AI for JDs

Laura Mueller, ***City Attorney***
Aniz Alani, ***Deputy City Attorney***
City of Dripping Springs

Texas City Attorneys Association
Fall 2025 Conference
Fort Worth,

Table of Contents

Part I:
AI & IT for JDs: Practical and Low-Cost Applications of GPT and Python in Municipal Law

Aniz Alani[1]
Office of the City Attorney
City of Dripping Springs, Texas

Texas City Attorneys Association – Fall Conference

Abstract

During a legislative session, small municipal law offices face the overwhelming task of tracking hundreds of bills, analyzing their local impact, and responding before critical deadlines pass. This paper presents a phased playbook for developing a powerful, AI-assisted legislative tracking system using accessible and often free tools.

In this paper, I chronicle the journey of the City of Dripping Springs, which evolved a chaotic spreadsheet into a reliable, automated workflow that saves dozens of staff hours and prevents critical notices from being missed. By leveraging AI as a collaborative partner for scripting and process design, the city's legal team automated the ingestion of bill summaries from the Texas Municipal League (TML), delivered real-time hearing alerts via email and Slack, and even generated draft position letters aligned with council priorities.

This paper provides a replicable, step-by-step guide for non-technical lawyers to build their own systems, starting with a well-structured spreadsheet and incrementally adding automation, scheduling, and a resilient Grist database backend only as needed.

[1] J.D. (Toronto), LL.M (Brit.Col.); of the Bar of British Columbia (Non-Practicing), Texas, and Massachusetts. Substantially all of this paper was written using the Gemini Pro 2.5 GPT model based on a review of all code scripts, GitHub logs, and ChatGPT and Merlin chat session transcripts. It was fact-checked and edited by its (human) author. All responsibility for errors are the (human) author's alone.

Preface

This paper will not age well. Any "how-to" guide involving AI is doomed to become out-of-date within weeks, if not sooner. This paper describes a lengthy, iterative process that began by relying on OpenAI's ChatGPT-4o. Many of the frustrations that were caused by programming errors or logic failures would likely not have occurred if the project had begun even a few months later when OpenAI released its ChatGPT-4.5 model. And the project may well have been significantly more advanced in its scope and reliability if it had been carried out with the benefit of ChatGPT-5, Gemini Pro 2.5, and their "command line" versions, Codex and Gemini-CLI respectively, all of which were commonly available at the time of this paper's writing in October 2025.

The central thesis of this paper is not that the applications it describes are "the" answer to the tasks for which they were created to perform. Rather, the general approach – using AI tools (e.g., ChatGPT, Gemini CLI, etc.) to create scripts that systematically analyze significant amounts of evolving data and organize it into a format that's intelligible and actionable – is one that lawyers should embrace regardless of which particular AI model or programming language or platform are used to execute the task.

Incorporating AI into one's law practice is not simply a matter or doing more work faster. As this paper will describe, the use of AI tools allowed the generation of work product that would have been *impossible* to do with anything less than a large army of humans working at breakneck speed. In practical terms, this means that the work simply would not have been done at all.

1. Introduction: The Practitioner's Dilemma

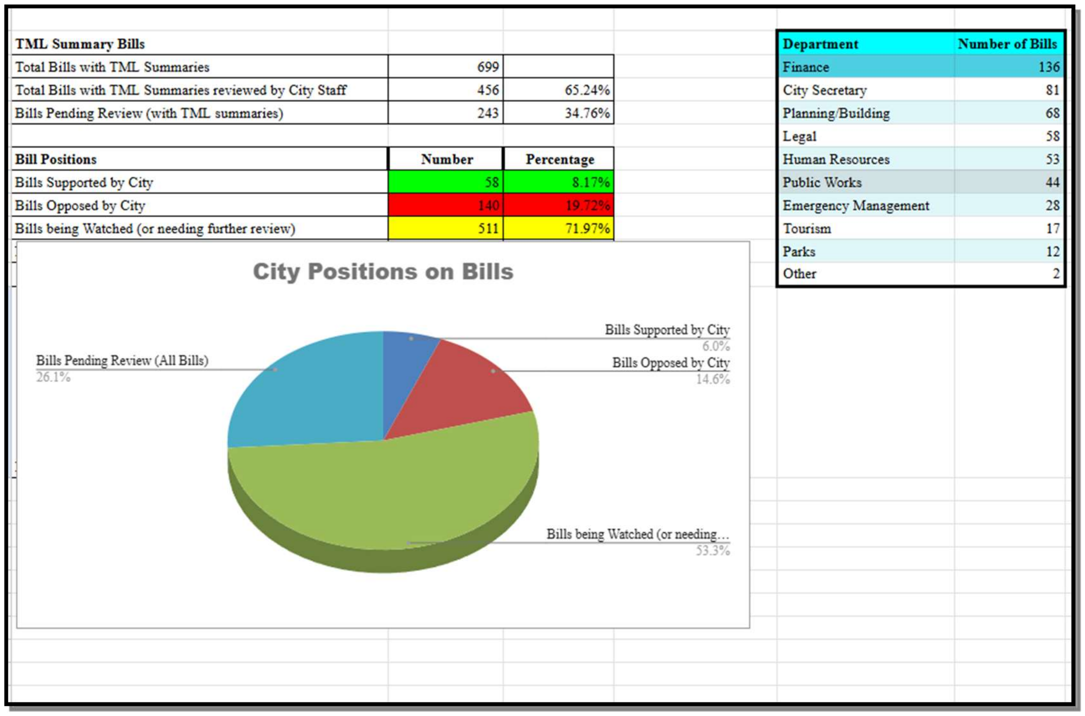
For city attorneys in small to mid-sized Texas municipalities, the start of a legislative session brings a familiar challenge: a deluge of information that far outstrips the capacity of the team assigned to manage it. New bills are filed daily, substitute versions emerge with little warning, and hearing notices are often posted with just days to spare, leaving legal staff scrambling to analyze impacts, brief leadership, and prepare testimony.

This environment creates significant operational risk, where a missed committee hearing or a misunderstood amendment can have lasting consequences for a city's finances, operations, and local authority. The core practitioner's dilemma is not a lack of information, but a profound lack of time and resources to filter, prioritize, and act on that information consistently.

Traditional methods, often centered on a shared spreadsheet and a flurry of emails, quickly break down under the strain. Staff spend hours manually copying and pasting bill summaries from various sources like the Texas Municipal League (TML) newsletter, leading to inconsistent data and version control problems. Critical hearing notices get buried in crowded inboxes, and the spreadsheet itself becomes slow and fragile as more columns, filters, and collaborators are added. This manual, reactive approach is not only inefficient but also unsustainable, creating a constant state of near-burnout for the staff tasked with the monumental job of legislative monitoring.

This paper offers a different path forward: a phased, AI-assisted approach that allows even the smallest legal teams to build a reliable, proactive legislative tracking system without hiring developers or purchasing expensive new software.

We present a detailed case study from the City of Dripping Springs, which transformed its ad-hoc tracking process into a stable, orchestrated workflow. The system now maintains a clean, centralized bill catalog, delivers timely alerts for hearings, and even helps generate draft position memos aligned with council-approved priorities. This journey demonstrates that by starting small, automating incrementally, and collaborating with AI as a research and coding partner, non-technical lawyers can create powerful tools that turn legislative chaos into actionable intelligence. This paper will provide a clear roadmap for that journey, guiding the reader from a simple spreadsheet in Phase A to a fully automated and resilient workflow in Phase D, showing how each step adds tangible value and builds a foundation for the next.



TML Summary Bills		
Total Bills with TML Summaries	699	
Total Bills with TML Summaries reviewed by City Staff	456	65.24%
Bills Pending Review (with TML summaries)	243	34.76%

Bill Positions	Number	Percentage
Bills Supported by City	58	8.17%
Bills Opposed by City	140	19.72%
Bills being Watched (or needing further review)	511	71.97%

Department	Number of Bills
Finance	136
City Secretary	81
Planning/Building	68
Legal	58
Human Resources	53
Public Works	44
Emergency Management	28
Tourism	17
Parks	12
Other	2

City Positions on Bills
Bills Supported by City 6.0%
Bills Opposed by City 14.6%
Bills Pending Review (All Bills) 26.1%
Bills being Watched (or needing… 53.3%

Figure 1- A high-level dashboard in Google Sheets provides at-a-glance intelligence, summarizing the city's position on tracked bills and the distribution of work across departments.

2. The AI Collaboration Model: A New Way to Work

The development of the legislative tracking system was fundamentally shaped by a new working model: treating generative AI as a collaborative partner rather than just a passive tool. This approach was not planned from the outset but emerged organically as the project's complexity grew. It proved to be a powerful force multiplier, enabling a non-technical lawyer to design, build, and maintain a sophisticated data pipeline that would have otherwise required a dedicated software developer or expensive consultant. The core of this model was a continuous dialogue with AI tools—primarily ChatGPT, but also others like GitHub Copilot and Merlin—to brainstorm solutions, generate code, and debug problems in real-time.

The journey began with simple, high-level questions aimed at overcoming the initial "blank page" problem that often stalls technical projects. Instead of trying to learn Python from scratch, the initial prompts focused on the desired outcome, such as, "What are the best tools for tracking legislative bills, and how can I move my existing Google Sheet into a more automated workflow?" The AI's response provided a landscape of options and, crucially, generated the first runnable Python script to interact with the LegiScan API. This initial piece of code, while imperfect, served as a tangible scaffold that could be iteratively refined, transforming an abstract goal into a concrete starting point and building momentum for the project.

This iterative process of refinement became the standard operational tempo. When a script failed or a new requirement emerged, the problem was articulated in plain English and pasted into the AI chat interface, along with the relevant code and any error messages. For example, when the Texas Municipal League (TML) website changed its HTML structure, breaking the summary-scraping script, a prompt like "My TML scraper is no longer finding bill summaries; here is the script and the new website HTML" yielded a corrected code snippet in minutes. This tight feedback loop of identifying a problem, describing it to an AI partner, and implementing the suggested solution dramatically lowered the barrier to entry for automation and fostered a mindset of continuous improvement rather than frustration. This collaborative dynamic is a key lesson from the project: for lawyers who can clearly define a problem and evaluate a proposed solution, AI can effectively bridge the technical gap, making sophisticated automation accessible without deep programming expertise.

3. Case Studies in AI-Assisted Debugging

The AI collaboration model is most powerful when applied to the real-world challenges of debugging and refining code. The development of the legislative tracking system produced several clear examples of this iterative process, where a problem was identified, presented to an AI, and solved in a matter of minutes.

(a) Correcting a TML Web Scraper

The script designed to scrape bill summaries from the Texas Municipal League (TML) newsletter was effective but brittle. When TML updated their website's HTML structure, the script, which was looking for specific HTML tags, could no longer find the data and failed. The error was presented to an AI with the following prompt: "My TML scraper is no longer finding bill summaries; here is the script and the new website HTML." The AI analyzed the new HTML, identified the changed tags, and provided a corrected Python snippet that used the updated selectors. This turned a potentially project-halting issue into a minor, ten-minute fix.

(b) Refining Google Sheets Formulas

Automation doesn't always involve complex Python scripts. In one instance, a Google Sheets formula designed to filter and display upcoming committee hearings was not correctly handling certain date formats. A prompt to the AI, "Could you please revise this formula? I'm looking to..." along with the problematic formula, resulted in a corrected version that properly accounted for the edge cases. This demonstrates the AI's utility as a general-purpose technical assistant, not just a Python coder.

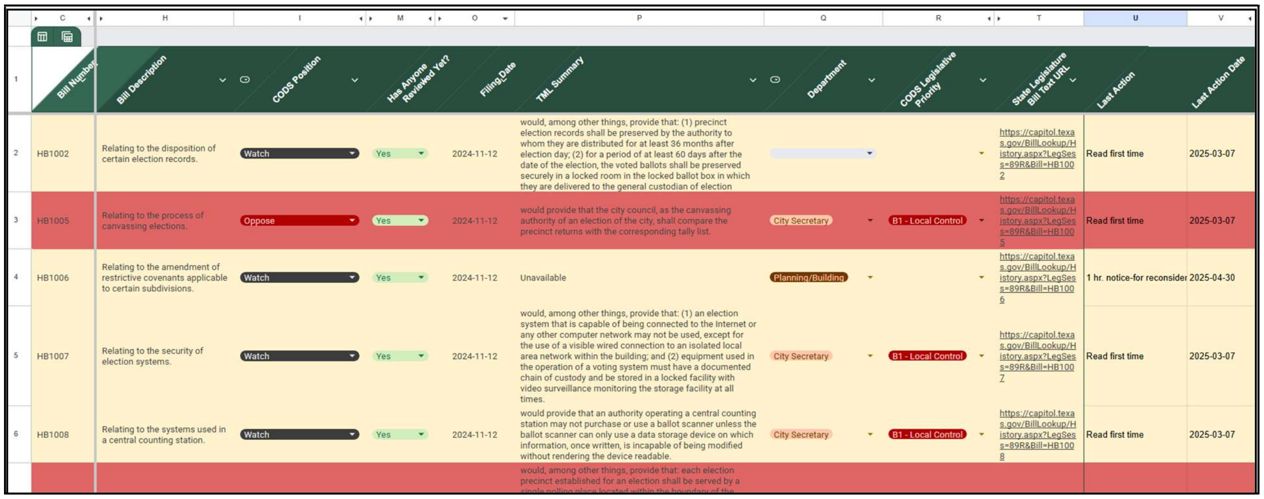
(c) GitHub Actions Permission Errors

When the automation scripts were moved to GitHub Actions for scheduling, they began to fail with a "permission denied" error. This was a novel issue, as the scripts ran perfectly in Google Colab. The error log was provided to the AI, which diagnosed the problem: the default permissions of the GitHub Actions runner were not sufficient to write files. The AI then provided the necessary YAML code to add to the workflow file, explicitly granting the correct permissions. This solved the issue and ensured the scheduled automations could run reliably.

