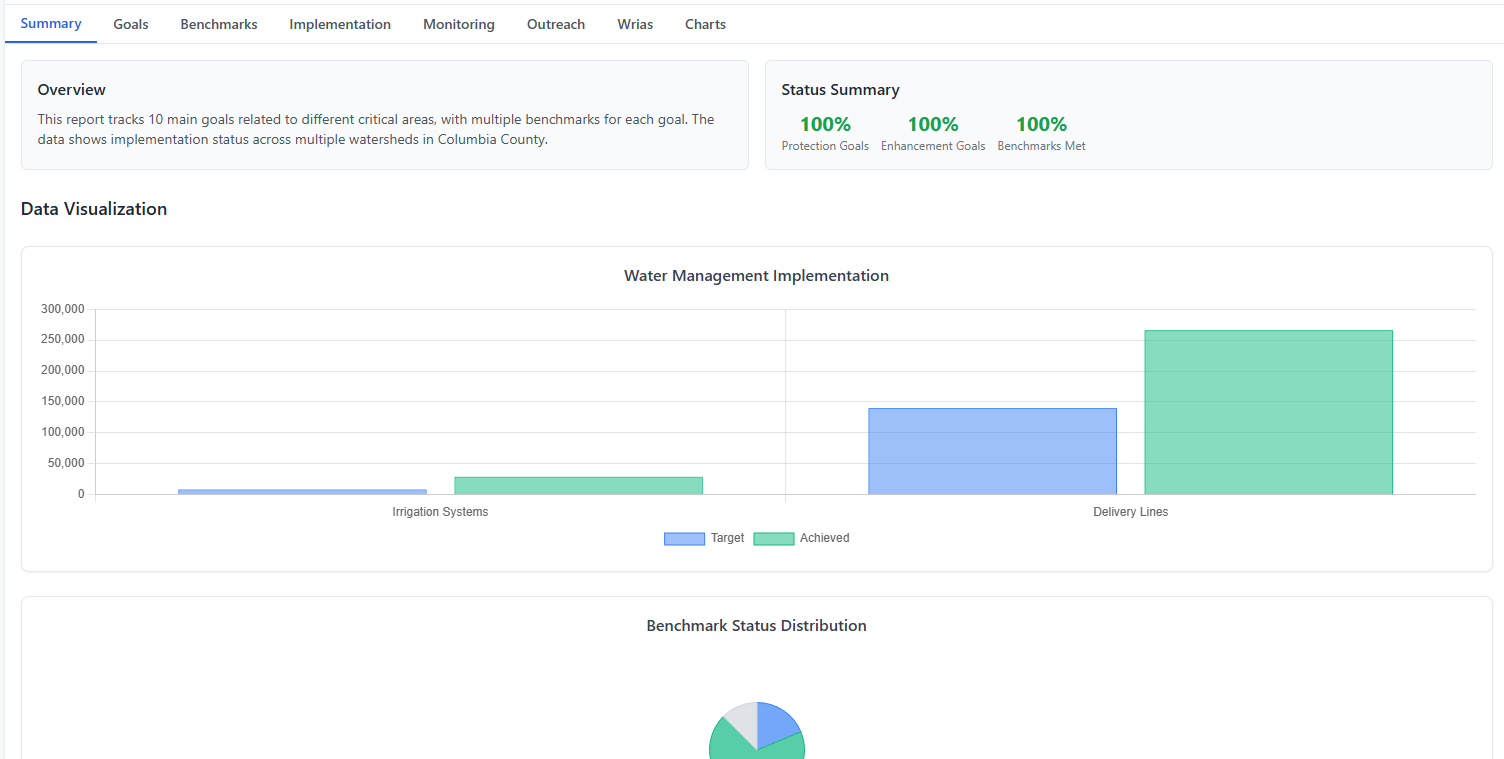
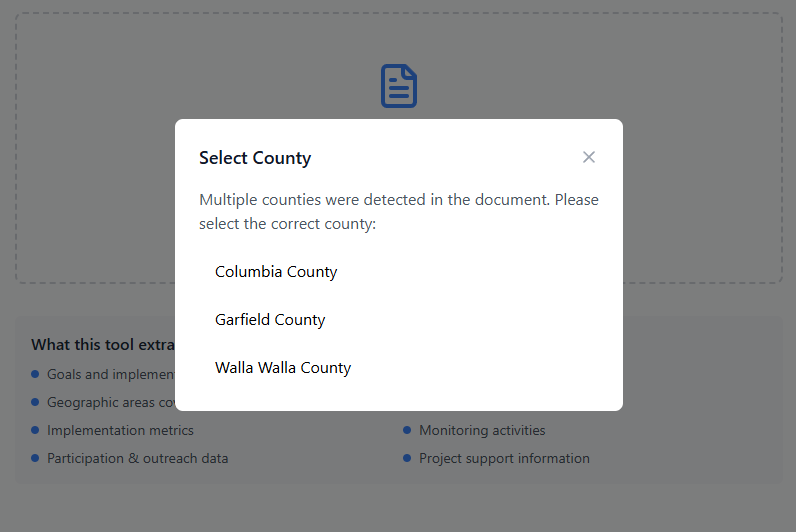
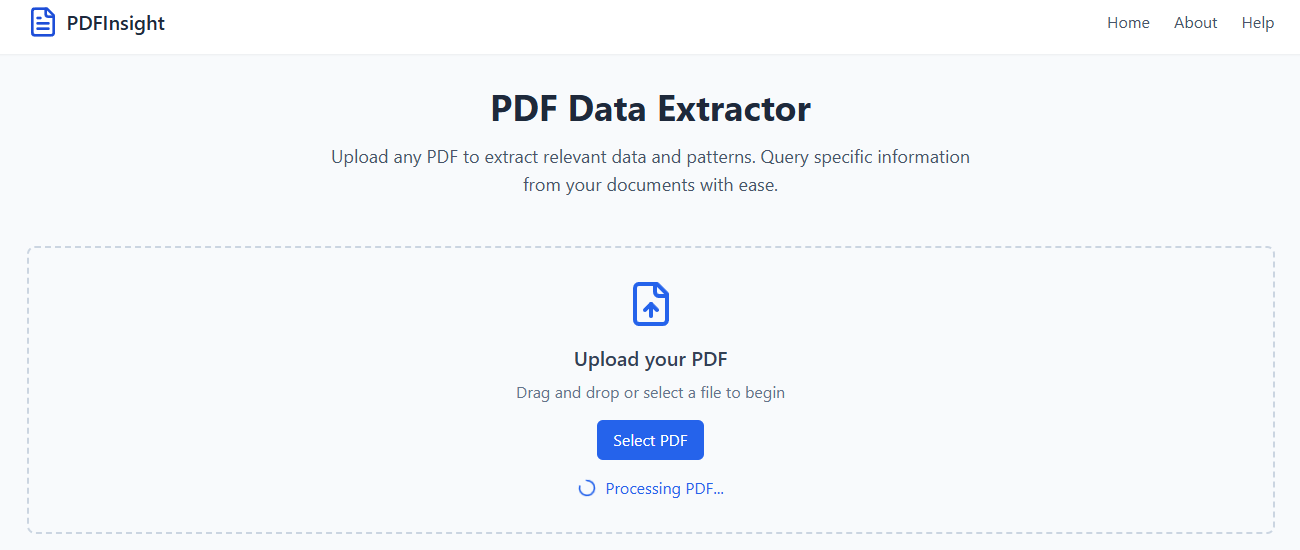
Many government and agricultural reports follow similar structures but have varying formats. Your task is to leverage an LLM model and build a robust extraction tool that can understand document context and extract meaningful data regardless of formatting variations.

