AI Tax Return Agent (Prototype)

Objective

The goal of this project is to develop an intelligent prototype that automates the process of preparing U.S. personal tax returns. The system enables users to upload standard tax forms such as W-2, 1099-INT, and 1099-NEC, automatically extracts key financial information using OCR and LLM-based parsing, applies 2024 IRS tax logic to compute liabilities, and generates a filled-out Form 1040 for download and review.
This prototype demonstrates how AI can simplify a traditionally time-consuming and error-prone task into an end-to-end automated workflow.

System Flow

1. Upload

- The user begins by selecting a filing status (Single, Married Jointly, Married Separately, Head of Household) and uploading one or more tax documents (PDF format).
- The web interface built with Flask + Bootstrap validates that all files are PDFs and temporarily stores them for parsing.
- Once submitted, the backend pipeline begins extracting and analyzing data from each uploaded document.

2. Parse

- Each uploaded file is passed through the pdf_parser.py module.
- The parser first attempts text extraction using pdfplumber.
- If the PDF is scanned or image-based, it automatically falls back to OCR using PyMuPDF (fitz) and pytesseract.
- Extracted text is sent to a lightweight LLM (ChatOpenAI via LangChain) that standardizes the output into structured JSON with fields like:

```
{
  "form_type": "W-2",
  "wages": 65000,
  "federal_income_tax_withheld": 7200,
  "interest_income": 0,
  "nonemployee_compensation": 0
}
```

- These parsed values are aggregated across multiple forms to produce a consolidated income and withholding summary.

3. Calculate

- Parsed data is sent to the tax_logic.py module, which implements the 2024 IRS tax brackets and standard deductions.
- Steps performed:
    - Aggregate total income (from W-2 + 1099s).
    - Apply the correct standard deduction based on filing status.
    - Compute taxable income.
    - Calculate federal tax owed using progressive brackets.
    - Subtract withholding to determine refund or amount due.
- The output is a structured dictionary like:

```
{
  "total_income": 65250,
  "deduction": 14600,
  "taxable_income": 50650,
  "tax": 6000,
  "withheld": 7200,
  "refund": 1200
}
```

4. Generate Form

- The summarized tax data is passed to the form1040_generator.py module.
- Using ReportLab, the module creates a simplified version of Form 1040, displaying filing status, income, deductions, taxable income, tax owed, amount withheld, and refund due
- The generated PDF (output_1040.pdf) is displayed inline within the results page for preview and can be downloaded with a single click.

Codebase Structure

The prototype follows a modular and well-organized architecture, with each component handling a distinct stage of the pipeline. This separation of concerns ensures maintainability, scalability, and clarity in function responsibilities.

```
Tax_Return/
│
├── app.py                  → Flask web server (upload, process, generate)
├── pdf_parser.py           → OCR + LLM-based document extraction
├── tax_logic.py            → IRS 2024 tax logic and refund computation
├── form1040_generator.py   → ReportLab PDF generator (Form 1040 output)
└── templates/
    ├── index.html          → File upload interface
    └── result.html         → Summary view + PDF preview
```

Logic Modules

1. app.py

- Acts as the **main controller**, managing all Flask routes (`/`, `/preview`, `/download`).
- Validates uploads, normalizes filing status, and securely handles temporary files.
- Integrates with parsing, computation, and generation modules to coordinate the full workflow.
- Returns rendered HTML pages for both input (`index.html`) and output (`result.html`).

2. pdf_parser.py

- Extracts text content using `pdfplumber`, with OCR fallback via `fitz` and `pytesseract` for scanned PDFs.
- Uses a **ChatOpenAI** model (via LangChain) to interpret unstructured text and return structured JSON.
- Identifies core fields: wages, federal tax withheld, interest income, and nonemployee compensation.
- Ensures resilience to varying form layouts and partial text quality.

3. tax_logic.py

- Implements the **2024 IRS tax brackets** and standard deduction values for different filing statuses.
- Aggregates income across all uploaded forms to determine total and taxable income.
- Applies progressive rates to compute total tax liability and refund/owed amount.
- Returns a unified summary dictionary for display and PDF generation.

4. form1040_generator.py

- Generates a **simplified IRS Form 1040 summary** using the `reportlab` library.
- Populates fields for total income, deductions, taxable income, tax, and refund/owed.
- Formats values for readability and includes color cues (green for refund, red for amount due).
- Outputs a downloadable PDF (`output_1040.pdf`) and supports inline preview in the web interface.