4. Phase A: Establishing a Single Source of Truth

Before any automation can be effective, the underlying data must be organized, consistent, and trusted by the entire team. The first and most critical phase of the Dripping Springs project was therefore not technical but organizational: consolidating all legislative tracking activities into a single, well-structured Google Sheet. This document became the undisputed "source of truth" for the entire 89th Legislative Session, providing a central hub that replaced scattered emails, duplicative documents, and the institutional knowledge held by single individuals. Establishing this foundation proved to be the most important step in the entire process, as the clarity and discipline it imposed made all subsequent automation possible.

The structure of this central spreadsheet was intentionally simple, designed around the key questions city attorneys need to answer during a session. It contained a few core tabs: **Bills**, to catalog each piece of legislation being tracked; **Summaries**, to hold the plain-language explanations from trusted sources like TML; **Priorities**, to list the specific legislative priorities adopted by the City Council, which would be used to assign a local impact level and tentative city position (Support/Oppose/Watch); and **Calendars**, to log upcoming committee and floor hearings. By segmenting the data logically, the spreadsheet was immediately more useful than a monolithic table, allowing staff to filter bills by priority, see all upcoming hearings in one place, or review all summaries related to a specific topic. The header row in each tab was locked, and data validation rules were used for columns like "Priority" and "Position" to ensure consistent entries from all users.



	C	H	I	M	O	P	Q	R	T	U	V
1	Bill Number	Bill Description	CODS Position	Has Anyone Reviewed Yet?	Filing Date	TML Summary	Department	CODS Legislative Priority	State Legislature Bill Text URL	Last Action	Last Action Date
2	HB1002	Relating to the disposition of certain election records.	Watch	Yes	2024-11-12	would, among other things, provide that: (1) precinct election records shall be preserved by the authority to whom they are distributed for at least 36 months after election day; (2) for a period of at least 60 days after the date of the election, the voted ballots shall be preserved securely in a locked room in the locked ballot box in which they are delivered to the general custodian of election			https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB1002	Read first time	2025-03-07
3	HB1005	Relating to the process of canvassing elections.	Oppose	Yes	2024-11-12	would provide that the city council, as the canvassing authority of an election of the city, shall compare the precinct returns with the corresponding tally list.	City Secretary	B1 - Local Control	https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB1005	Read first time	2025-03-07
4	HB1006	Relating to the amendment of restrictive covenants applicable to certain subdivisions.	Watch	Yes	2024-11-12	Unavailable	Planning/Building		https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB1006	1 hr. notice-for reconsider	2025-04-30
5	HB1007	Relating to the security of election systems.	Watch	Yes	2024-11-12	would, among other things, provide that: (1) an election system that is capable of being connected to the Internet or any other computer network may not be used, except for the use of a visible wired connection to an isolated local area network within the building; and (2) equipment used in the operation of a voting system must have a documented chain of custody and be stored in a locked facility with video surveillance monitoring the storage facility at all times.	City Secretary	B1 - Local Control	https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB1007	Read first time	2025-03-07
6	HB1008	Relating to the systems used in a central counting station.	Watch	Yes	2024-11-12	would provide that an authority operating a central counting station may not purchase or use a ballot scanner unless the ballot scanner can only use a data storage device on which information, once written, is incapable of being modified without rendering the device readable.	City Secretary	B1 - Local Control	https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB1008	Read first time	2025-03-07
						would, among other things, provide that: each election precinct established for an election shall be served by a					

Figure 2 - The main "Bills" tab of the Google Sheet served as the central registry, with clear color-coding for the city's position on each piece of legislation.

This initial phase required establishing a clear manual process and the discipline to follow it, which was a crucial prerequisite for any later technical work. The team agreed on a simple daily routine, including who was responsible for checking legislative websites for new hearing notices and pasting them into the Calendars tab. This manual discipline ensures that the core data is reliable and up-to-date, preventing a "garbage in, garbage out" scenario when automation is introduced later. For any law office seeking to replicate this process, this non-technical first step is non-negotiable; a chaotic manual process will only yield chaotic automated results.

Monday Morning Checklist (Phase A)

- Create a new Google Sheet and invite the legislative tracking team with edit access.
- Build the core tabs: `Bills`, `Summaries`, `Calendars`, and `Priorities`.

- For a few high-priority bills, manually enter the basic information (bill number, author, caption) and paste in a trusted summary from a source like TML, making sure to include a source link and date.

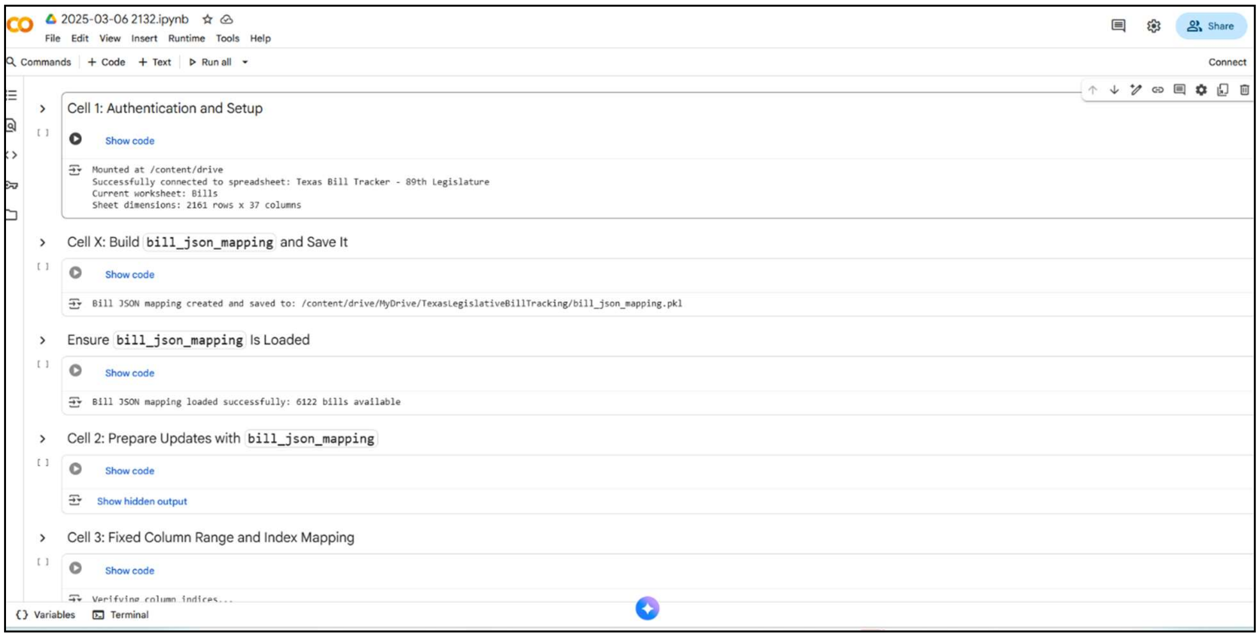
- Use simple filters to create and save useful views, such as "High Priority Bills" or "Hearings This Week," that users can understand at a glance.

- Assign and document the daily routine for checking official sources and manually updating the sheet based on newly introduced bills. (In the first phase of our project, this was a weekly exercise that coincided with the release of TML's weekly legislative update.)

5. Phase B: Light Automation with Accessible Tools

Once the single source of truth was established and the manual workflow felt stable, the project entered its second phase: introducing targeted automation to eliminate the most repetitive and time-consuming tasks. The guiding principle was to automate only what was causing the most friction, ensuring that each new script provided an immediate and tangible return on the time invested in creating it. This approach avoids the common pitfall of building complex, all-encompassing solutions and instead focuses on pragmatic wins that make the existing manual process faster and more reliable. The goal was not to replace the spreadsheet, but to feed it with clean, structured data more efficiently than a human could.

For a legal office without dedicated IT staff, the primary barrier to automation is often the complexity of setting up a programming environment. To overcome this, the project leaned heavily on Google Colab, a free, browser-based tool that allows anyone to write and run Python code without installing any software. Colab notebooks provide an interactive, step-by-step format that is perfect for non-programmers, as code can be run in small, manageable cells with explanatory text and comments included alongside. This accessible environment became the proving ground for all early automations, allowing for rapid experimentation and making it easy to share a working script with a colleague—all they needed was a web browser and a link.



Cell 1: Authentication and Setup

Show code

```
Mounted at /content/drive
Successfully connected to spreadsheet: Texas Bill Tracker - 89th Legislature
Current worksheet: Bills
Sheet dimensions: 2161 rows x 37 columns
```

Cell X: Build `bill_json_mapping` and Save It

Show code

```
Bill JSON mapping created and saved to: /content/drive/MyDrive/TexasLegislativeBillTracking/bill_json_mapping.pkl
```

Ensure `bill_json_mapping` Is Loaded

Show code

```
Bill JSON mapping loaded successfully: 6122 bills available
```

Cell 2: Prepare Updates with `bill_json_mapping`

Show code

Show hidden output

Cell 3: Fixed Column Range and Index Mapping

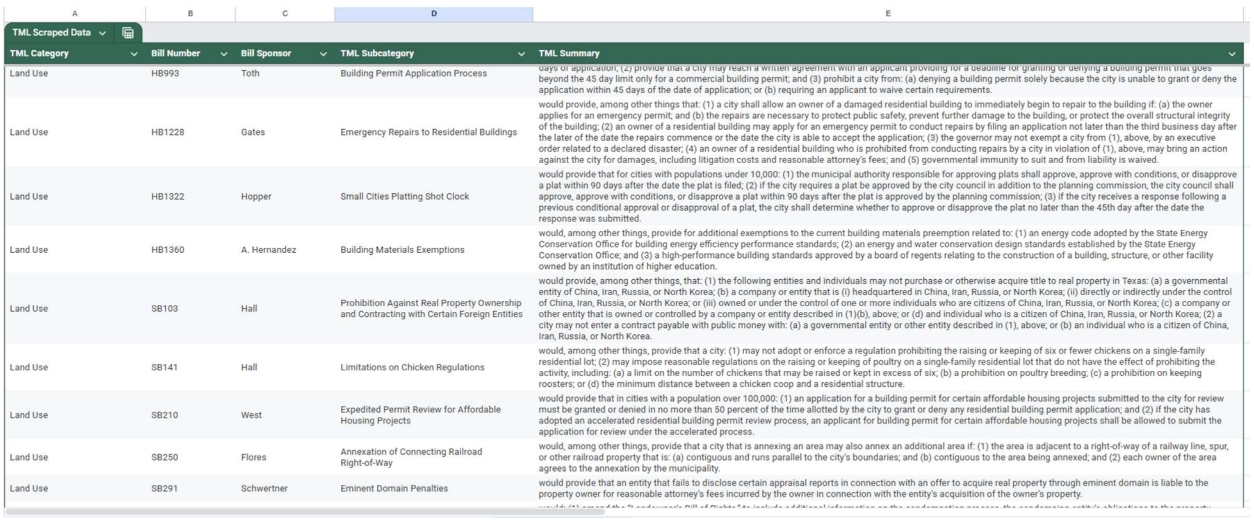
Show code

```
Verifying column indices...
```

Figure 3 - Google Colab provides an interactive, browser-based environment where Python scripts can be run cell by cell, making the automation process accessible to non-developers.

The first major automation targeted the weekly task of processing the Texas Municipal League (TML) Legislative Update newsletter. Manually, this involved a staff member opening the newsletter, identifying relevant bills, and meticulously copying and pasting each summary into the `Summaries` tab of the Google Sheet—a process that was both tedious and prone to error.