## **Requirements**

### **Core Functionality**

1. **PDF Upload & Processing**: Accept PDF uploads via drag-and-drop interface
2. **Intelligent Text Extraction**: Extract text while preserving document structure
3. **Context-Aware Parsing**: Distinguish between main content, sub-bullets, examples, and administrative text
4. **Data Categorization**: Organize extracted information into logical categories
5. **Statistical Dashboard**: Display extracted data with charts and summaries
6. **Export Capability**: Allow users to export results

General example screenshots to help you visualize what it could be, this is not dictating what your tool should look like, this is only examples:



### **Technical Specifications**

Use your own programming languages to build the tool, but below are our recommendations:

#### **Frontend Recommendations**

* **Framework**: React (with TypeScript preferred)
* **Styling**: Tailwind CSS or similar modern framework
* **Charts**: Chart.js, D3.js, or Recharts
* **File Upload**: react-dropzone or similar
* **UI Components**: Clean, modern interface similar to provided screenshots

#### **Backend Recommendations**

* **Runtime**: Node.js
* **PDF Processing**: pdf-parse, pdf2pic, or similar libraries
* **LLM Integration**: OpenAI GPT-4, Anthropic Claude, or similar
* **API Framework**: Express.js, Fastify, or similar

#### **Data Structure**

Your tool should extract and categorize:

typescript

interface ExtractedReport {  
 summary: {  
 totalGoals: number;  
 totalBMPs: number;  
 completionRate: number;  
 };  
 goals: Goal[];  
 bmps: BMP[];  
 implementation: ImplementationActivity[];  
 monitoring: MonitoringMetric[];  
 outreach: OutreachActivity[];  
 geographicAreas: GeographicArea[];  
}

### **Test Data Source**

**Use Mississippi Watershed Plans as your test PDF documents**:

* Website: <https://www.mdeq.ms.gov/wp-content/uploads/SurfaceWaterBasinMgtNonPointSourceBranch/Watershed_Plans/MS_Watershed_Plans.htm>
* These contain unstructured content (goals, implementation measures, monitoring) in varying formats
* Download 3-5 different watershed plans to test format variations