Document Parsing Approach

The document parsing pipeline combines deterministic text extraction and LLM-based semantic understanding to ensure accuracy across both digital and scanned tax forms. It follows a hybrid strategy that maximizes robustness while keeping computation efficient.

When a file is uploaded, the system first tries direct text extraction using pdfplumber, which works well for machine-readable PDFs. If the extracted text is empty or incomplete, the file is rendered into images using fitz (PyMuPDF) and processed through pytesseract OCR to recover textual content. Once the raw text is available, it is passed to a LangChain-powered ChatOpenAI model with a structured prompt instructing the model to return data in a strict JSON schema.

This design ensures:

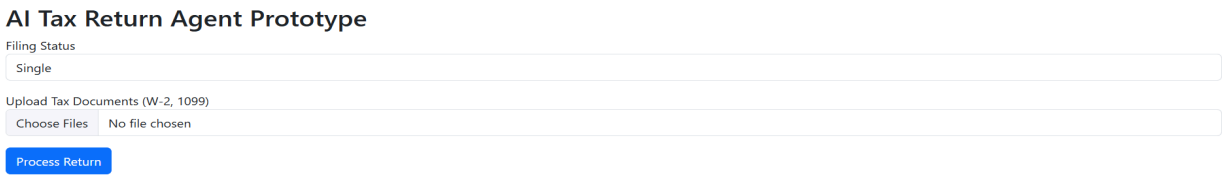
- Consistent field extraction regardless of document source or layout.
- Automatic handling of scanned and image-based forms.
- Reliable normalization of text into numeric fields for downstream tax calculation.
- Minimal manual intervention and improved adaptability for new form types.

Demo

This section demonstrates a complete run of the **AI Tax Return Agent Prototype** using sample W-2 and 1099 documents. The workflow illustrates the simplicity of the user experience from uploading tax forms to viewing and downloading a completed W2 Form.

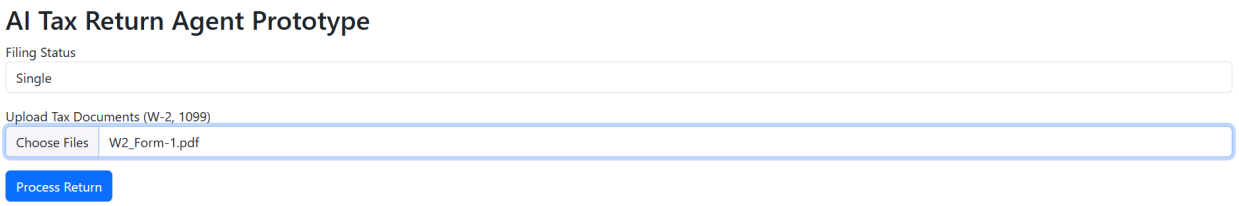
Step 1: Launch the Application

When the Flask server is running, the homepage displays a clean web interface where users can select their filing status and upload tax documents. The system supports multiple PDF files (W-2, 1099-INT, 1099-NEC).



