AI Tax Agent Prototype

Introduction

The AI Tax Agent is a user-friendly web app that helps take the stress out of filing taxes. Built to follow the 2025 U.S. federal tax rules, it collects a few key details from the user like income, filing status, and dependents and then automatically calculates their taxes. It not only figures out if you owe money or due to a refund but also creates a clear, downloadable report. This demo offers a glimpse into how AI can make tax filing simpler, more accurate, and less overwhelming for everyone.

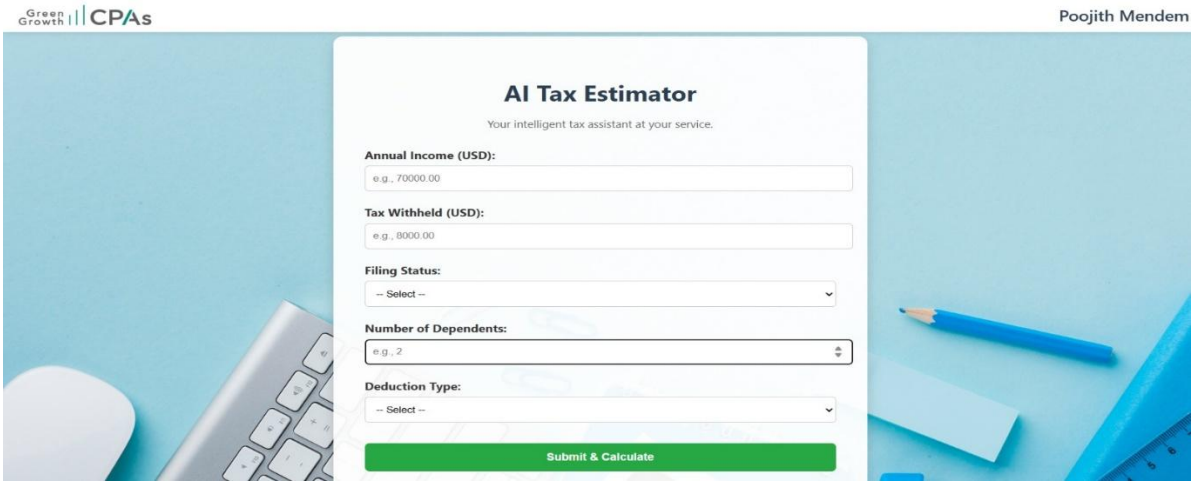
System Architecture Overview

- **Architecture Type**: Client-server model designed for efficient tax return automation.
- **Front-End**: Responsive user interface built with HTML, CSS featuring a form to capture user inputs (income, tax withheld, filing status, deduction type).
- **Back-End**: Flask framework serves as the core server, orchestrating data processing and agent execution.
- **AI Agents**:
  - **Agent 1**: Performs tax calculations using CrewAI, processing inputs and returning results.
  - **Agent 2**: Generates detailed tax reports using CrewAI, saving outputs as text files.
- **Storage**: Local file system (main/ directory) with name as Tax_report.md.

Data Flow

The data flow within the system follows a linear process from user input to result delivery:

- **User Input**: The user enters data (income, filing status, tax withheld) via an HTML form on the front-end.



- **Form Submission**: The form data is sent to the Flask back-end via a POST request to the / route.
- **Back-End Processing**:
  - Flask parses the form data and passes it to a CrewAI crew consisting of two agents.
  - Agent 1 (Tax Calculation) processes the input to compute the tax result.

```
Crew: crew
├── Task: d28b7643-2a47-4677-bbe5-61625bd3b51f
    Assigned to: Tax Calculation Specialist
    Status: Completed
    └── Agent: Tax Calculation Specialist
        Status: Completed
└── Task: 787a0c0f-bd4d-4704-9225-73a758792fed
    Assigned to: Tax Report Writer
    Status: Completed
    ├── Agent: Tax Report Writer
        Status: In Progress
        └── Using Ask question to coworker (2)
    └── Agent: Tax Report Writer
        Status: Completed
                                  Task Completion
Task Completed
Name: 787a0c0f-bd4d-4704-9225-73a758792fed
Agent: Tax Report Writer
```