### **Accuracy Requirements**

Your extraction must achieve:

* **≥75% accuracy** in identifying main goals, BMPs and activities
* **≥75% accuracy** in extracting quantitative metrics
* **Zero false positives** for exact copied content (e.g. name of something)
* **Proper categorization** of different content types

### **Evaluation Criteria**

#### **Extraction Accuracy (40%)**

* Correctly distinguishes between main content and supporting text (no misinterpretations of the meanings or context)
* Accurately extracts numerical targets and metrics
* Properly categorizes different types of information
* Handles format variations gracefully

#### **Technical Excellence (35%)**

* Clean, maintainable code architecture
* Proper error handling and validation
* Efficient PDF processing pipeline
* Smart use of LLM APIs

#### **User Experience (15%)**

* Intuitive upload interface
* Clear data visualizations and graphs
* Responsive design
* Loading states and error messaging

#### **Documentation (10%)**

* Clear README detailing your implementation approach and framework
* Code comments explaining logic and steps

## **Deliverables**

### **1. Live Demo**

**Deploy your working application** to one of the following platforms:

* **Netlify** (recommended for static sites with serverless functions)
* **Vercel** (recommended for Next.js applications)
* **Railway** or **Render** (for full-stack applications)
* Or another platform of your choice, but it must be an easy-access URL that you can share with us

**Requirements for live demo:**

* Fully functional PDF upload and processing
* All dashboard features working
* Easy-access URL (no downloading or difficult setup/login steps)

### **2. Complete Codebase**

project-name/  
├── frontend/ # React application  
├── backend/ # Node.js API (or serverless functions)  
├── README.md # Framework and approach  
├── EXTRACTION\_LOGIC.md # Explain your steps  
├── DEPLOYMENT.md # Deployment configuration details  
├── package.json # Dependencies  
└── docker-compose.yml # Optional: containerized setup

### **3. Documentation**

* **README.md**: Implementation approach, framework, and setup/usage instructions
* **EXTRACTION\_LOGIC.md**: Detailed explanation of your steps to achieve accurate extraction
* **TESTING.md**: How you validated accuracy with the EPA documents
* **DEPLOYMENT.md**: How the live demo is configured and deployed

## **Deployment Considerations**

### **Environment Variables**

Ensure your deployment handles:

env

OPENAI\_API\_KEY=your\_key\_here  
ANTHROPIC\_API\_KEY=your\_key\_here  
NODE\_ENV=production

*Do not send us your keys when you submit your zipped codebase! We can get our own keys for testing.*

### **File Upload Considerations**

* Handle large PDF (100MB or more) processing efficiently
* Consider using temporary storage for serverless deployments, if needed

## **Submission Requirements**

### **Code Ownership**

By submitting this assessment, you agree that:

* The complete codebase becomes our property upon submission
* You grant full rights to use, modify, and distribute the code
* You will be compensated for your time regardless of selection outcome

### **Submission Format**

1. **Email the complete codebase** as a ZIP file
2. **Include the live demo URL** prominently in your email body
3. **Include all documentation** listed above

**Email Subject**: PDF Extraction Tool Submission - [Your Name]

## **Success Indicators**

### **What We're Looking For**

* **Working live demo** that we can test immediately
* **Problem-solving approach**: How you tackle the accuracy challenge
* **Technical judgment**: Smart use of regex, LLMs, and validation
* **Code quality**: Clean, readable, maintainable code
* **Attention to detail**: Handles edge cases and provides good UX
* **Production readiness**: Proper deployment and error handling

### **Red Flags**

* Live demo doesn't work or has major bugs
* Poor handling of document structure variations
* Extracting obviously wrong data
* No consideration for accuracy measurement
* Deployment issues or missing environment configuration

## **Compensation**

**Fixed payment** will be provided to all qualifying submissions (those that meet basic technical requirements and include a **working** live demo). Submissions that DO NOT WORK will not be paid. Payment will be processed within 5 business days of submission through EasyPay.

* Submitted within 4 days of assignment being sent = $800
* Submitted within 7 days of assignment being sent = $600
* Submitted within 10 days of assignment being sent = $400

No compensation if more than 10 days have passed of assignment being sent. We assume you’ve decided not to do the assignment if nothing is submitted by the 10th day.

## **Getting Started**

1. Review the watershed plans to understand document structure
2. Plan your extraction approach (regex + LLM hybrid recommended)
3. Set up your deployment pipeline early (don't leave this to the end)
4. Build a simple accuracy testing framework first, since accuracy is the highest weight in the evaluation criteria
5. Focus on precision over too many features
6. Test your live demo thoroughly with different PDFs that have different formats before submission (we will be testing using several different PDFs)

**Good luck!** We're excited to see your approach to this challenging but important problem. Remember: a **working** live demo that we can test immediately is just as important as clean code.