Using the AI collaboration model, a Python script was developed to handle this entire workflow automatically. The script was designed to check the TML website for the latest newsletter URL, download the page's HTML content, and then parse it to extract each bill number, its associated summary, and any category information provided by TML. The output was a clean, structured dataset (like a CSV file) that could be easily uploaded or pasted into the Google Sheet, turning an hour of manual work into a few minutes of execution time.



TML Scraped Data

TML Category	Bill Number	Bill Sponsor	TML Subcategory	TML Summary
Land Use	HB993	Toth	Building Permit Application Process	days of application; (2) provide that a city may reach a written agreement with an applicant providing for a deadline for granting or denying a building permit that goes beyond the 45 day limit only for a commercial building permit; and (3) prohibit a city from: (a) denying a building permit solely because the city is unable to grant or deny the application within 45 days of the date of application; or (b) requiring an applicant to waive certain requirements.
Land Use	HB1228	Gates	Emergency Repairs to Residential Buildings	would provide, among other things that: (1) a city shall allow an owner of a damaged residential building to immediately begin to repair to the building if: (a) the owner applies for an emergency permit; and (b) the repairs are necessary to protect public safety, prevent further damage to the building, or protect the overall structural integrity of the building; (2) an owner of a residential building may apply for an emergency permit to conduct repairs by filing an application not later than the third business day after the later of the date the repairs commence or the date the city is able to accept the application; (3) the governor may not exempt a city from (1), above, by an executive order related to a declared disaster; (4) an owner of a residential building who is prohibited from conducting repairs by a city in violation of (1), above, may bring an action against the city for damages, including litigation costs and reasonable attorney's fees; and (5) governmental immunity to suit and from liability is waived.
Land Use	HB1322	Hopper	Small Cities Platting Shot Clock	would provide that for cities with populations under 10,000: (1) the municipal authority responsible for approving plats shall approve, approve with conditions, or disapprove a plat within 90 days after the date the plat is filed; (2) if the city requires a plat be approved by the city council in addition to the planning commission, the city council shall approve, approve with conditions, or disapprove a plat within 90 days after the plat is approved by the planning commission; (3) if the city receives a response following a previous conditional approval or disapproval of a plat, the city shall determine whether to approve or disapprove the plat no later than the 45th day after the date the response was submitted.
Land Use	HB1360	A. Hernandez	Building Materials Exemptions	would, among other things, provide for additional exemptions to the current building materials preemption related to: (1) an energy code adopted by the State Energy Conservation Office for building energy efficiency performance standards; (2) an energy and water conservation design standards established by the State Energy Conservation Office; and (3) a high-performance building standards approved by a board of regents relating to the construction of a building, structure, or other facility owned by an institution of higher education.
Land Use	SB103	Hall	Prohibition Against Real Property Ownership and Contracting with Certain Foreign Entities	would provide, among other things, that: (1) the following entities and individuals may not purchase or otherwise acquire title to real property in Texas: (a) a governmental entity of China, Iran, Russia, or North Korea; (b) a company or entity that is (i) headquartered in China, Iran, Russia, or North Korea; (ii) directly or indirectly under the control of China, Iran, Russia, or North Korea; or (iii) owned or under the control of one or more individuals who are citizens of China, Iran, Russia, or North Korea; (c) a company or other entity that is owned or controlled by a company or entity described in (1)(b), above; or (d) and individual who is a citizen of China, Iran, Russia, or North Korea; (2) a city may not enter a contract payable with public money with: (a) a governmental entity or other entity described in (1), above; or (b) an individual who is a citizen of China, Iran, Russia, or North Korea.
Land Use	SB141	Hall	Limitations on Chicken Regulations	would, among other things, provide that a city: (1) may not adopt or enforce a regulation prohibiting the raising or keeping of six or fewer chickens on a single-family residential lot; (2) may impose reasonable regulations on the raising or keeping of poultry on a single-family residential lot that do not have the effect of prohibiting the activity, including: (a) a limit on the number of chickens that may be raised or kept in excess of six; (b) a prohibition on poultry breeding; (c) a prohibition on keeping roosters; or (d) the minimum distance between a chicken coop and a residential structure.
Land Use	SB210	West	Expedited Permit Review for Affordable Housing Projects	would provide that in cities with a population over 100,000: (1) an application for a building permit for certain affordable housing projects submitted to the city for review must be granted or denied in no more than 50 percent of the time allotted by the city to grant or deny any residential building permit application; and (2) if the city has adopted an accelerated residential building permit review process, an applicant for building permit for certain affordable housing projects shall be allowed to submit the application for review under the accelerated process.
Land Use	SB250	Flores	Annexation of Connecting Railroad Right-of-Way	would, among other things, provide that a city that is annexing an area may also annex an additional area if: (1) the area is adjacent to a right-of-way of a railway line, spur, or other railroad property that is: (a) contiguous and runs parallel to the city's boundaries; and (b) contiguous to the area being annexed; and (2) each owner of the area agrees to the annexation by the municipality.
Land Use	SB291	Schwertner	Eminent Domain Penalties	would provide that an entity that fails to disclose certain appraisal reports in connection with an offer to acquire real property through eminent domain is liable to the property owner for reasonable attorney's fees incurred by the owner in connection with the entity's acquisition of the owner's property.

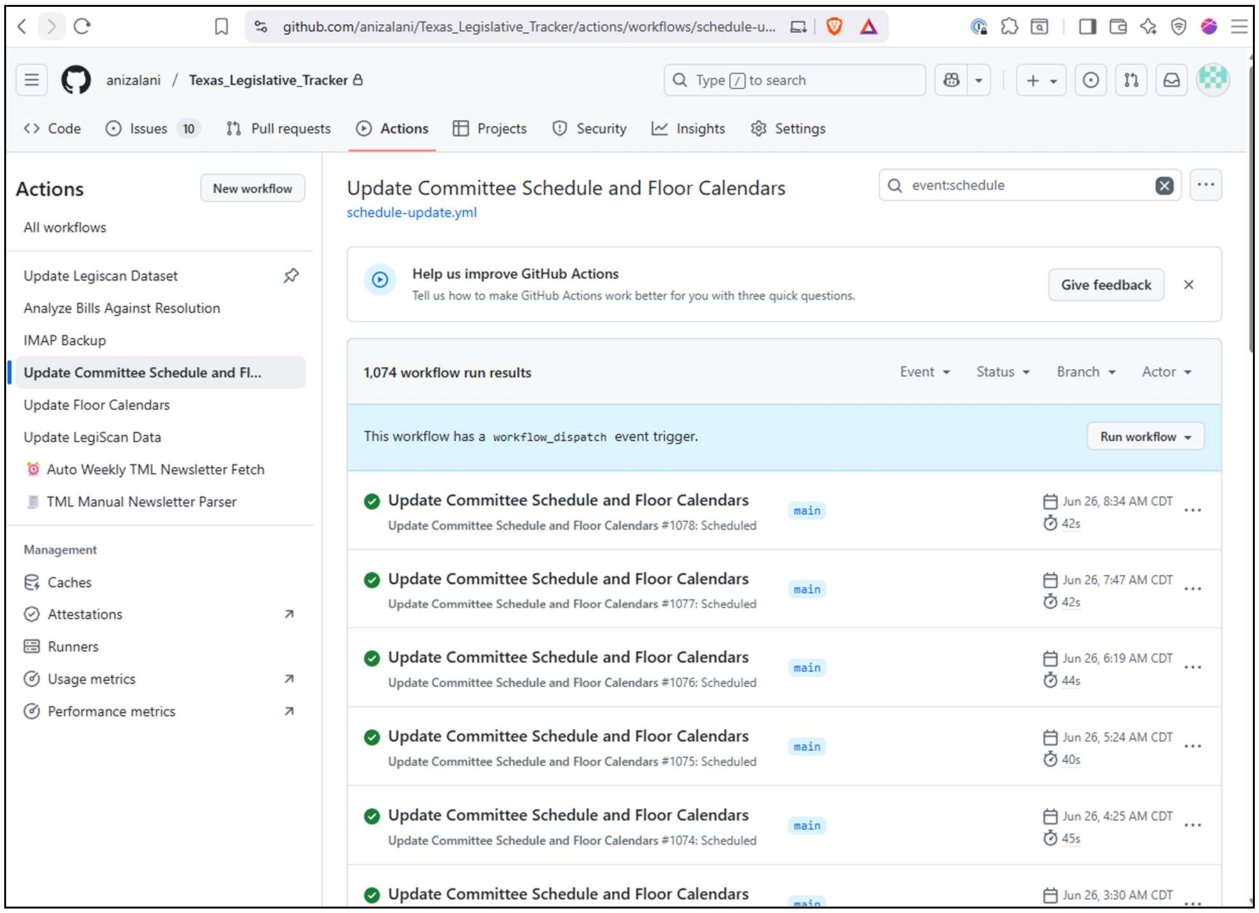
Figure 4 - The result of the TML scraping script—a clean, structured table of bill summaries, sponsors, and categories, ready for analysis.

A second critical automation was developed to monitor the Texas Legislature Online (TLO) website for hearing and floor calendar postings. Missing a hearing notice is one of the biggest risks in legislative tracking, and the manual process of checking multiple pages on the TLO site each morning was time-consuming. A dedicated scraper script was created to automate this surveillance. It would navigate to the relevant calendar pages, extract the agenda details for upcoming hearings, and identify any bills already being tracked in the master Google Sheet. This script provided an early warning system, flagging when a high-priority bill was scheduled for a hearing and ensuring the legal team had as much time as possible to prepare testimony or a position letter. Like the TML script, this automation fed its findings directly back into the `Calendars` tab of the central spreadsheet, enriching the team's single source of truth without requiring them to change their core workflow.

6. Phase C: Set-and-Forget Scheduling with GitHub Actions

While the light automations of Phase B were a significant improvement, they still required a person to manually open a Google Colab notebook and click "Run" to execute the scripts. This introduced a potential point of failure; if the designated person was busy, sick, or simply forgot, the updates wouldn't happen, and the team would be flying blind. The goal of Phase C was to remove this dependency on manual execution and create a truly "set-and-forget" system that would run the automations on a reliable, predictable schedule. This transition from interactive scripts to a fully orchestrated workflow marked a major leap in the system's maturity and dependability.

The project found its scheduling solution in GitHub Actions, a free and powerful automation platform built into the GitHub source code repository service. While GitHub is primarily known for hosting code, its Actions feature allows users to run scripts and workflows on servers in the cloud based on a schedule or other triggers. This was the perfect tool for the legislative tracking system, as it provided a robust, serverless environment to execute the Python scripts without requiring the city to maintain its own server. A simple configuration file was added to the project's GitHub repository, instructing the platform to run the TML newsletter scraper every Friday afternoon and the TLO calendar scrapers every few hours, seven days a week.



Actions

All workflows

- Update Legiscan Dataset
- Analyze Bills Against Resolution
- IMAP Backup
- Update Committee Schedule and Fl...
- Update Floor Calendars
- Update LegiScan Data
- Auto Weekly TML Newsletter Fetch
- TML Manual Newsletter Parser

Management

- Caches
- Attestations
- Runners
- Usage metrics
- Performance metrics

Update Committee Schedule and Floor Calendars
schedule-update.yml

event:schedule

Help us improve GitHub Actions
Tell us how to make GitHub Actions work better for you with three quick questions.

1,074 workflow run results

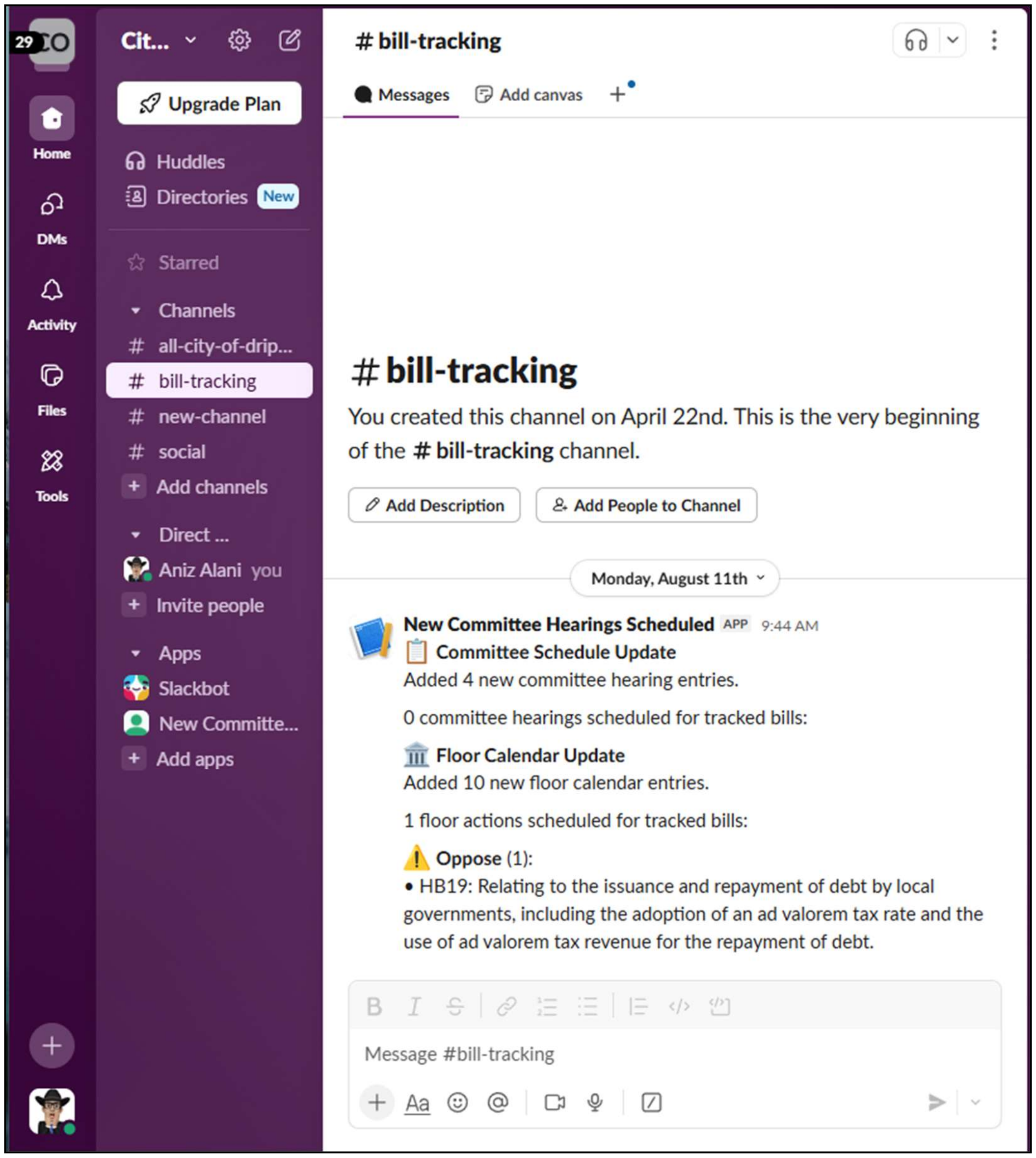
This workflow has a `workflow_dispatch` event trigger.

✓ Update Committee Schedule and Floor Calendars Update Committee Schedule and Floor Calendars #1078: Scheduled	main	Jun 26, 8:34 AM CDT	42s
✓ Update Committee Schedule and Floor Calendars Update Committee Schedule and Floor Calendars #1077: Scheduled	main	Jun 26, 7:47 AM CDT	42s
✓ Update Committee Schedule and Floor Calendars Update Committee Schedule and Floor Calendars #1076: Scheduled	main	Jun 26, 6:19 AM CDT	44s
✓ Update Committee Schedule and Floor Calendars Update Committee Schedule and Floor Calendars #1075: Scheduled	main	Jun 26, 5:24 AM CDT	40s
✓ Update Committee Schedule and Floor Calendars Update Committee Schedule and Floor Calendars #1074: Scheduled	main	Jun 26, 4:25 AM CDT	45s
✓ Update Committee Schedule and Floor Calendars	main	Jun 26, 3:30 AM CDT	

Figure 5 - The GitHub Actions interface provides a detailed log of every automated run, offering transparency and proof that the system is operating reliably around the clock.

Migrating the scripts to this new environment also necessitated a more professional approach to managing sensitive information, such as API keys for services like LegiScan. Instead of pasting these credentials directly into a Colab notebook, they were stored as encrypted "Secrets" within the GitHub repository's settings. This is a critical security practice that ensures that private keys are never exposed in the code itself, allowing the project to be shared and managed without compromising its credentials. The scripts were modified to securely access these secrets at runtime, a small but vital change that made the automated workflow both secure and portable.

The final piece of the scheduled workflow was creating an automated alert system to deliver the results to the team. After the scraper scripts finished running, a subsequent step in the GitHub Action would trigger a notification summarizing what had changed. These alerts were sent via both email and a dedicated Slack channel, providing a concise digest such as: "3 new hearings posted for tomorrow; 1 high-priority bill has a new substitute version." Each item in the alert included a direct link to the relevant filtered view in the master Google Sheet, allowing staff to go from the notification to the specific data with a single click. As documented in the project's execution logs, this automated system achieved a remarkable success rate of over 97% across more than a thousand runs, ensuring that timely, critical legislative updates were never missed, regardless of any single team member's availability.



bill-tracking

You created this channel on April 22nd. This is the very beginning of the # bill-tracking channel.

New Committee Hearings Scheduled APP 9:44 AM
📋 **Committee Schedule Update**
Added 4 new committee hearing entries.

0 committee hearings scheduled for tracked bills:

🏛️ **Floor Calendar Update**
Added 10 new floor calendar entries.

1 floor actions scheduled for tracked bills:

⚠️ **Oppose** (1):
• HB19: Relating to the issuance and repayment of debt by local governments, including the adoption of an ad valorem tax rate and the use of ad valorem tax revenue for the repayment of debt.

Figure 6 - Automated alerts delivered to a dedicated Slack channel provide a real-time summary of new hearings and floor actions for tracked bills.

7. AI as a Strategic Partner for Substantive Analysis

Once a reliable system for tracking bills is in place, the focus shifts from data collection to substantive analysis and strategic response. The AI collaboration model extends far beyond simply writing the code for the system; it becomes a powerful partner in the day-to-day work of a city attorney's office. This was demonstrated in two distinct case studies during the 89th session, one involving rapid impact assessment and the other involving in-depth policy development.

(a) Case Study: Rapid Qualitative Impact Analysis of HB 150

Not every bill requires a deep, multi-day analysis. Often, the need is for a quick, high-level assessment to determine if a bill warrants further attention. When HB 150, a bill proposing the creation of a new Texas Cyber Command, was filed, a simple query was posed to an AI assistant to determine its potential effects on the City of Dripping Springs.

In seconds, the AI distilled the bill's dense text into a concise summary of practical implications. It identified that the new command would likely provide the city with enhanced cybersecurity support for critical infrastructure, but would also introduce new compliance and training standards. This rapid analysis allowed the legal team to immediately grasp the bill's operational impact and flag it for future monitoring without a time-consuming manual review.

(b) Case Study: In-Depth Policy Development for SB 542

A more complex and illustrative example of AI partnership arose with SB 542, a bill concerning the ability of Property Owners' Associations (POAs) to levy fines during mandatory water restrictions. What began as a simple request to prepare a position memo evolved into an iterative, collaborative dialogue with an AI assistant that spanned the entire policy response lifecycle.

1. **Initial Analysis and Flaw Identification:** The AI first analyzed the bill and immediately identified its critical flaw: the failure to define a "reasonable period" for landscaping to recover after restrictions were lifted, creating a significant risk of litigation.

2. **Brainstorming and "What If" Scenarios:** The dialogue then shifted to policy development. The AI was prompted to explore the implications of a potential amendment that would allow cities to define the recovery period themselves. It analyzed the benefits (local flexibility) and the challenges (administrative burden), and even proposed specific legislative language for the amendment.

3. **Iterative Document Drafting:** Over a series of prompts, the AI was guided to draft, refine, and reformat a comprehensive position memo for a council member's testimony. The process moved from bullet points to full-sentence prose, incorporating feedback at each stage to build the final document.

4. **Communications and Media Strategy:** Finally, the AI was asked to generate "pithy media lines" and soundbites based on the memo. It produced a list of concise, memorable statements designed to clearly communicate the city's position to the media and the public (e.g., *"In a drought, green grass isn't the priority—ensuring sufficient water for future generations is."*).

This case study demonstrates the maturation of the AI collaboration model from a tool for overcoming technical hurdles to a genuine partner in strategic legal work. It allowed a small legal office to move with speed and depth, from initial bill analysis to legislative amendment strategy and public communications, in a fraction of the time it would have taken traditionally.

8. Post-Session Evolution: Addressing the Limits of Spreadsheets with Grist

By the end of the 89th Legislative Session, the automated workflow centered on Google Sheets had proven to be a resounding success. It had reliably tracked thousands of bills, processed dozens of newsletters, and delivered hundreds of timely alerts, all while operating autonomously on a GitHub Actions schedule. However, the high-pressure environment of the session also exposed the inherent limitations of using a spreadsheet as the system's central database. As the volume of data grew into thousands of rows across multiple tabs, the Google Sheet became noticeably sluggish, filters took longer to apply, and the risk of data corruption or accidental deletion by a collaborator became a significant concern.

The most pressing technical challenge was the difficulty of maintaining clean, relational data within the flat structure of a spreadsheet. For example, linking a single bill from the `Bills` tab to its multiple summaries in the `Summaries` tab and its various hearing dates in the `Calendars` tab required complex and fragile formulas. This complexity made the system difficult to query for anything beyond simple filters and created a constant need for data hygiene scripts to fix broken links and remove duplicate entries. It became clear that while the spreadsheet was the perfect tool to get the system up and running quickly, a more robust, database-centric solution was needed to ensure the project's long-term viability and scalability beyond a single legislative session.

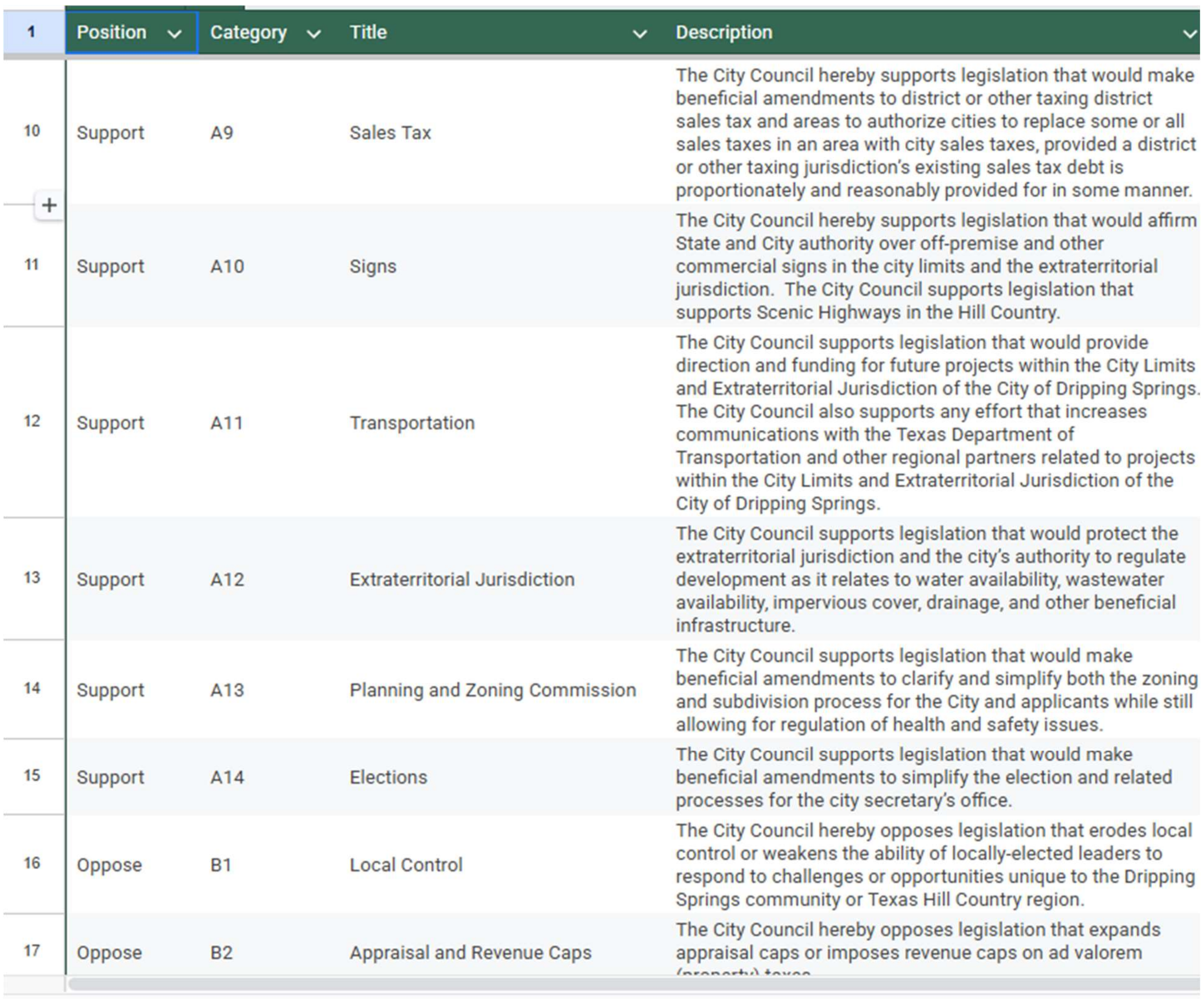
In the period following the regular session, the project entered its next phase of evolution: migrating the data and workflow to Grist. Grist was selected because it uniquely combines the familiar, user-friendly interface of a spreadsheet with the powerful backend of a true relational database. This hybrid approach allows non-technical users to view, sort, and filter data in a comfortable grid format, while the underlying structure supports robust relationships between tables, preventing data duplication and ensuring integrity. The migration involved creating a formal schema that defined the relationships between Bills, Summaries, Calendars, and other data tables, something that was only implicitly handled in the spreadsheet.

To automate the creation of this new home, a dedicated Python script (`grist_setup.py`) was developed to provision the entire database structure programmatically via the Grist API. This script creates all the necessary tables, defines the columns and their types (e.g., text, date, reference), and establishes the crucial links between them, such as connecting a record in the `TML_Data` table back to the master `Bills` table. This "infrastructure-as-code" approach ensures that the database can be recreated perfectly at any time, providing a level of stability and predictability that a manually configured spreadsheet could never match. While this Grist-based system has not yet been rolled out to the broader team, it represents the next logical step in the project's maturation, creating a session-agnostic foundation that will be faster, more reliable, and far more scalable for the 90th Legislature and beyond.

9. Advanced Automations: From Data to Actionable Intelligence

With a reliable and continuously updated stream of legislative data flowing into the central tracking system, the project was able to move beyond simple monitoring and develop advanced automations that turned raw data into actionable legal intelligence. These workflows were designed to directly support the substantive work of the city attorney's office, helping to analyze the local impact of proposed legislation and accelerate the process of drafting official responses. This represented a significant evolution from merely tracking bills to actively engaging with them, all driven by the clean data foundation that had been established in the earlier phases of the project. These tools provided direct leverage for the legal team, saving time on analysis and allowing attorneys to focus on strategy rather than administrative overhead.

One of the most powerful of these advanced tools envisioned but not fully developed for use was a script designed to automate the initial drafting of position letters. When the city council adopts its legislative priorities, those policies are often articulated in high-level resolutions. To bridge the gap between these broad policy statements and the specific language of a newly filed bill, a Python script would use a text-analysis technique called TF-IDF (Term Frequency–Inverse Document Frequency). This script compares the text of a bill against the city's approved policy documents and calculates a similarity score, allowing it to flag bills that are either strongly aligned with or strongly opposed to the city's stated positions. This automated analysis would provide a crucial first-pass filter, highlighting the most impactful bills and seeding the creation of a draft position letter, which could then be quickly refined by an attorney.



1	Position	Category	Title	Description
10	Support	A9	Sales Tax	The City Council hereby supports legislation that would make beneficial amendments to district or other taxing district sales tax and areas to authorize cities to replace some or all sales taxes in an area with city sales taxes, provided a district or other taxing jurisdiction's existing sales tax debt is proportionately and reasonably provided for in some manner.
11	Support	A10	Signs	The City Council hereby supports legislation that would affirm State and City authority over off-premise and other commercial signs in the city limits and the extraterritorial jurisdiction. The City Council supports legislation that supports Scenic Highways in the Hill Country.
12	Support	A11	Transportation	The City Council supports legislation that would provide direction and funding for future projects within the City Limits and Extraterritorial Jurisdiction of the City of Dripping Springs. The City Council also supports any effort that increases communications with the Texas Department of Transportation and other regional partners related to projects within the City Limits and Extraterritorial Jurisdiction of the City of Dripping Springs.
13	Support	A12	Extraterritorial Jurisdiction	The City Council supports legislation that would protect the extraterritorial jurisdiction and the city's authority to regulate development as it relates to water availability, wastewater availability, impervious cover, drainage, and other beneficial infrastructure.
14	Support	A13	Planning and Zoning Commission	The City Council supports legislation that would make beneficial amendments to clarify and simplify both the zoning and subdivision process for the City and applicants while still allowing for regulation of health and safety issues.
15	Support	A14	Elections	The City Council supports legislation that would make beneficial amendments to simplify the election and related processes for the city secretary's office.
16	Oppose	B1	Local Control	The City Council hereby opposes legislation that erodes local control or weakens the ability of locally-elected leaders to respond to challenges or opportunities unique to the Dripping Springs community or Texas Hill Country region.
17	Oppose	B2	Appraisal and Revenue Caps	The City Council hereby opposes legislation that expands appraisal caps or imposes revenue caps on ad valorem (property) taxes

Figure 7 - The city's official legislative priorities, stored in a structured format, serve as the source material for automated bill analysis and letter drafting.

Another high-impact automation was created to help users visualize the real-world impact of bills that dealt with geographic boundaries, such as those affecting annexation, zoning authority, or extraterritorial jurisdiction (ETJ). Abstract legislative language describing setbacks, buffers, and service areas can be difficult to interpret, but a map makes the consequences immediately clear. A reproducible GIS (Geographic Information System) workflow was developed using a Python script (`make_map.py`) that could take the parameters from a bill—such as a 500-foot buffer around a specific type of waterway—and apply them to the city's official boundary and hydrology shapefiles. The script would then generate a clear, publication-ready map highlighting the specific parcels or areas within the city that would be affected by the proposed law. This tool proved invaluable for briefing councilmembers on complex bills like HB 2494 during the 89th session, transforming a dense legal clause into a single, intuitive image that clarified the stakes instantly.

10. Conclusion: A Reproducible Playbook for Texas Cities

The Texas legislative cycle will always be a fast-moving and demanding environment, but the experience of the City of Dripping Springs demonstrates that small city attorney offices can not only keep pace but can do so with a level of efficiency and accuracy that was previously out of reach. By embracing a thoughtful blend of simple, accessible tools and targeted automation, the city’s legal team transformed its legislative monitoring from a reactive, manual chore into a proactive, data-driven process. This journey, from a single well-organized spreadsheet to a fully automated workflow, proves that it is possible to build a powerful system without a dedicated IT department or a significant budget. The key ingredients were a commitment to a phased approach, a willingness to experiment, and the strategic use of AI as a collaborative partner to bridge the technical gap.

The playbook is intentionally incremental, designed to be adopted one step at a time. It begins not with code, but with the organizational discipline of establishing a single source of truth that the entire team can rely on. From there, each subsequent phase—introducing light automation with Google Colab, scheduling tasks with GitHub Actions, and eventually graduating to a more robust Grist database—builds upon the last, adding value without disrupting the core workflow. This model mitigates risk and allows an office to find its own "good enough" point, whether that is a simple spreadsheet fed by a few key scripts or a fully orchestrated, database-driven system.

Ultimately, the most significant shift is a cultural one: treating legislative monitoring as a shared, repeatable business process rather than a heroic, ad-hoc effort performed by one or two individuals every two years. The tools and techniques detailed in this paper are available to every municipality in Texas, and they offer a clear path toward a more sustainable and effective approach to legislative engagement. This paper and the accompanying presentation are intended to make that leap easier by providing a clear roadmap, highlighting the potential pitfalls, and sharing the reusable artifacts from our journey. We invite our fellow city attorneys to adapt these ideas, remix the scripts, and contribute their own improvements, fostering a collaborative environment where all Texas cities can devote less time to chasing deadlines and more time to the substantive policy conversations that shape our communities.

Appendix A: Tooling Checklist

This checklist provides a quick reference for the key tools used in each phase of the legislative tracking project. Most of these tools are free or are part of standard office suites like Google Workspace, making the system highly accessible.

Phase	Step	Tool	Notes
A	Central Spreadsheet	Google Sheets	Create tabs for `Bills`, `Summaries`, `Calendars`, `Priorities`, and `Letters`. Enable version history to track all changes.
B	Light Automation	Google Colab	Use for running initial Python scripts in a browser-based environment. No local installation is required.
B	Data Ingestion	Python Scripts	Develop individual scripts for tasks like parsing TML newsletters (`scrape_tml_newsletter.py`) or processing LegiScan data.
C	Scheduling	GitHub Actions	Configure a `.github/workflows/` file to run scripts on a schedule (e.g., hourly, daily, weekly).
C	Alerts & Notifications	Slack / Email	Use an incoming webhook URL for Slack or a simple email library in Python to send automated digests.
D	Database (Optional)	Grist	Use the self-hosted or cloud version of Grist as a more robust, relational backend.

Appendix B: Configuration Quick Reference

Proper configuration, especially the secure management of secrets, is critical for a reliable automated workflow. Below are the key configuration elements for both local execution and scheduled runs on GitHub Actions.

1. Local Environment (`.env` file)

When running scripts on a local machine, create a file named **`.env`** in the project's root directory to store sensitive credentials. This file should be added to your **`.gitignore`** to prevent it from ever being committed to source control.

```
# .env file for local development

# Your API key from the LegiScan service

LEGISCAN_API_KEY="YOUR_LEGISCAN_API_KEY_HERE"

# Grist configuration (if used)

GRIST_API_KEY="YOUR_GRIST_API_KEY_HERE"

GRIST_DOC_ID="YOUR_GRIST_DOCUMENT_ID_HERE"

GRIST_API_URL="http://localhost:8484" # Adjust if using a different Grist
server
```

2. GitHub Actions Secrets

For automated workflows running on GitHub, store these same credentials as encrypted secrets in your repository's settings (**`Settings`** > **`Secrets and variables`** > **`Actions`**).

```
LEGISCAN_API_KEY

GRIST_API_KEY

GRIST_DOC_ID

SLACK_WEBHOOK_URL (if using Slack for notifications)
```

Appendix C: Glossary

API (Application Programming Interface): A set of rules and tools that allows different software applications to communicate with each other. APIs are used to access data or services programmatically.

API Key: A unique identifier used to authenticate a user or application when accessing an API. It helps control access and track usage.

Buffering (GIS): Creating a zone around a geographic feature (e.g., 1km around a waterway) to analyze proximity impacts.

Cloud-Hosted: A deployment model where software or services are run on infrastructure provided by a cloud service provider (e.g., Oracle Cloud, AWS, Google Cloud). Offers scalability and ease of access but may raise concerns about data sovereignty and security.

Command-Line AI (e.g., Gemini CLI): AI tools that operate via a terminal interface, allowing for direct interaction with files and system commands.

Coordinate Reference System (CRS): A system that uses coordinates to define locations on Earth. Example: EPSG:3083 (Texas Albers).

cron: A time-based job scheduler in Unix-like operating systems used to automate tasks by running scripts or commands at specified intervals.

Dashboard: A visual interface that provides an overview of key data points. In this context, a Google Sheets dashboard was used to summarize bill positions and departmental responsibilities at a glance.

Data Validation (Spreadsheets): Rules applied to spreadsheet cells to ensure consistent and correct data entry.

Debug: The process of identifying and fixing errors or bugs in software code. In AI-assisted workflows, debugging often involves pasting error messages into an AI chat to receive corrected code or explanations.

.env File: A file used to store environment variables (like API keys) locally and securely, often excluded from version control.

Grist: A modern, web-based data management tool that combines the user-friendly interface of a spreadsheet with the powerful data-linking capabilities of a relational database.

GitHub: A cloud-based platform used for version control and collaborative software development. It allows users to store, manage, and share code repositories, track changes, and automate workflows using tools like GitHub Actions.

GitHub Actions: An automation platform built into GitHub that allows you to run code and orchestrate workflows on a schedule or in response to events within your code repository.

gitignore: A configuration file used in Git repositories to specify which files or directories should be ignored by version control. Commonly used to exclude sensitive files like .env or large datasets from being tracked.

Google Colab: A free, cloud-based service from Google that provides a hosted environment for writing and running Python code in a "notebook" format, accessible through any web browser.

Hallucination: When an AI generates plausible-sounding but factually incorrect or fabricated information.

HTML (HyperText Markup Language): The standard language used to create and structure content on the web. HTML defines elements such as headings, paragraphs, links, and tables, and is the foundation of most websites. In automation workflows, HTML is often parsed by scripts to extract structured data from web pages (a process known as web scraping).

Infrastructure-as-Code: A practice where infrastructure setup (like databases or servers) is automated and managed through code.

LegiScan: A third-party service that provides comprehensive, machine-readable legislative data, including bill texts, histories, sponsors, and calendars, accessible via an API.

Linux: An open-source operating system commonly used for servers and development environments. It powers many cloud services and is favored for its stability and flexibility.

LLM (Large Language Model): A type of artificial intelligence model trained on vast amounts of text data to understand and generate human-like language. LLMs are capable of answering questions, summarizing documents, writing code, and more. Examples include OpenAI's GPT-5 and Google's Gemini.

LLM Model: A redundant but commonly used phrase referring to a Large Language Model. While technically "LLM" already includes "model," this phrasing is often used in practice to emphasize the AI system itself (e.g., “the LLM model used for drafting”).

Oracle Cloud: A cloud computing platform offering infrastructure and services, including virtual machines, databases, and storage. The "Always Free" tier provides no-cost access to basic resources suitable for small projects.

Python: A high-level, general-purpose programming language known for its readability and simplicity. Widely used in legal tech and automation projects, Python enables tasks such as web scraping, data analysis, and document generation. Its extensive library ecosystem and compatibility with tools like Google Colab and GitHub make it especially accessible for non-developers.

Python Script: A file containing code written in the Python programming language, used to automate tasks or analyze data.

Repository (Repo): A digital project folder within GitHub that contains all the files, code, and version history for a specific project. Repositories can be public or private and are used to organize and manage development work.

Scrape (Web Scraping): The process of extracting data from websites using automated scripts. Often used to collect structured information from HTML pages.

Secrets (GitHub): Encrypted variables stored in GitHub repositories to securely manage credentials and sensitive data.

Self-Hosted: A deployment model where software or services are run on infrastructure owned or managed by the user or organization, rather than relying on third-party cloud providers. Offers greater control and privacy but requires technical expertise.

Shapefile: A geospatial vector data format used in GIS software to represent geographic features like boundaries and waterways.

Slack: A cloud-based messaging platform used for team communication. In this project, Slack was used to deliver automated legislative alerts via webhook integration.

SLM (Small Language Model): A scaled-down version of a large language model (LLM), designed to run efficiently on local or resource-constrained environments. SLMs are useful for tasks that require lower latency, privacy, or offline operation.

Texas Legislature Online (TLO): The official website for tracking Texas legislative activity, including bill texts and hearing schedules.

TF-IDF (Term Frequency–Inverse Document Frequency): A numerical statistic used in text analysis to reflect how important a word is to a document in a collection or corpus. It is often used to measure the similarity between documents.

TML Legislative Update: The weekly newsletter published by the Texas Municipal League during a legislative session, which provides plain-language summaries of bills relevant to Texas cities.

Token (Authentication Token): A secure, temporary credential used to verify identity and authorize access to systems or APIs. Tokens are often used in place of passwords for automated workflows.

YAML: A human-readable data serialization format often used for configuration files, including GitHub Actions workflows.

YOLO Mode (AI Execution): A mode in command-line AI tools that allows autonomous execution of commands without user confirmation. Stands for "You Only Live Once."

Appendix D: Getting Started with Key Technologies

This appendix provides a high-level, practical guide for setting up the core technologies required for this project. The goal is to demystify the initial steps so you can begin experimenting with automation.

1. Setting Up Your GitHub Repository

GitHub is the central hub for storing your Python scripts and, more importantly, for running the scheduled automations using GitHub Actions.

- **Create an Account:** Go to GitHub.com and sign up for a free account.
- **Create a Repository:** A repository (or "repo") is simply a project folder. Click "New" to create one.
- **Choose Private vs. Public (Critically Important):** You will be given the choice to make your repository public or private. **You should always choose Private** unless you are certain that the repository will contain no confidential data**.** A private repository is only visible to you and collaborators you explicitly invite. This is essential for protecting your scripts, internal notes, and any sensitive data you might be working with. Public repositories are visible to the entire world, and you should never store API keys, credentials, or any non-public city information in them.

2. How to Work with AI for Coding (Gemini, ChatGPT, etc.)

The "AI Collaboration Model" described in this paper is a dialogue. You do not need to know how to code; you need to know how to describe a problem clearly.

- **Start with a Clear Goal:** Instead of saying "I need a script," be specific: "I need a Python script that can open the TML newsletter webpage and find every bill number mentioned."
- **Provide Context and Iterate:** For your first draft, the AI will make some assumptions. If the script doesn't work, copy the error message and paste it back into the chat with a comment like, "I ran the script you gave me, and I got this error. What does it mean?" The AI will then help you debug it.
- **Use the Right Tool for the Job:**
    - **Web-based AI (ChatGPT, Gemini):** Excellent for brainstorming, generating initial scripts from scratch, and debugging error messages.

- **Command-Line AI (Gemini CLI):** A more advanced tool that can not only write code but can also read your files, execute commands on your system, and help manage the entire project interactively.

3. Enabling the Google Sheets API (A One-Time Setup)

To allow your Python scripts to read from and write to your Google Sheet, you must enable its API. This process can seem intimidating, but it is a straightforward, one-time setup.

1. **Go to the Google Cloud Console:** Log in with your Google account.
2. **Create a New Project:** Give it a memorable name like "Legislative Tracker."
3. **Enable APIs:** In the search bar, find and enable two APIs: the **Google Drive API** and the **Google Sheets API**.
4. **Create a Service Account:** This creates a special "robot" user for your script. Go to "Credentials," click "Create Credentials," and select "Service Account." Give it a name, grant it "Editor" permissions, and click done.
5. **Generate a JSON Key:** Once the service account is created, go to its "Keys" tab, click "Add Key," and create a new JSON key. A file will automatically download to your computer. **Treat this file like a password.**
6. **Share Your Google Sheet:** This is the most important and most often missed step. Open the JSON file you downloaded and find the `client_email` address (it will look like `leg-tracker@...iam.gserviceaccount.com`). Go to your master Google Sheet, click the "Share" button, and paste in that email address, giving it "Editor" access, just as you would for a human colleague. Your script can now securely access and modify your sheet.

Appendix E: Advanced Option: Using a Free Cloud Server

While GitHub Actions is an excellent scheduler, you may eventually want more control or need a persistent, "always-on" computer to host an application like Grist. This is where a free-tier cloud server can be a powerful, no-cost option.

1. Why Use a Cloud Server?

- **Full Control Over Scheduling:** Instead of being limited by GitHub's scheduler, you can use a standard Linux utility called `cron` to run your scripts at any interval you choose (e.g., every five minutes).

- **Self-Hosting Applications:** It provides a permanent home for web applications like Grist, so your database is always accessible from a browser, just like any other website.

2. Getting Started with Oracle Cloud Free Tier

Oracle offers a generous "Always Free" tier that includes small virtual machines (VMs), which are essentially your own personal computers running in the cloud.

1. **Sign Up:** Go to the Oracle Cloud website and register for a free account. You will need a credit card for identity verification, but you will not be charged for using the "Always Free" resources.

2. **Create a VM Instance:** From the dashboard, navigate to "Compute" and "Instances." Create a new instance, making sure to select an "Always Free-eligible" shape (machine size) and operating system (Ubuntu is a great choice for beginners).

3. **Connect via SSH:** To control your cloud server, you will connect to its command line using a protocol called SSH (Secure Shell). The setup process will guide you through creating an SSH key pair, which is a secure way to log in without a password.

3. Scheduling Scripts with `cron`

`cron` is a built-in task scheduler on Linux. You can edit its configuration file, called a `crontab`, to tell your server to run specific commands at specific times.

- **How it works:** You can get your scripts onto the server by cloning your private GitHub repository. Then, you can add a line to your `crontab` to execute one of those scripts.

- **Example `crontab` entry:**

```
# Edit the crontab by running the command: crontab -e

# This line runs the TLO hearing scraper every hour, at the top of the hour.

0    *    *    *    *    /usr/bin/python3    /home/ubuntu/your-repo-name/Texas_Legislative_Tracker/update_committee_schedule.py
```

This advanced option provides the ultimate flexibility and is the natural next step for a project that needs to grow beyond scheduled reports into a persistent, interactive application.

Appendix F: Advanced Option: Using a Free Cloud Server

While GitHub Actions is an excellent scheduler, you may eventually want more control or need a persistent, "always-on" computer to host an application like Grist. This is where a free-tier cloud server can be a powerful, no-cost option.

1. Why Use a Cloud Server?

- **Full Control Over Scheduling:** Instead of being limited by GitHub's scheduler, you can use a standard Linux utility called `cron` to run your scripts at any interval you choose (e.g., every five minutes).
- **Self-Hosting Applications:** It provides a permanent home for web applications like Grist, so your database is always accessible from a browser, just like any other website.

2. Getting Started with Oracle Cloud Free Tier

Oracle offers a generous "Always Free" tier that includes small virtual machines (VMs), which are essentially your own personal computers running in the cloud.

- **Sign Up:** Go to the Oracle Cloud website and register for a free account. You will need a credit card for identity verification, but you will not be charged for using the "Always Free" resources.
- **Create a VM Instance:** From the dashboard, navigate to "Compute" and "Instances." Create a new instance, making sure to select an "Always Free-eligible" shape (machine size) and operating system (Ubuntu is a great choice for beginners).
- **Connect via SSH:** To control your cloud server, you will connect to its command line using a protocol called SSH (Secure Shell). The setup process will guide you through creating an SSH key pair, which is a secure way to log in without a password.

3. Scheduling Scripts with `cron`

`cron` is a built-in task scheduler on Linux. You can edit its configuration file, called a `crontab`, to tell your server to run specific commands at specific times.

- **How it works:** You can get your scripts onto the server by cloning your private GitHub repository. Then, you can add a line to your `crontab` to execute one of those scripts.

- Example `crontab` entry:

```
# Edit the crontab by running the command: crontab -e

# This line runs the TLO hearing scraper every hour, at the top of the hour.
0    *    *    *    *    /usr/bin/python3    /home/ubuntu/your-repo-name/Texas_Legislative_Tracker/update_committee_schedule.py
```

This advanced option provides the ultimate flexibility and is the natural next step for a project that needs to grow beyond scheduled reports into a persistent, interactive application.

Appendix G: Responsible AI Collaboration: Limitations and Pitfalls

While AI assistants are incredibly powerful, they are fallible. Treating them as a magical black box can lead to frustration, security risks, and broken code. To collaborate effectively, it is essential to understand their limitations.

1. Inaccuracy and "Hallucinations"

AI models are designed to generate plausible-sounding text, and that includes code. They can sometimes "hallucinate" and produce code that is subtly wrong, uses outdated library functions, or is completely non-functional, all while presenting it with complete confidence.

- **Best Practice: Trust, but verify.** Always test the code the AI generates. Treat it as a brilliant but sometimes forgetful junior assistant whose work must always be checked before it goes into production.

2. Security Risks with Sensitive Data

Publicly available, web-based AI chat tools are not the place to paste sensitive information. Any data you submit in a prompt could potentially be used to train future models or be seen by human reviewers.

- **Best Practice: Never paste API keys, passwords, or confidential city data into a public AI chat.** Use placeholders like `"YOUR_API_KEY_HERE"` in your prompts. Store your actual credentials securely using a `.env` file for local work or GitHub Secrets for scheduled actions, as described in Appendix B.

3. Outdated Knowledge Base

AI models are trained on a snapshot of the internet from a specific point in time. They may not be aware of the latest version of a Python library or a recent change to a website's API. A script that worked perfectly a year ago might be obsolete today.

- **Best Practice:** If a script is not working, ask the AI about the specific library it's using. It's also a good habit to search for the official documentation of a tool or library to confirm that the AI's approach is still current.

4. Risk of Over-Reliance

It is easy to fall into the trap of simply copying and pasting code without understanding how it works. This becomes a major problem when you need to make a small change or debug an issue, as you will have no foundational knowledge to draw upon.

- **Best Practice:** Ask the AI to explain the code it wrote. A follow-up prompt like, "Can you explain this script to me, line by line?" is a powerful way to learn. This turns a simple transaction into a valuable tutoring session.

5. Context Limitations

In a long and complex conversation, an AI model may "forget" details or instructions from earlier in the chat. You might find it reverting to a previous, incorrect version of a script or losing track of a key constraint you mentioned.

- **Best Practice:** If the AI seems to be losing the thread, restate the most important context in your prompt. It is also helpful to break down a large, multi-step problem into a series of smaller, self-contained conversations.

Appendix H: Advanced CLI Techniques for Resilient Automation

As you become more comfortable working with a command-line AI like the Gemini CLI, you can adopt more advanced techniques to manage complex tasks and increase your efficiency.

1. Making Long-Running Tasks Resilient to Interruption

Complex tasks, like processing thousands of bill records, can take a long time and are inevitably interrupted by network glitches, command timeouts, or other issues. To avoid starting over from scratch, you can direct the AI to log its progress.

- **The Strategy:** Create two text files: an `instruction.md` file that lists the steps for the overall task, and a `progress.log` file to track completed steps.
- **Initial Prompt:** "You are going to help me process all the bills in my database. Your instructions are in `instruction.md`. As you complete each step, I want you to append the name of the step to `progress.log`. Before you begin, read `progress.log` to see what has already been done."
- **Recovery:** If the process is interrupted, you can simply restart the AI with the exact same prompt. It will read the log file, see which steps are already completed, and automatically resume its work from where it left off.

2. Understanding Execution Modes: Supervised vs. YOLO

The Gemini CLI has different execution modes that control how it runs commands on your system.

- **Supervised Mode** (The Default): In this mode, every time the AI wants to run a command (like executing a Python script or listing files), it will first ask for your permission. You will see a confirmation prompt and must type 'y' to approve the action. This is the safest way to work, especially when you are starting out, as it gives you a final chance to review any potentially harmful or incorrect commands.
- **YOLO Mode** (Autonomous Operation): For advanced users who trust their workflow, there is "YOLO (You Only Live Once)" mode. When you enable this mode, you give the AI permission to execute commands without asking for confirmation each time. This allows for true, unsupervised automation, where the AI can carry out a multi-step plan from start to finish without any human intervention. However, it carries a significant risk; a misunderstood instruction could lead the AI to perform an unintended action.

- **Recommendation:** Become highly proficient in Supervised Mode first. Understand the kinds of commands the AI generates and get a feel for its behavior. Only use YOLO mode for well-defined, non-destructive tasks where you have a high degree of confidence in your instructions and the AI's ability to execute them correctly.

Appendix I: The SB 1844 Geospatial Analysis

1. The Challenge: Pinpointing the Impact of a Complex Bill

A significant challenge in legislative analysis is moving beyond the text of a bill to understand its tangible, real-world impact. SB 1844, and its companion HB 2494, dealt with municipal disannexation under specific circumstances, making its potential effects highly dependent on geography and existing infrastructure. For the City of Dripping Springs and other municipalities, the key questions were:

1. Which Texas cities provide municipal water or sewer service to *some*, but not all, of their territory?
2. How does a floor amendment, which limits the bill's scope to only those areas *adjacent to a navigable waterway*, change the list of affected cities?

Answering these questions required a detailed geospatial analysis. This appendix outlines the technical workflow, which relied heavily on AI-assisted Python scripting, to provide a data-driven answer.

2. The Process: A Multi-Step Geospatial Workflow

The analysis was conducted using a Python script in a Google Colab notebook, which allowed for interactive development and visualization. The AI's role was crucial in generating and debugging the code, particularly for handling specialized geospatial libraries and data formats.

(a) Step 1: Data Sourcing and Preparation

The first step was to gather the necessary datasets from various government sources.

- **Municipal Boundaries:** A shapefile of all incorporated city limits in Texas (`tl_2024_48_place.shp`) was obtained from the U.S. Census Bureau.
- **Utility Service Areas:** Shapefiles representing the Certificate of Convenience and Necessity (CCN) boundaries for all water (`PUCT_CCN_WATER_TSMS.shp`) and sewer (`PUCT_CCN_SEWER_TSMS.shp`) utilities were sourced from the Public Utility Commission of Texas.
- **Utility Ownership Data:** Excel spreadsheets (`water.xlsx`, `sewer.xlsx`) were used to identify which utilities were municipally owned.
- **Navigable Waterways:** A shapefile of the Navigable Waterway Network Lines (`Navigable_Waterway_Network_Lines.shp`) was sourced from the U.S. Army Corps of Engineers.

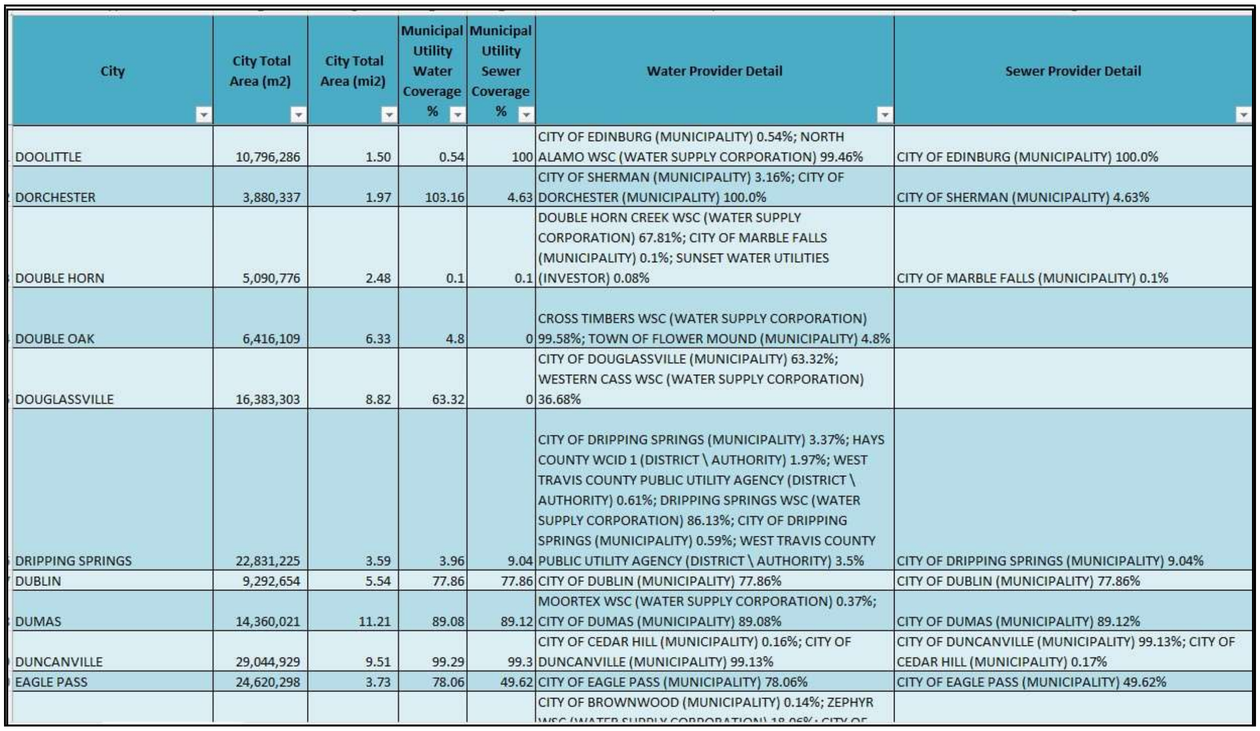
All spatial data was projected into a consistent Coordinate Reference System (CRS: `EPSG:3083`, Texas Albers) to ensure accurate area and distance calculations.

(b) Step 2: Calculating Municipal Service Coverage

The core of the initial analysis was to determine what percentage of each city's area was covered by its own municipal water and sewer services.

1. **Spatial Intersection:** The script iterated through each city in the municipal boundaries shapefile. For each city, it performed a spatial intersection with the water and sewer CCN shapefiles to find all utility providers operating within that city's limits.
2. **Coverage Calculation:** For each intersecting utility, the script calculated the precise area of overlap and expressed it as a percentage of the city's total area.
3. **Filtering for Municipal Providers:** The results were filtered to identify only the coverage provided by utilities classified as "MUNICIPAL".

This process produced a foundational list of cities that provided some, but not 100%, of their own water or sewer services, representing the initial pool of municipalities potentially affected by the bill.



City	City Total Area (m2)	City Total Area (mi2)	Municipal Utility Water Coverage %	Municipal Utility Sewer Coverage %	Water Provider Detail	Sewer Provider Detail
DOOLITTLE	10,796,286	1.50	0.54	100	CITY OF EDINBURG (MUNICIPALITY) 0.54%; NORTH ALAMO WSC (WATER SUPPLY CORPORATION) 99.46%	CITY OF EDINBURG (MUNICIPALITY) 100.0%
DORCHESTER	3,880,337	1.97	103.16	4.63	CITY OF SHERMAN (MUNICIPALITY) 3.16%; CITY OF DORCHESTER (MUNICIPALITY) 100.0%	CITY OF SHERMAN (MUNICIPALITY) 4.63%
DOUBLE HORN	5,090,776	2.48	0.1	0.1	DOUBLE HORN CREEK WSC (WATER SUPPLY CORPORATION) 67.81%; CITY OF MARBLE FALLS (MUNICIPALITY) 0.1%; SUNSET WATER UTILITIES (INVESTOR) 0.08%	CITY OF MARBLE FALLS (MUNICIPALITY) 0.1%
DOUBLE OAK	6,416,109	6.33	4.8	0	CROSS TIMBERS WSC (WATER SUPPLY CORPORATION) 99.58%; TOWN OF FLOWER MOUND (MUNICIPALITY) 4.8%	
DOUGLASSVILLE	16,383,303	8.82	63.32	0	CITY OF DOUGLASSVILLE (MUNICIPALITY) 63.32%; WESTERN CASS WSC (WATER SUPPLY CORPORATION) 36.68%	
DRIPPING SPRINGS	22,831,225	3.59	3.96	9.04	CITY OF DRIPPING SPRINGS (MUNICIPALITY) 3.37%; HAYS COUNTY WCID 1 (DISTRICT \ AUTHORITY) 1.97%; WEST TRAVIS COUNTY PUBLIC UTILITY AGENCY (DISTRICT \ AUTHORITY) 0.61%; DRIPPING SPRINGS WSC (WATER SUPPLY CORPORATION) 86.13%; CITY OF DRIPPING SPRINGS (MUNICIPALITY) 0.59%; WEST TRAVIS COUNTY PUBLIC UTILITY AGENCY (DISTRICT \ AUTHORITY) 3.5%	CITY OF DRIPPING SPRINGS (MUNICIPALITY) 9.04%
DUBLIN	9,292,654	5.54	77.86	77.86	CITY OF DUBLIN (MUNICIPALITY) 77.86%	CITY OF DUBLIN (MUNICIPALITY) 77.86%
DUMAS	14,360,021	11.21	89.08	89.12	MOORTEX WSC (WATER SUPPLY CORPORATION) 0.37%; CITY OF DUMAS (MUNICIPALITY) 89.08%	CITY OF DUMAS (MUNICIPALITY) 89.12%
DUNCANVILLE	29,044,929	9.51	99.29	99.3	CITY OF CEDAR HILL (MUNICIPALITY) 0.16%; CITY OF DUNCANVILLE (MUNICIPALITY) 99.13%	CITY OF DUNCANVILLE (MUNICIPALITY) 99.13%; CITY OF CEDAR HILL (MUNICIPALITY) 0.17%
EAGLE PASS	24,620,298	3.73	78.06	49.62	CITY OF EAGLE PASS (MUNICIPALITY) 78.06%	CITY OF EAGLE PASS (MUNICIPALITY) 49.62%
					CITY OF BROWNWOOD (MUNICIPALITY) 0.14%; ZEPHYR	

Figure 8 - An excerpt from a very long list of municipalities with water and sewer CCN maps indicating that portions of their city would be subject to disannexation under an introduced bill.

(c) Step 3: Incorporating the Navigable Waterway Amendment

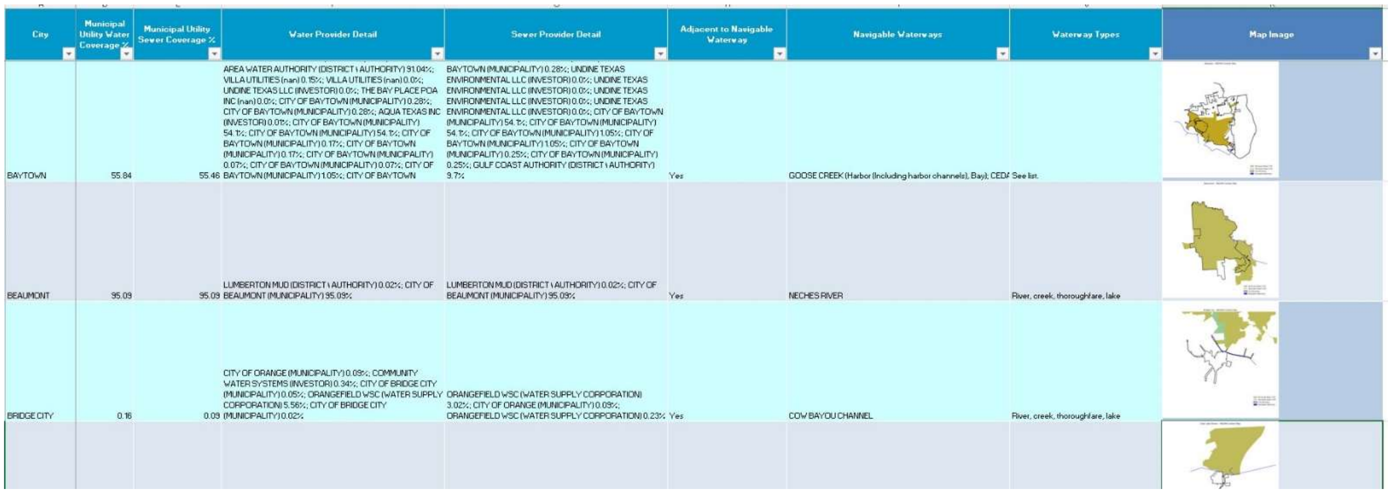
The floor amendment added a critical geographical constraint. To account for this, the analysis was refined:

1. **Buffering Waterways:** To define "adjacency," the script created a 1-kilometer buffer around the lines in the navigable waterways shapefile. This created a polygon area representing all land within 1km of a waterway.
2. **Identifying Adjacent Cities:** The script then performed another spatial intersection, this time identifying all cities whose boundaries overlapped with the buffered waterway area.
3. **Final Filtering:** The foundational list of cities with partial municipal service was then filtered down, keeping only those that were also adjacent to a navigable waterway.

Throughout this process, the AI provided essential assistance in debugging errors, such as handling `NoneType` objects that appeared due to null geometries in the source shapefiles.

3. The Outcome: A Precise, Actionable Map

The final output of this script was an Excel spreadsheet (`Cities_Affected_by_HB2494_Final.xlsx`) containing a list of the specific Texas cities that met the precise, multi-part criteria of the SB 1844 floor amendment. The spreadsheet also included details on the specific waterways each city was adjacent to and the type of waterway (e.g., "Inland Waterway," "Coastal Waterway").



City	Municipal Utility Water Coverage %	Municipal Utility Sewer Coverage %	Water Provider Detail	Sewer Provider Detail	Adjacent to Navigable Waterway	Navigable Waterways	Waterway Types	Map Image
BAYTOWN	55.84	55.46	AREA WATER AUTHORITY (DISTRICT \ AUTHORITY) 91.04%; VILLA UTILITIES (nan) 0.15%; VILLA UTILITIES (nan) 0.0%; UNDINE TEXAS LLC (INVESTOR) 0.0%; THE BAY PLACE POA INC (nan) 0.0%; CITY OF BAYTOWN (MUNICIPALITY) 0.28%; CITY OF BAYTOWN (MUNICIPALITY) 0.28%; AQUA TEXAS INC (INVESTOR) 0.01%; CITY OF BAYTOWN (MUNICIPALITY) 54.1%; CITY OF BAYTOWN (MUNICIPALITY) 54.1%; CITY OF BAYTOWN (MUNICIPALITY) 0.17%; CITY OF BAYTOWN (MUNICIPALITY) 0.17%; CITY OF BAYTOWN (MUNICIPALITY) 0.07%; CITY OF BAYTOWN (MUNICIPALITY) 0.07%; CITY OF BAYTOWN (MUNICIPALITY) 1.05%; CITY OF BAYTOWN	BAYTOWN (MUNICIPALITY) 0.28%; UNDINE TEXAS ENVIRONMENTAL LLC (INVESTOR) 0.0%; UNDINE TEXAS ENVIRONMENTAL LLC (INVESTOR) 0.0%; UNDINE TEXAS ENVIRONMENTAL LLC (INVESTOR) 0.0%; UNDINE TEXAS ENVIRONMENTAL LLC (INVESTOR) 0.0%; CITY OF BAYTOWN (MUNICIPALITY) 54.1%; CITY OF BAYTOWN (MUNICIPALITY) 54.1%; CITY OF BAYTOWN (MUNICIPALITY) 1.05%; CITY OF BAYTOWN (MUNICIPALITY) 1.05%; CITY OF BAYTOWN (MUNICIPALITY) 0.25%; CITY OF BAYTOWN (MUNICIPALITY) 0.25%; GULF COAST AUTHORITY (DISTRICT \ AUTHORITY) 9.7%	Yes	GOOSE CREEK (Harbor (Including harbor channels), Bay); CEDA	See list.	
BEAUMONT	95.09	95.09	LUMBERTON MUD (DISTRICT \ AUTHORITY) 0.02%; CITY OF BEAUMONT (MUNICIPALITY) 95.09%	LUMBERTON MUD (DISTRICT \ AUTHORITY) 0.02%; CITY OF BEAUMONT (MUNICIPALITY) 95.09%	Yes	NECHES RIVER	River, creek, thoroughfare, lake	
BRIDGE CITY	0.16	0.09	CITY OF ORANGE (MUNICIPALITY) 0.09%; COMMUNITY WATER SYSTEMS (INVESTOR) 0.34%; CITY OF BRIDGE CITY (MUNICIPALITY) 0.05%; ORANGEFIELD WSC (WATER SUPPLY CORPORATION) 5.56%; CITY OF BRIDGE CITY (MUNICIPALITY) 0.02%	ORANGEFIELD WSC (WATER SUPPLY CORPORATION) 3.02%; CITY OF ORANGE (MUNICIPALITY) 0.09%; ORANGEFIELD WSC (WATER SUPPLY CORPORATION) 0.23%	Yes	COW BAYOU CHANNEL	River, creek, thoroughfare, lake	

Figure 9 - The addition of maps illustrating the water and sewer map boundaries relative to each affected community's city limits helped validate the data.

This exercise transformed a complex legal question into a concrete, data-driven answer. It demonstrates the power of combining AI-assisted programming with publicly available geospatial data to provide city attorneys with precise, actionable insights into the real-world impact of legislation.

Appendix JTable of Figures

Part II: The Legal Ethics of AI

If AI wrote this Paper

AI Overview

Attorneys using AI must uphold ethical duties, ensuring the AI is **accurate**, **fair** (free from bias), and **transparent**. Key considerations include protecting **client confidentiality** by understanding data security, securing **informed client consent** for AI use, ensuring the attorney's **competence** and proper supervision of AI tools, maintaining **accountability** for AI-generated outcomes, and managing potential **unauthorized practice of law** by AI.

Accuracy and Truthfulness

- **Verify AI Output**: Attorneys are responsible for the accuracy of AI-generated content, such as documents or legal advice, and must ensure it is based on sound legal reasoning.
- **Beware of Errors**: AI can produce incorrect information or "hallucinations," so vigilant human review and verification are essential.

Bias and Fairness

- **Address Biased Data**: AI systems can perpetuate bias from historical data; attorneys must recognize and address this potential for discriminatory or unfair outcomes.
- **Ensure Equitable Treatment**: AI tools should treat all groups equitably, and lawyers must audit systems to detect and correct discriminatory patterns, according to the National Jurist.

Client Confidentiality

- **Secure Data**: Lawyers must ensure AI systems have robust security, proper access controls, and encryption to protect client data from breaches and unauthorized access.
- **Understand Data Flow**: Understand how AI vendors use and store client data to avoid inadvertently breaching confidentiality or attorney-client privilege.

The Life of a City Attorney

While the AI-created summary touches on the many issues related to the potential ethical pitfalls, a look at the Disciplinary Rules of Professional Conduct is also essential. Many states, including Texas, have created rules and guidance for attorneys on the use. At the end of this paper, please find a list of documents that will provide additional guidance. The primary concerns for attorneys, especially for city attorneys, is maintaining confidentiality and using AI in a way that does not mislead our clients or the courts.

Wi$h Li$t – Protection of Confidentiality while Using AI

Rule 1.05 of the Texas Disciplinary Rules of Professional Conduct emphasizes one of the most important duties of an attorney – confidentiality. TEX. DISC. R. OF PROF. CONDUCT RULE 1.05. The confidentiality of client information includes both privileged and unprivileged information. Attorneys must protect information, public or private, of the City, meaning that even if information could or must be released under the Public Information Act, the information should still be protected from inadvertent release. The confidentiality issue arises with the use of Artificial Intelligence (AI) because information, privileged or unprivileged – confidential or public, is often required to be given or uploaded to AI platforms that have access to the internet for the program to provide the information that the attorney desires. The key to successful use of AI, or any software or program which an attorney uses, is to ensure that the program or software does not share the information with any outside entity or store the information for future use without the direction of the attorney. Texas Rules of Evidence as they relate to attorney-client privilege should also be consulted.

To protect your client's information, the first step is to adopt a policy related to: (1) which programs can be used, including AI programs; and (2) how those programs and what information can be provided to the AI programs. Policies within a legal department or law firm, and policies for the cities themselves, are needed. To determine which programs will secure our client's information, the attorney must work with their IT professionals on what security measures exist within the program as well any contractual provisions can help assure confidentiality. Even if confidentiality is reasonably assured, some information may still need to be kept confidential from an AI program. This is a determination that can best be made by the attorney in consultation with the attorney's client. For example, under Rule 1.05 (b)(1)(i), a city could ask that specific information not be sent through an AI program to avoid disclosure.

Rule 1.05(c) allows attorneys to reveal confidential information in some situations. First, is when we are specifically authorized to do so, either through express authorization or after consultation. Direction from a clear AI policy created in consultation with the client could fit this exception.

Ruin the Friendship – Issues Related to Providing AI created information to the Court

A similar issue, that has appeared in the news frequently, is where an attorney has not adequately reviewed and edited information provided by a software program. AI can provide a first draft but should never provide a final draft.

Rule 3.03 requires that an attorney is truthful with a court. TEX. DISC. R. OF PROF. CONDUCT RULE 3.03. This includes not misleading the court with incorrect information provided by AI including incorrect facts or incorrect or made-up citations, aptly deemed "hallucinations". The attorney also needs to ensure that any software program is including cases or law that are adverse to their argument, something that would require specific direction to the software or AI program.

The key to this issue is quite simple. Treat AI like a tool or a non-attorney employee who is there to assist the attorney, but not to complete the work for the attorney. Check the facts. Check the citations. Even if the cases are correctly cited by a software program, interpreting case law requires skills that are uniquely suited to attorneys and not software programs. It also requires the knowledge of other case law and legal precedents that may also inform on the interpretation. All of this knowledge should be used to read and interpret a case or a software program that has summarized a case. While it is useful to have an AI program summarize a case or create a draft document, these have to be reviewed by a competent attorney to ensure accuracy. Courts and clients reasonably expect this analysis, and the rules require it. And finally, as required by Rule 3.03, if misinformation or misleading analysis is provided to a court, fix it as soon as possible.

The Life of a Showgirl – Being Transparent on use of AI to the Public, the Client, and the Court

Both the Rules and recent legislation effective on January 1, 2026 require an attorney specifically, and cities in general, to be transparent on the use of AI. TEX. H.B. 149 (89th Leg.); TEX. S.B. 1964 (89th Leg.). Notices on the city's website are required. TEX. BUS. & COMM. CODE § 552.051 (eff. Jan. 1, 2026).

An attorney should also inform its client, in the case of an in-house attorney, or inform all of its clients, if an out-house attorney, what software programs related to AI, or any software program that has access to the client's information, and access to the internet are being by the attorney. This way the client can give informed consent to the use of these programs and perform their own security analyses related to their use. And in the case of cities, the city will be able to comply with recent legislation that requires that the city let the public know what all AI is being used on its behalf. *See* TEX. H.B. 149 (89th Leg.).

CANCELLED! – Being Aware of Possible Bias in AI

AI relies on what users, like ourselves, but also the public in general, enter into the systems. Any use of mass sourced information could lead to inadvertent biases when the information is processed by the computer program especially when societal biases and developer biases are concerned. Using the same critical thinking analysis that would be used for any news or information obtained from the internet should be used when information is taken from an AI program.

<u>The Fate of OphelAI</u>

Top Tips:

1. Read and analyze everything written by AI like it was written by an intelligent non-attorney.
2. Ensure the AI program you are using will protect the city's data by reviewing their terms of service and getting with your IT professional.
3. Check for biases or incorrect interpretations of cases.
4. Let your client, and potentially the public, know which AI programs you are using.

APPENDIX - Resources for Responsible Use of AI

ABA :	https://www.americanbar.org/news/abanews/aba-news-archives/2024/07/aba-issues-first-ethics-guidance-ai-tools/
Futurism Magazine:	https://futurism.com/judge-humiliating-punishment-lawyers-using-ai
Houston Law Review:	https://houstonlawreview.org/article/137782-navigating-the-power-of-artificial-intelligence-in-the-legal-field
Justia :	https://www.justia.com/trials-litigation/ai-and-attorney-ethics-rules-50-state-survey/
New York City Bar:	https://www.nycbar.org/committees/task-force-on-digital-technologies/
Reuters:	https://www.reuters.com/legal/legalindustry/short-circuit-court-ai-hallucinations-legal-filings-how-avoid-making-headlines-2025-08-04/
State Bar of Texas:	https://www.texasbarpractice.com/artificial-intelligence-toolkit/