  - Agent 2 (Report Generation) creates a detailed tax report based on the same input.

```
Crew: crew
└── Task: d28b7643-2a47-4677-bbe5-61625bd3b51f
    Status: Executing Task...
    └── Agent: Tax Calculation Specialist
        Status: Completed

Crew: crew
└── Task: d28b7643-2a47-4677-bbe5-61625bd3b51f
    Assigned to: Tax Calculation Specialist
    Status: Completed
    └── Agent: Tax Calculation Specialist
        Status: Completed
                                  Task Completion
Task Completed
Name: d28b7643-2a47-4677-bbe5-61625bd3b51f
Agent: Tax Calculation Specialist
```

  - **Result Handling**:
    - Agent 1's JSON output is parsed by Flask to extract the message field, which is passed to the template for display.
    - Agent 2's output is saved as a text file in the main directory with a filename as Tax_report.md.

- **Response**: The front-end renders the tax summary and, if a report is generated, provides links to view or download it.

This flow ensures seamless interaction between the user interface and the AI-driven back-end, with data processed and returned efficiently.

User Interface and Back-End Interaction

- **Front-End**:
  - Responsive HTML form gathers inputs with validation, initially hidden and activated by "Calculate Tax" button.
  - Presents tax summary using Jinja2, with "View" and "Download" options for reports.
- **Back-End**:
  - Flask routes include / (handles form submission and results) and /download/<Tax_report> (delivers reports).
  - Orchestrates CrewAI agents (Agent 1 for tax, Agent 2 for reports) and leverages Jinja2 for dynamic UI updates.
- **Interaction**:
  - Form data triggers CrewAI processing via Flask, updating the UI with results; download_report facilitates report retrieval

Tax Calculation Engine

The tax calculation engine leverages a CrewAI agents (Agent 1) to compute federal income tax based on 2025 U.S. rules. Key operations include:

- It starts by collecting simple details like income, filing status, deduction type, tax withheld, and number of dependents.
- Then it applies the correct standard deduction based on filing status (like $15,000 for Single, $30,000 for Married Filing Jointly).
- It calculates taxable income by subtracting deductions from total income and applies the appropriate progressive tax brackets.
- If the user has dependents, it adds $2,000 credit per dependent, lowering the total tax burden.
- After that, it checks how much tax was already withheld by the employer to see whether the user gets a refund or owes more.
- Finally, it wraps everything up in a clean JSON output, containing a friendly message and detailed step-by-step calculations.

Data Security

As a prototype, data security is implemented at a foundational level with future enhancements in mind:

- **Input Validation**: Client-side validation ensures required fields and numeric inputs while the back end converts data to floats, mitigating parsing errors.
- **Data Handling**: Inputs are processed in-memory without persistent storage, and reports are saved locally in the main directory, restricted to the server environment.
- **Limitations**: Lacks encryption, authentication, and secure transmission (e.g., HTTPS), posing risks to sensitive data over unsecured networks in a production setting.
- **Future Considerations**: A production version will incorporate HTTPS, user authentication, and encryption for data protection. Compliance with IRS e-filing standards, GDPR, and proper user data handling will be ensured through data anonymization and audit logging.

Limitations and Improvements

- **Improvements**: Future enhancements could include:
  - Named Entity Recognition to automatically extract data from documents, minimizing manual input.
  - Use of MCP Server to directly populate a 1040 form with calculated data.
  - Integration with IRS systems to provide live tax details for enhanced agent accuracy.
- **Limitations**: The prototype is constrained by its reliance on static 2025 tax rules, lacking real-time updates, and does not support multi-user concurrency, limiting scalability in a shared environment.

Future Enhancements

To make the AI Tax Agent even more powerful and production-ready, future work includes:

- Supporting itemized deductions for a more personalized tax return
- Adding secure data handling and encryption for user privacy
- Dockerizing the entire application for easier deployment across environments

Tools, Libraries and Frameworks

- **Python 3.11**: Core language for back-end logic.
- **Flask**: Web framework for routing and UI integration.
- **CrewAI**: Framework for AI agent orchestration (tax calculation and reporting).
- **HTML/CSS**: Front-end technologies for the user interface.
- **Jinja2**: Templating engine for dynamic HTML rendering.