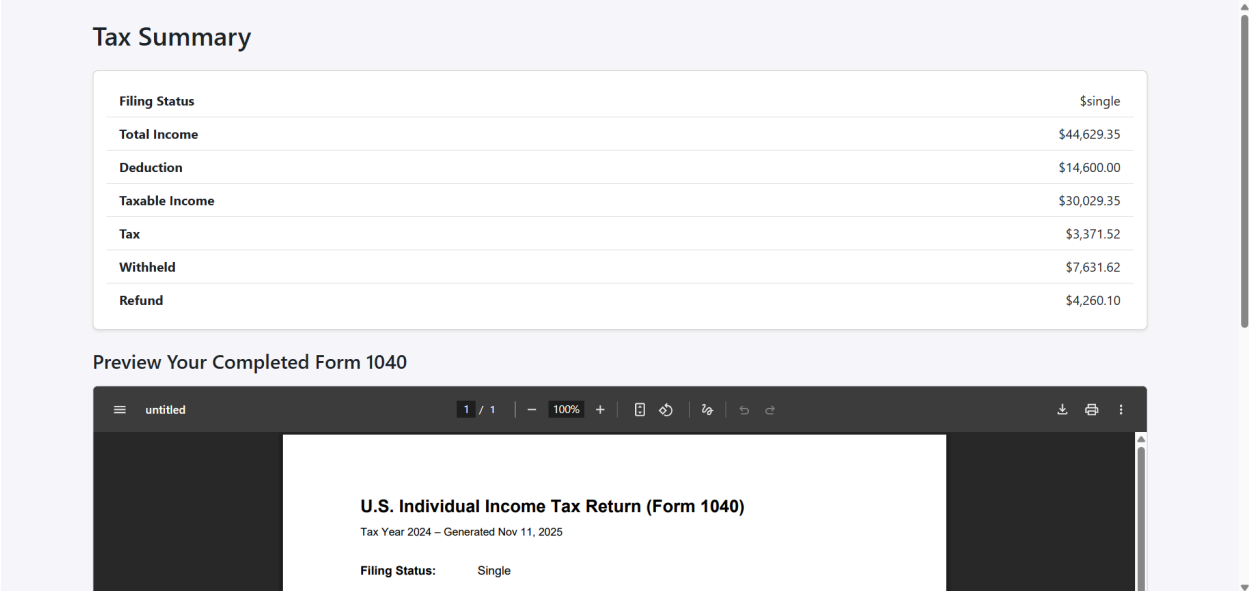
Step 2: Upload Tax Documents

The user selects their filing status (for example, **Single**) and uploads a W-2 form. Once uploaded, clicking **“Process Return”** triggers the backend pipeline that parses and computes the tax summary.



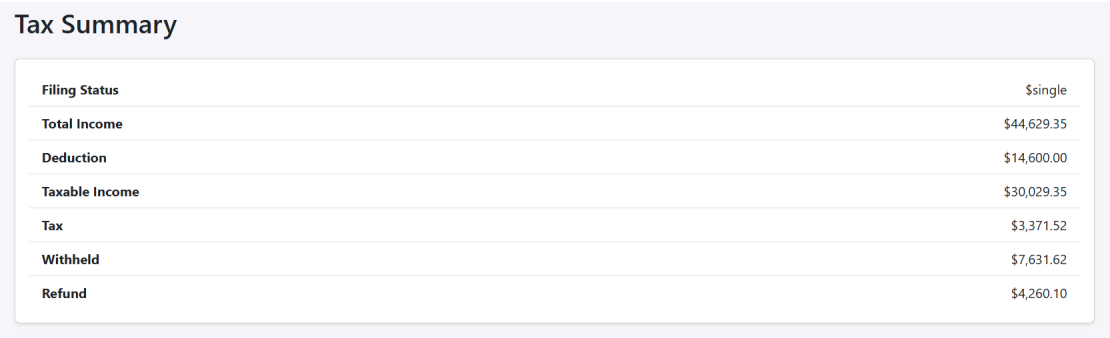
Step 3: View Tax Summary

Within seconds, the system extracts relevant fields using OCR and the LLM parser, calculates tax liability based on the 2024 IRS brackets, and displays a detailed summary of the results including Total Income, Deduction, Taxable Income, Tax, Withheld, and Refund as well as preview if the completed Form 1090



