# **Feedback on the Assignment**

I enjoyed this exercise to build an end-to-end prototype to process and analyze federal regulations, and uncover and display insights for users.

**Simplification and Extension**

To keep the scope manageable during prototyping, I worked with Title 1 of the eCFR dataset. This allowed me to validate the data model, processing pipeline, and API structure. The solution is designed to be easily extendable to include the full eCFR dataset in the future with minimal changes.

**Backend**

On the backend side, I implemented logic to:

* Download and parse current eCFR data
* Store and manage the data on the server
* Build APIs to serve structured summaries for analysis
* Expose useful endpoints to support the frontend UI

The backend was central to the assignment, especially in calculating meaningful insights, including:

* Word count per agency
* Historical change tracking
* Unique checksums for data integrity

I also added a custom metric:

Average word count per section per agency,

aiming to evaluate consistency and verbosity across different agencies.

**Frontend**

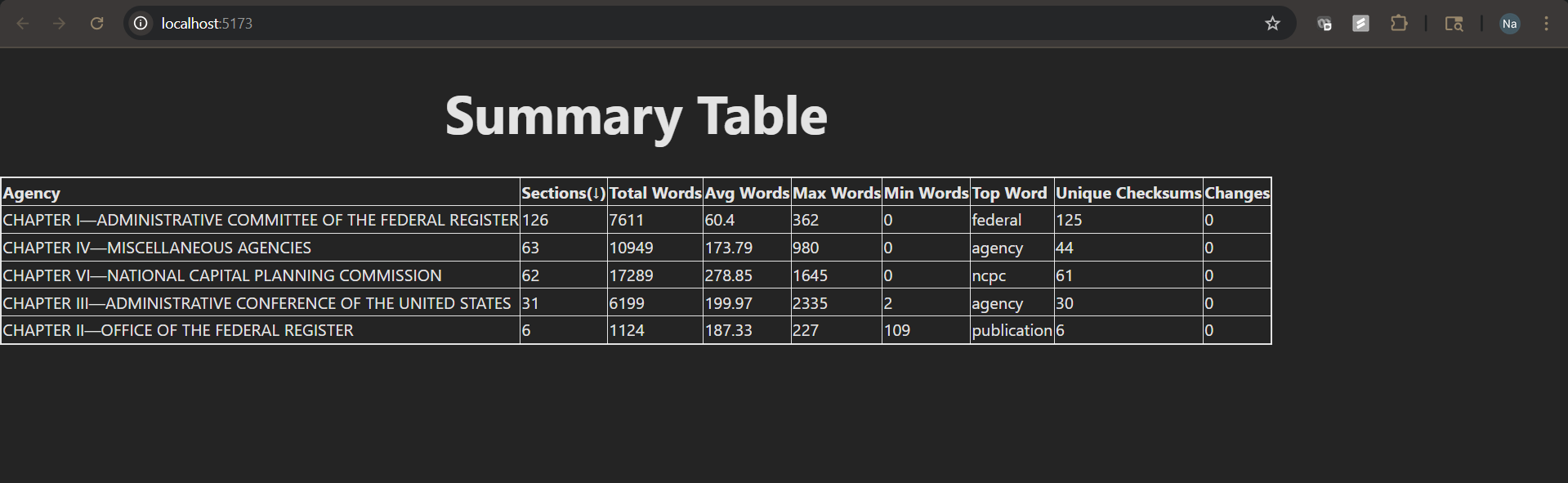
On the frontend, I built a simple UI to visualize the summary data returned from the API. I chose the React framework to implement the frontend, for its component-based architecture, ease of integration with REST APIs, and wide community support. I used Vite as the build tool to ensure fast development and hot module reloading.

# **Duration**

The task took about 12 hours, including:

* 0.5 hours: Reading and fully understanding the assessment instructions
* 2 hours: Exploring eCFR documentation to understand the data format and how to download it
* 1 hours: Data ingestion + parsing
* 3 hours: Backend API implementation
* 4 hours: Frontend UI
* 1 hour: documentation, review, and polish

# **Screenshot of UI**



# **How to run the API and UI (using Windows as an example)**

**API**

1. Open a terminal, navigate to the project root folder, and activate the Python virtual environment:

    .\venv\Scripts\Activate.ps1

2. Start the API server using Uvicorn:

    python -m uvicorn app.main:app --reload

3. Open your browser and go to: <http://127.0.0.1:8000/docs>

 You’ll see an interactive Swagger UI with three endpoints under the default tag:

GET /api/sync-xml — Fetch XML data  
GET /api/sections — Parse data into sections  
GET /api/summary — Generate summary data by agency

**UI**

1. Open a new terminal and navigate to the frontend directory:  cd front/dashboard

2. Start the React development server:   npm run dev

3. Open your browser and go to: [http://localhost:5173](http://localhost:5173/)

This will display the summary table UI, which fetches data from the API.