Project Directory Structure

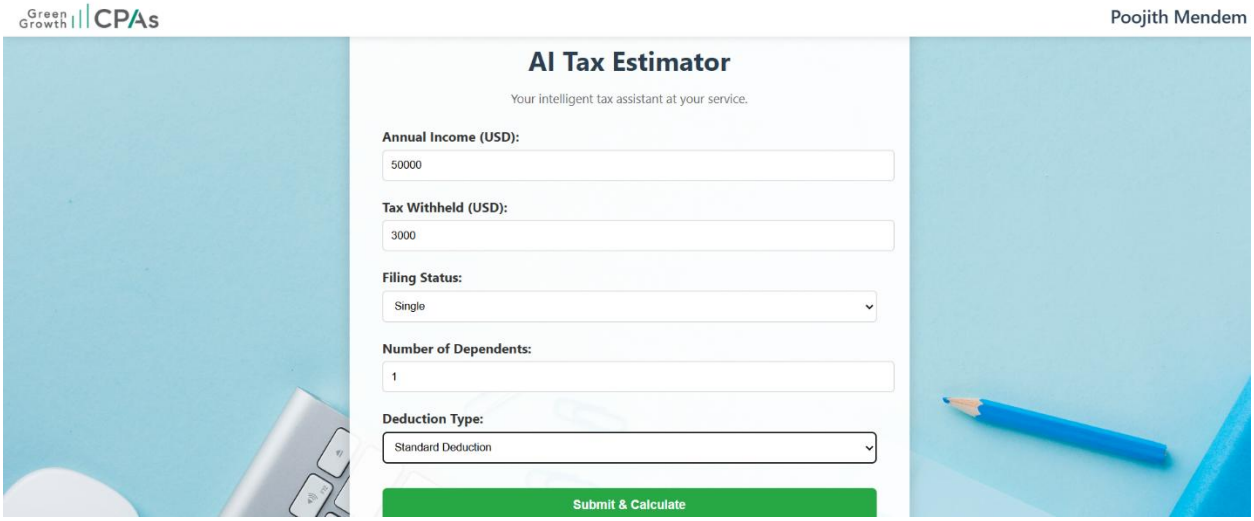
```
AI Tax Agent/
├── app.py
├── agents.py
├── task.py
├── Dockerfile
├── .env
├── requirements.txt
├── static/
│   ├── style.css
│   ├── background.jpg
│   └── logo.png
├── templates/
│   └── index.html
└── tax_report.md
```

Summary and Approach

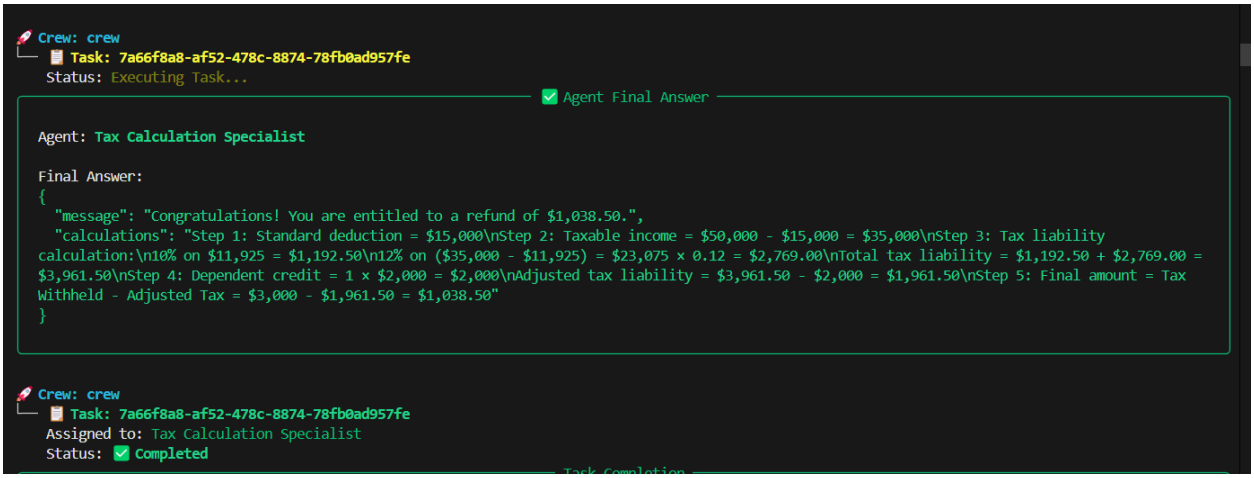
The AI Tax Agent prototype was developed by designing a modular Flask application with CrewAI agents to handle tax calculations and reporting. The approach involved building a responsive UI, implementing a tax engine with 2025 rules, and testing with diverse scenarios. This method ensured a functional prototype that meets the case study goals, with a focus on usability and AI integration, laying the groundwork for future scalability

Demo

Clean and interactive frontend where users enter income, tax withheld, filing status, dependents, and deduction type.