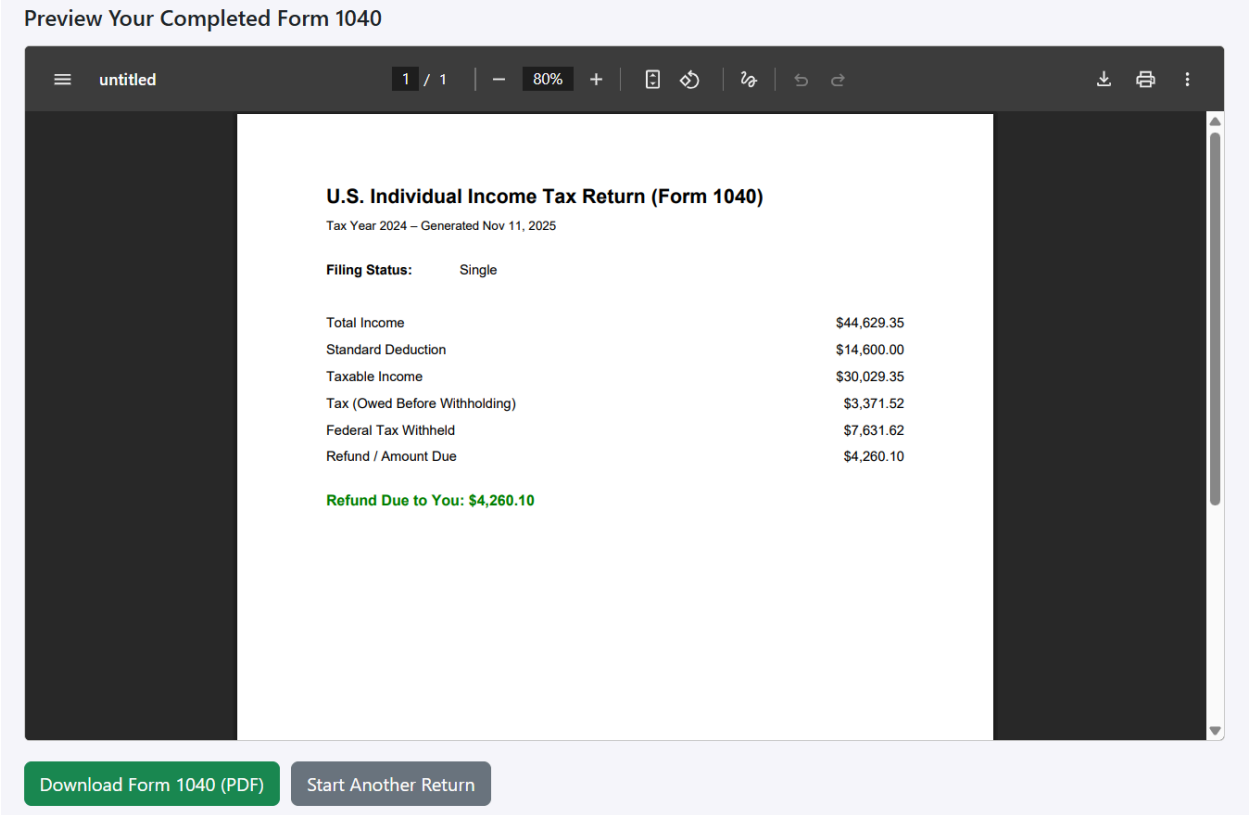
Step 4: Preview Completed Form 1040

The application generates a fully formatted **Form 1040** using ReportLab, embedding the user's calculated data. The preview appears directly within the browser using an inline PDF viewer.



Step 5: Download the Final Report

Users can download their completed **Form 1040 (PDF)** for filing or record-keeping.
The form clearly displays filing status, taxable income, and refund due, formatted in an easy-to-read layout.



Limitations, Future Scope, and Reflection

Limitations

While the prototype successfully demonstrates an end-to-end tax automation pipeline, several practical constraints remain:

- **Limited Form Coverage:** Currently supports only W-2, 1099-INT, and 1099-NEC. Other common forms such as 1098 (mortgage interest) or Schedule C are not yet implemented.
- **Basic Error Handling:** Parsing accuracy can degrade for poorly scanned or low-resolution documents. OCR outputs may occasionally misread numerical values or field labels.
- **Simplified Tax Logic:** The system assumes federal tax computation for standard deduction cases only without itemized deductions, dependents, credits, or state taxes.
- **No Persistent Storage:** All data is processed in memory for privacy, meaning session data is not saved between runs.

Future Scope

To move from prototype to production-ready, several enhancements can be integrated:

- **Multi-Form and Multi-Year Support:** Extend compatibility to additional IRS forms and multiple tax years.
- **AI-Assisted Field Validation:** Use cross-form reconciliation (e.g., matching employer names or income totals) to detect inconsistencies automatically.
- **Secure Cloud Integration:** Implement encrypted document storage, e-filing APIs, and user authentication for secure submissions.
- **Comprehensive Audit Trail:** Generate an explainability report detailing how each value was extracted and calculated for transparency.

Reflection

Developing the **AI Tax Return Agent Prototype** provided valuable insights into bridging document intelligence with financial computation. The OCR + LLM parsing layer worked exceptionally well for structured and semi-structured tax forms, demonstrating that generative models can reliably interpret domain-specific layouts when guided by structured prompts. Integrating this with deterministic logic (tax calculation) and procedural generation (PDF output) showcased how hybrid AI + rule-based systems can automate traditionally manual processes with accuracy and interpretability. The main challenges involved handling the wide variation in PDF layouts and tuning prompts to ensure consistent JSON output. Once stabilized, the workflow proved fast, scalable, and extensible providing a strong foundation for more advanced AI-driven financial automation in the future.