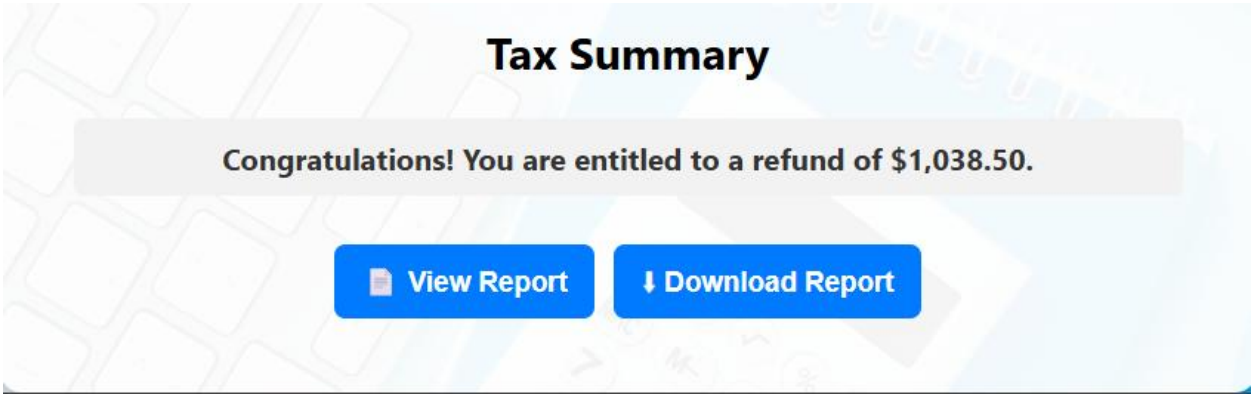
Agent 1 (TaxBot) calculates the tax step-by-step using 2025 rules and returns the result and full breakdown in JSON.



Agent 2 (ReportBot) converts the tax result into a clear, readable markdown report with all calculations explained.

```
└── 🔧 Using Ask question to coworker (1)
                                  Agent Final Answer

Agent: Tax Report Writer

Final Answer:
# U.S. Federal Tax Report

## Taxpayer Information
- **Income:** $50,000.00
- **Filing Status:** Single
- **Standard Deduction:** $15,000.00
- **Tax Withheld:** $3,000.00
- **Dependents:** 1

## Tax Calculation Steps

### Step 1: Calculate Standard Deduction
- **Standard Deduction:** $15,000.00

### Step 2: Calculate Taxable Income
- **Taxable Income:**
  \[
  \text{Income} - \text{Standard Deduction} = 50,000 - 15,000 = 35,000
  \]

### Step 3: Calculate Tax Liability
- **Tax Brackets for 2023:**
  - 10% on income up to $11,925
  - 12% on income over $11,925 up to $44,725

- **Tax Calculation:**
  - **10% Bracket:**
    \[
    10\% \text{ on } 11,925 = 0.10 \times 11,925 = 1,192.50
                                                  Ln 184, Col 1   Spaces: 2   UTF-8   CRLF   {} Python
```

Displays the final message like refund or amount owed directly on the webpage after calculation.



Detailed downloadable tax report with breakdowns, credits, and smart suggestions to increase refunds.

Tax Report

- **Income**: $50,000.00
- **Filing Status**: Single
- **Deduction Type**: Standard
- **Deduction Applied**: Based on standard deduction for selected status.
- **Tax Withheld**: $3,000.00
- **Dependents**: 1
- **Taxable Income**:
- The standard deduction of $15,000 is subtracted from the total income of $50,000.
- Thus, the taxable income is calculated as follows:

[\text{Taxable Income} = \text{Income} - \text{Standard Deduction} = 50,000 - 15,000 = 35,000]

Tax Bracket Breakdown

For the taxable income of $35,000, the tax is calculated as follows:

- **10% Bracket**:
- Income up to $11,925
- Tax Calculation: [11,925 \times 0.10 = 1,192.50]
- **12% Bracket**: