UG_SB 4
Budget Justification Automation

Cassidy Berghoff, Shreya Goel, Michael Griswold, Youwon Song

Client: Brian Gelder
TA: Saptarshi Biswas
Instructor: Simanta Mitra

1. Requirements

Our client is asking us to improve the process to create a budget justification document. This process involves a "One Page Budget" spreadsheet and a Streamlyne pdf document. The following are the requirements for the base process:

1. Validate the numbers on the spreadsheet against the numbers in the Streamlyne pdf document.
2. Translate the numbers on the spreadsheet into a budget justification document that describes each spending in detail.

Our expected users are research professors and grant workers at Iowa State University. These users have little to no experience in coding or technical skills. Furthermore, the users have many different uses for the budget justifications. Some need to justify their spending in accordance to USDA standards, while others need to have it up to NIH standards.

With our users in mind, we created two sets of requirements for our tool: frontend requirements and backend requirements.

Frontend Requirements:

- Create a user-friendly interface that is easy to understand.
- Allow users to view, download, and delete documents.
- Communicate errors from computation to the user for solving.
- Support secure user authentication and account management.
- Support cloud document storage and multiple user access.
- Display validation results in both document and natively in-browser.

Backend Requirements:

- Accurately extract information from the spreadsheet.
- Accurately extract information from the Streamlyne pdf.
- Validate the information from the spreadsheet against the Streamlyne information.
- Provide informative exceptions for cases where something has broken that the user can fix.
- Standardize documents by providing multiple, commonly used templates to create budget documents

- Incorporate features that use AI to help reuse documents and make modifications as needed
- Accurately insert and integrate spreadsheet data into the corresponding sections of the budget justification document.
- Support multiple budget structures and year spans without requiring manual adjustment.
- Automatically reconcile data across multiple spreadsheet sheets to ensure consistency in the generated document.

2. Design

We split the overall project into 6 steps:

1) User logs into the website and uploads a spreadsheet and Streamlyne pdf
2) Spreadsheet and Streamlyne totals are extracted
3) Spreadsheet totals and Streamlyne totals are validated against one another
4) Spreadsheet totals are inserted into a word document(Template)
5) User can view and download document
6) User customizes document as needed

Our tech stack is as follows:
Frontend:

- Flask for backend integration
- HTML and Bootstrap CSS for the UI
- JavaScript for frontend logic
- Vercel for frontend hosting
- Render for backend deployment
- Supabase for cloud storage and user authentication

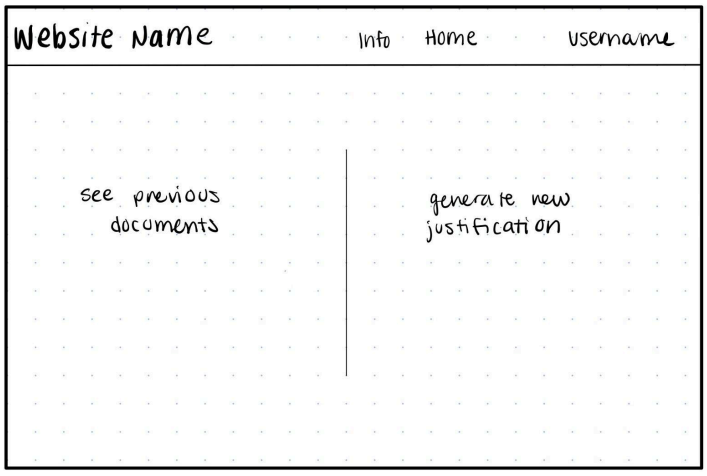
Backend:

- Pretty much all of the code for Backend is done using Python and specific python libraries
- Pandas library for spreadsheet extraction and Data Frame to store the information
- Regex and pypdf for streamlyne extraction
- Docxptl and Jinja for insertion of data into templates and working with word documents
- Python docx library for additional features to modify and customize document

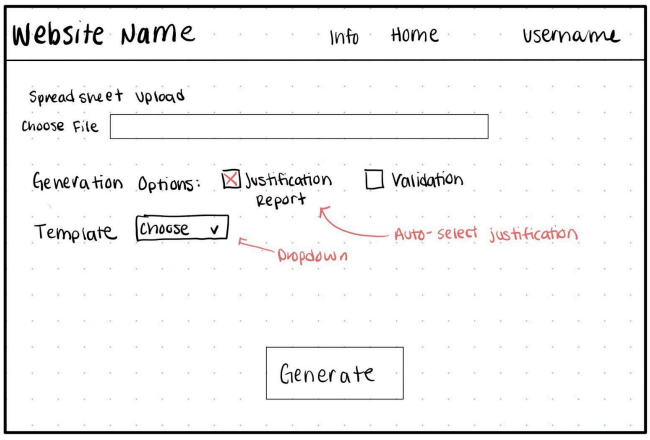
- Gemini AI (the free version) for AI features

To design the frontend experience, Cassidy used Notability on her iPad to create a set of screen sketches to illustrate a basic UI and also illustrate the flow of the application. These screen sketches were approved by the client, and then used by Michael to create the frontend UI.

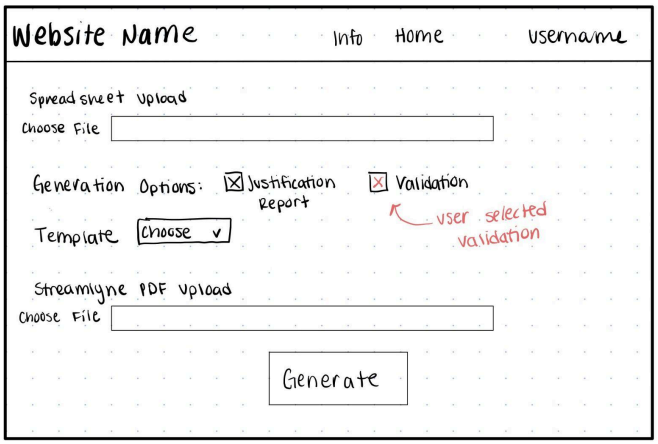
Below is the screen sketches for our UI:



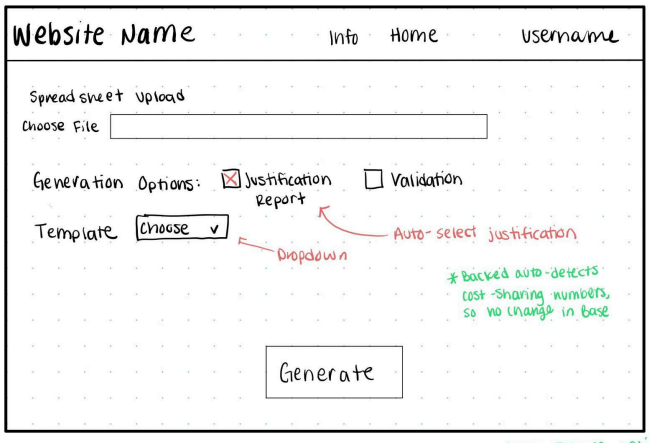
Home page - signed in



generation page - base

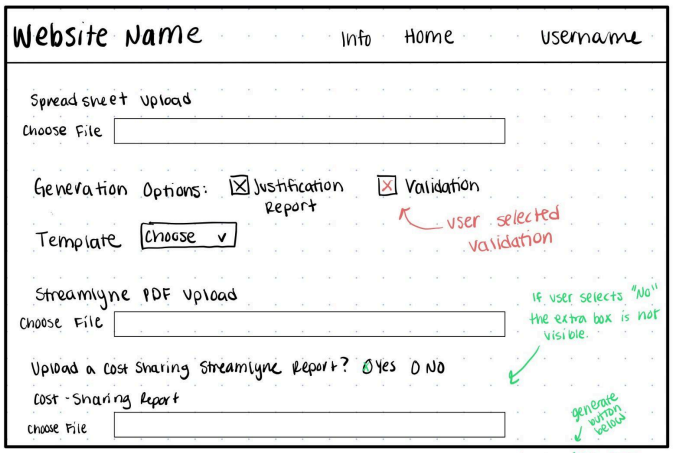


generation page - Base + validation



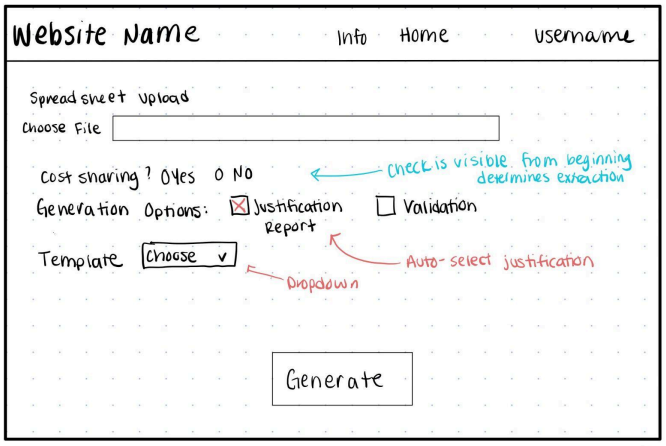
cost share option 1

generation page - Base



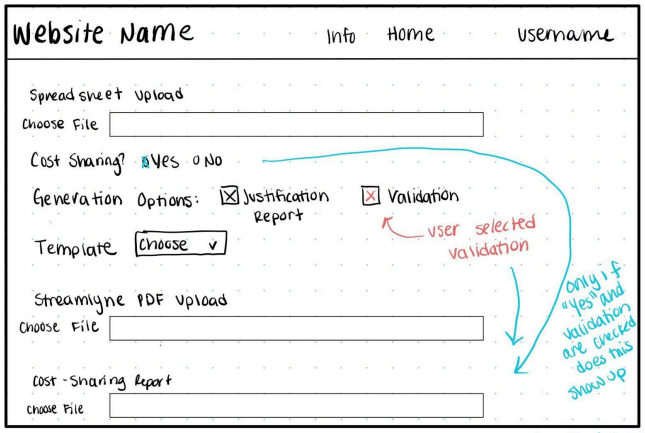
Cost Sharing Option 1

generation page - Base + validation



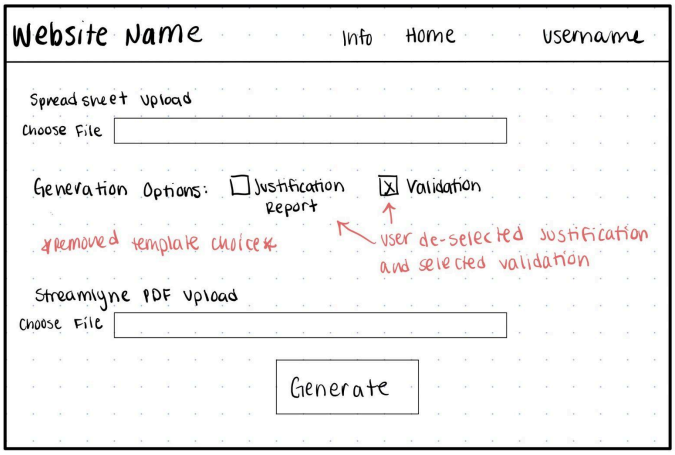
Cost sharing option 2

generation page - Base

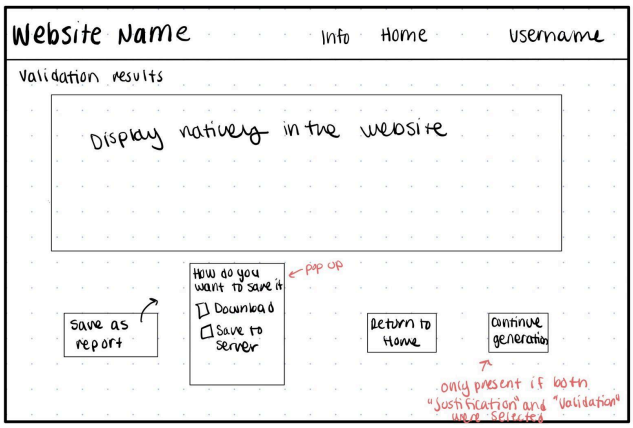


Cost sharing opt. 2

generation page - Base + validation

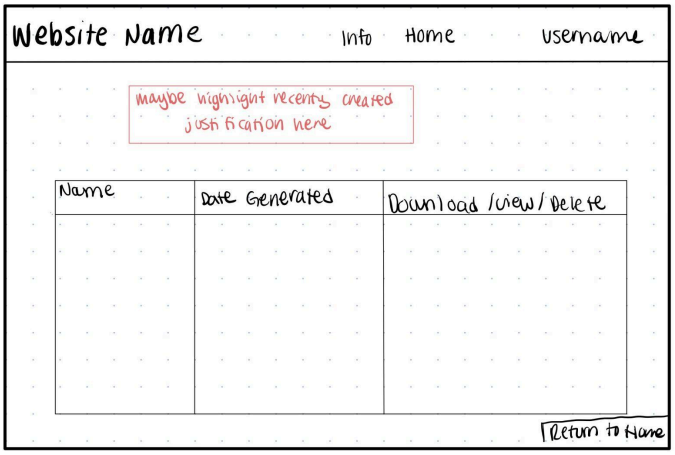


generation page - validation only



Validation step

User has selected validation and hit "Generate".



Report Display

Get to this screen either from creating a justification or by hitting "see previous" documents on home page

3. Work Done

Our team is made up of four members. The work was split up across the steps of our project. This meant that the development of our project was being developed end-to-end from the beginning, ensuring that a workable product was being created as soon as possible.

3.1 Michael Griswold - Steps 1 & 5

Michael was responsible for the frontend user interface (Step 1) and the document viewing system (Step 5). This involved designing and implementing the

web pages that users interact with to upload spreadsheets, validate results, generate documents, and view the generated documents. The frontend interface was developed using HTML, Bootstrap CSS and custom CSS, and JavaScript. Flask was used for integrating the frontend with the backend Python code.

The frontend was designed to prioritize simplicity and usability because the users aren't necessarily expected to have a technical background or experience. The different tools were separated into multiple pages, such as document generation and document viewing. JavaScript was used to update the UI based on the user's selections. For example, selecting the validation option on the generation page displayed an additional upload for Streamlyne PDFs. The UI was constantly updated and received changes throughout the development process as backend features evolved and were added.

A major part of the work was to integrate the frontend interface with the backend APIs using Flask. JavaScript fetch requests were used to send spreadsheet and PDF files to the Flask backend so they could be processed, and to retrieve the generated documents.

One challenge of the development was moving the project from a local version and deployment on the school server to being a fully deployed cloud application outside of the school server. Since the frontend, backend, and cloud storage plus authentication were all hosted/deployed on separate services, the communication between the services had to be connected carefully. The frontend was hosted on Vercel, the backend was deployed using Render, and the cloud storage and user authentication was handled through Supabase. There initially were challenges with HTTPS communication, CORS, and syncing everything across the different services.

Another challenge was refactoring the system to use cloud storage rather than local storage, and implementing user authentication. The final implementation required users to create an account, log into their accounts, and only have access to the documents that they generated. This implementation required token handling between the frontend and backend, storage paths unique to each user, and signed URLs for document access. These changes converted what would have been a single-user system to a system that could be used by multiple users at once.

User authentication and cloud storage were added using Supabase. User authentication was added so that users can securely access their own generated documents.

The document management system allows users to view, download, and delete the documents they have generated. The preview documents are PDF files, and downloads are .docx files. Documents were automatically converted from .docx

to .pdf using LibreOffice, allowing in-browser previewing of the document. Both documents were then stored in Supabase tied to the user's ID.

Validation reports can be viewed either in the PDF format or displayed directly in the website as a native HTML view. This enabled flexibility for the users to review the validation data in whatever format they prefer. Error handling was also implemented so that backend exceptions would be communicated to the user rather than being completely behind the scenes.

3.2 Cassidy Berghoff - Steps 2 & 3

Cassidy was responsible for data extraction (Step 2) and validation (Step 3). Data extraction involved identifying the important information from the spreadsheet and Streamlyne and translating them into a pandas DataFrame. Validation involved accurately matching the information from the spreadsheet to the Streamlyne document and comparing the numbers.

Data extraction was split between the spreadsheet and the Streamlyne document. For the spreadsheet, the pandas library was used to read the document. There are three sheets in the spreadsheet: "Summary", "Tuition", and "Travel". All three sheets are in a different format, and different manipulation had to be developed to extract the information. This manipulation was done using the pandas and numpy python libraries. This level of data manipulation using pandas was new to Cassidy, so she had to learn a lot of skills to make it work.

Furthermore, the data extraction step had to be reworked multiple times. Reworking prioritized removing as many assumptions as possible in the extraction. These assumptions include: the column that headers are located in, the number of years the budget spans, whether cost-share is present, and more. For assumptions that cannot be changed, error handling using python Exceptions was implemented to inform the user when an assumption has failed and how to remedy it.

Data extraction for Streamlyne used the Regular Expressions library. This was a new tool that Cassidy had to use. The pdf was read using pypdf, which turned the spreadsheet into a long string. Different RegEx strings were used to grab the relevant data from the string. This proved difficult to find standardized expressions that work across examples. However, the regular expressions used were able to accurately extract the information into Data Frames. RegEx was used over AI for base functionality to ensure accuracy of the numbers.

The second step that Cassidy was responsible for was data validation (Step 3). This step involved matching the rows from the spreadsheet DataFrame to the rows of the Streamlyne DataFrame. This proved difficult as the DataFrame's have different categories, and the entries have different names. An example of this can be

seen in Salary information: in the spreadsheet, a research assistant would be described as “Research Asst-Halftime” in the category “other_personnel”. Meanwhile, in the Streamlyne pdf, that same position would be described as “Graduate Assistant 1 TBD” and found in the “Salary/Wages - Student Hourly” category. The code had to match those two together. To do this, Cassidy learned a new skill called fuzzing. Using the python library called rapidfuzz, the names are matched according to their “best match” and not a direct map. This has worked seamlessly.

Data validation was also used inside the spreadsheet. The sheets Tuition and Travel have a yearly summary that need to be compared to the yearly totals in the Summary sheet. This has the same matching problem as the Tuition and Summary split the students into different categories, but with different names. This allowed for further checks to make sure that the data was as valid and correct as possible.

3.3 Youwon Song - Step 4

Youwon was responsible for document generation (Step 4). This step had two parts: creating the Word document templates, and building the Python code to populate them with data.

The first part involved creating the budget justification templates in Word. Specifically, Template1.docx (for USDA grants) and Template2.docx (for NIH grants) were built from scratch using sample budget justification documents from real grant submissions as references. The templates cover all major sections of a budget justification: key personnel, other personnel, fringe benefits, travel, materials and supplies, equipment, consultant services, indirect costs, and tuition. Placeholders were embedded directly into the docx files using docxtpl’s Jinja2 syntax - {{variable}} for individual values and {% for %} loops for repeating sections such as personnel entries and travel trips.

Designing the placeholders required careful planning: each variable had to be named descriptively enough to map clearly to a specific budget figure, while the loop structures had to handle a variable number of entries (e.g., one to five personnel, one to several travel trips) without breaking the document layout. This required studying the formatting conventions of each grant type and determining which sections needed dynamic data versus fixed boilerplate text, since USDA and NIH formats differ in how they present cost breakdowns and narrative justifications.

The second part involved building a context-generation function in Python (build_context()) that reads the extracted data and produces a dictionary mapping every placeholder to its value. This required translating raw pandas DataFrames from the Extractor into structured Python objects the templates could iterate over.

For the personnel section, each person's salary was computed per year, fringe benefit rates were applied by category (faculty, P&S, graduate student, and undergraduate), and descriptive text was pulled from a dedicated "Descriptions" column in the spreadsheet when available. The travel section required grouping individual trip rows into categories (conference attendance versus field visits) and reconciling raw Travel sheet dollar amounts with escalation-adjusted totals computed by the Summary sheet's formulas, since year-over-year cost escalation meant the two sources did not always agree.

For materials and supplies, the code first checked whether the Extractor's DataFrames contained description text, and fell back to reading sub-rows directly from the spreadsheet when they did not. Indirect costs were handled by reading the F&A rate from the spreadsheet's formula rows rather than hardcoding a value, so that the generated document always reflects the rate actually used in the budget.

Throughout, robustness across spreadsheets that vary in structure, differing numbers of budget years, presence or absence of a Descriptions column, different indirect cost calculation methods, required careful fallback handling to ensure the generation step would not fail on an unfamiliar spreadsheet.

3.4 Shreya Goel - Step 6

Shreya was responsible for creating the templates and incorporating additional features that use AI to customize the document. Template creation involved standardizing the various sample documents we received and separating them by the type of project (USDA, NIH, etc.) This was primarily done on Word and had placeholders for numbers that would be inserted using Python. Template choice was incorporated on the backend so the user could generate a document for the template of their choice.

There were 2 main additional features to be incorporated: reusability of documents and appending sections/text to the document. Reusing the document enabled the user to reuse certain wording from a previous document that could be reused for a new document. This involved the user uploading a spreadsheet and an old word document, as well as the section title they wanted reused. The backend code uses Python to extract the section based on the title from the provided document, the Pandas Dataframe for the updated numbers from the spreadsheet, and Gemini AI to replace the numbers in the reused section with the numbers from the provided spreadsheet which are stored in the Pandas Dataframe. Gemini AI was particularly helpful because it could parse the text and determine where the

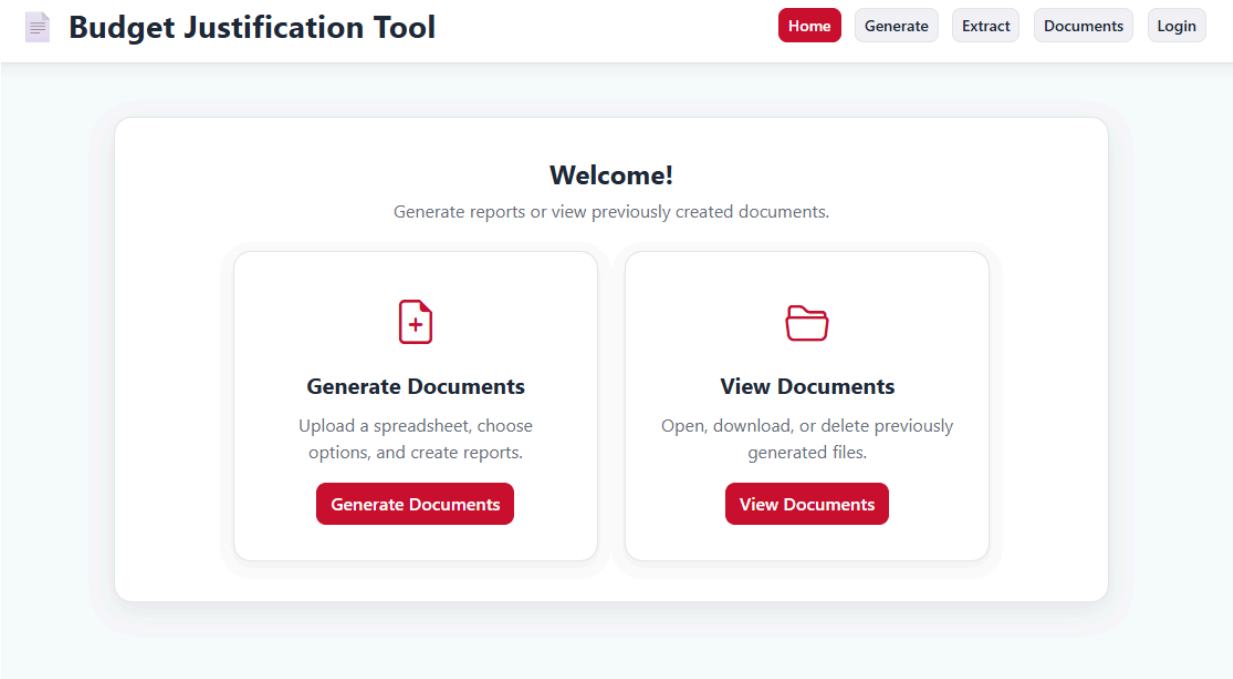
numbers were, and interpret the dataframe's labels and values to replace the numbers with the correct ones. Being able to interpret the text and understand the surrounding context was very important in this feature, which Gemini accomplished well.

The second additional feature is adding additional text to the document; this is done in two ways. Either, the user can add a section to the end of the document by providing the section title and text they want displayed. This is particularly useful in the case there was additional background information that the user wanted to add but wasn't necessarily in the spreadsheet. The other way is text can be inserted in a specific spot of a particular section in the document. The user gives the section name and line number they want text inserted, and it would be put after the provided line. This is useful in case there's a sentence or simple description they'd like to add. Python code and features from the docx library were used to achieve this. A common issue that comes up is if the user provided text has typos or grammatical errors; the user would only realize this once the changes are made and they'd have to go through the whole process again just to fix their mistakes. To prevent this from becoming a tedious process, Gemini AI was used here to proofread text submitted and fix any errors so that the final version would be error free and reduce human labor and error.

Core technologies learned throughout this process were using Gemini AI, Python, and Regex. While the additional features eventually used Gemini, it was originally done using regex, which was both challenging and tedious. Gemini was then later used, which itself was also something new to learn. There were a few challenges with testing these features once AI was incorporated because there are a limited number of calls you can make, and testing usually requires quite a few calls. However, once the testing phase was done, Gemini proved to be a good decision and quite powerful. Finally, I personally had not worked extensively with Python as I'm more of a low-level programmer. Learning multiple Python libraries and functionalities was a good experience and utilized a lot in this project.

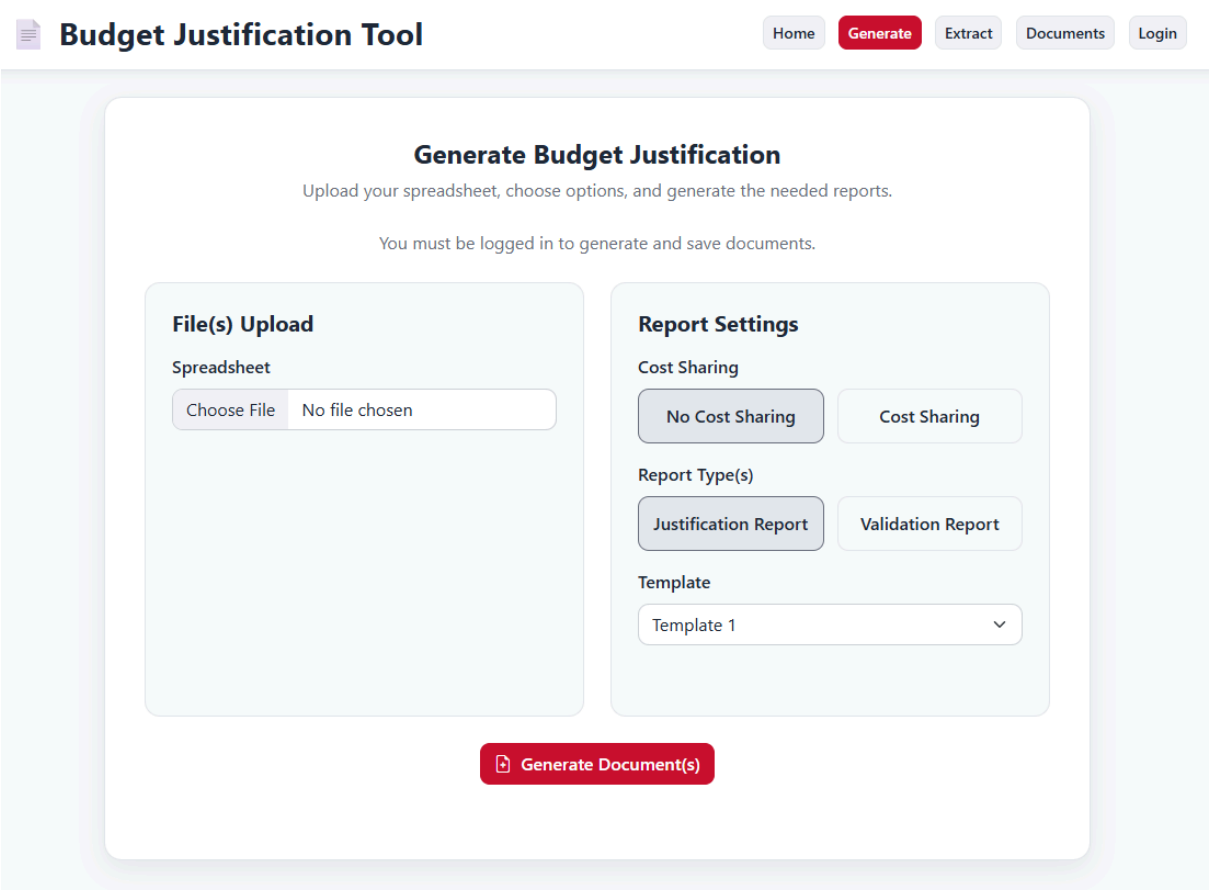
4. Results Achieved

We were able to fulfill all of our requirements, and even go above and beyond to complete some additional features. Below are screenshots that show the flow of the website.



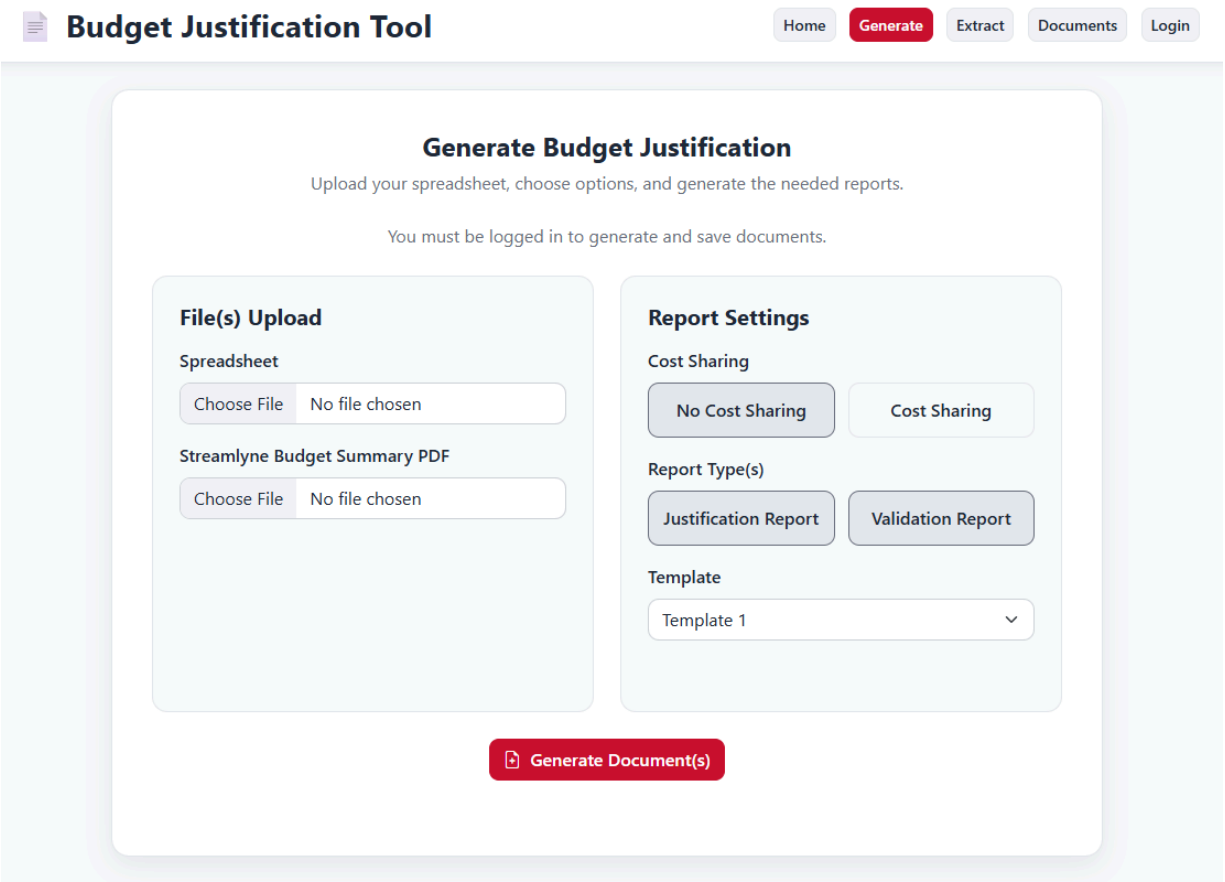
Home Page

This is the first page a user will see upon opening the website. There are short descriptions about what features they can access on the website, such as generating documents and viewing documents.

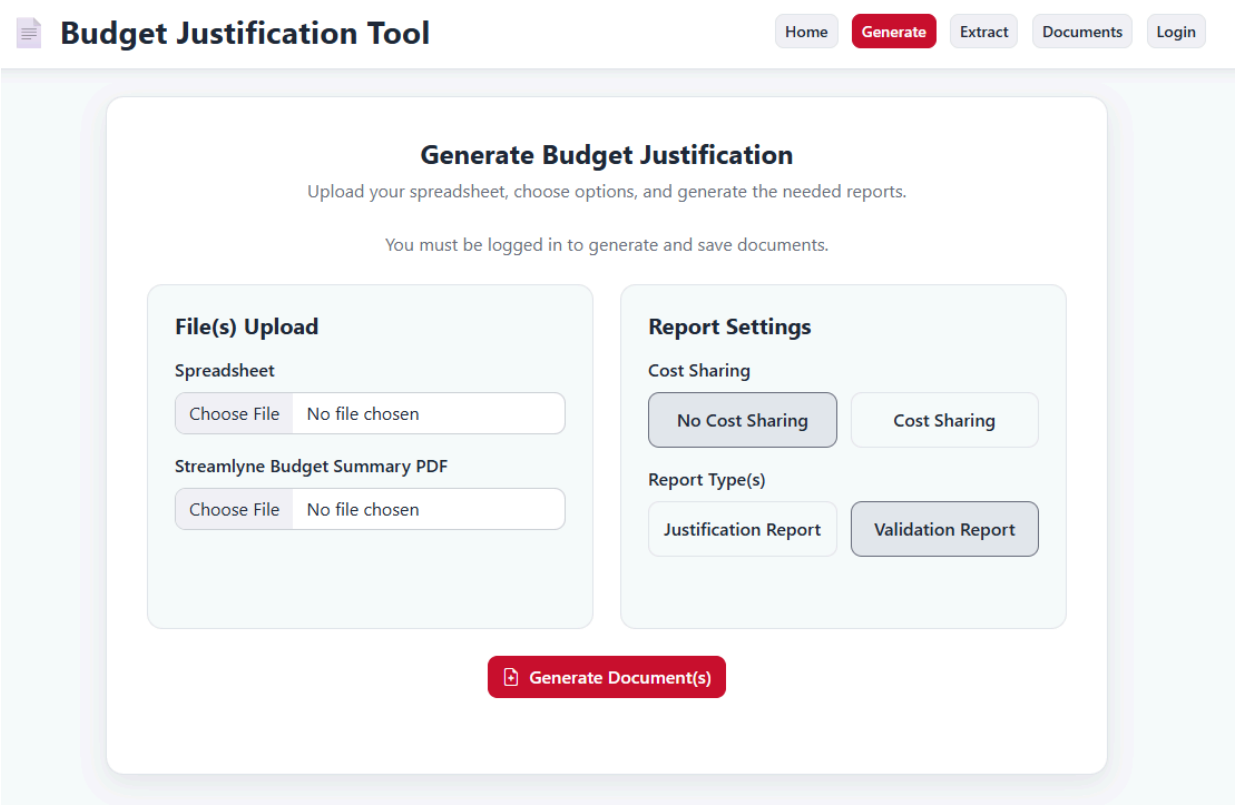


Generate Page with Default Settings (Justification Report enabled, no cost sharing)

This is the generation page, and the settings here are the default settings. The default settings have only a justification report selected. That setting only requires a spreadsheet file upload. No cost sharing is enabled, and template 1 is selected for the template dropdown. After the user has uploaded everything and selected their preferred settings, they will click the generate button at the bottom.

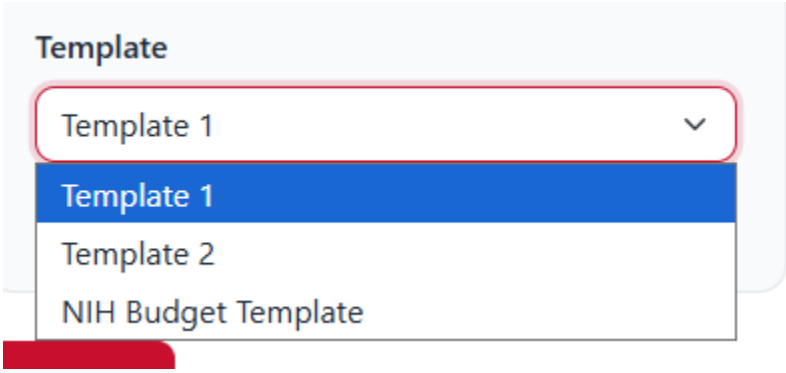


Generate Page with Justification Report & Validation Report Selected
With both justification report and validation report selected for the report types, there is both a spreadsheet and a streamlyne budget summary upload.



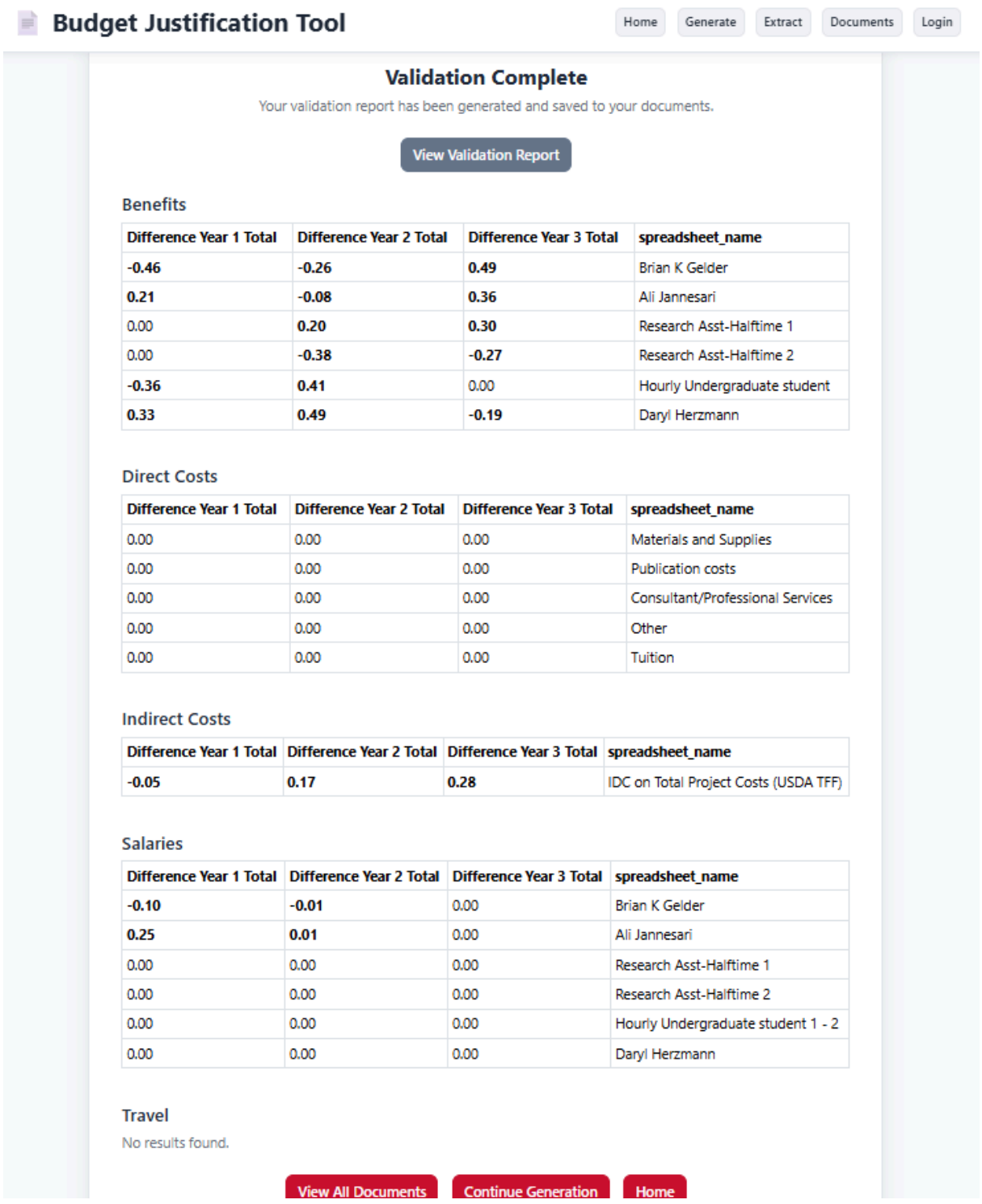
Generate Page with only Validation Report selected

With only a validation report selected, the template selection is unnecessary.



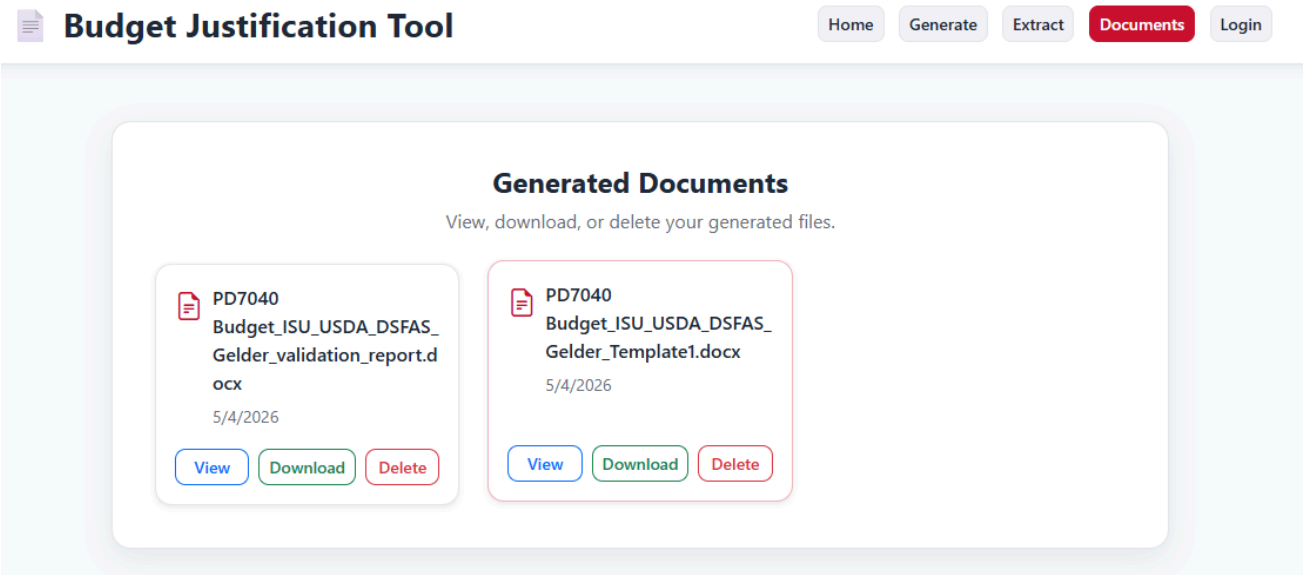
Template Choice Dropdown

These are the templates the user has an option of selecting.



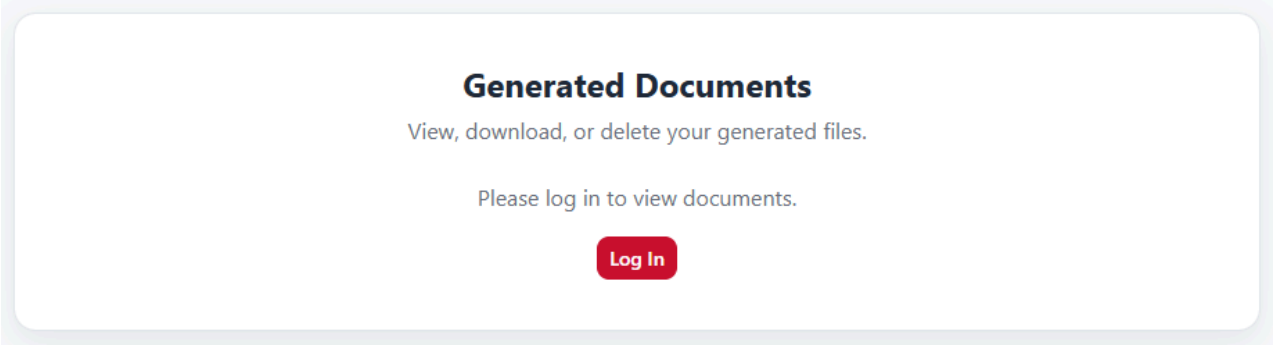
Validation Report Native In-Browser View

If the user chose to generate a validation report, the validation report will show up as an in-browser preview for easy access. They also have the option of opening it as a separate document.



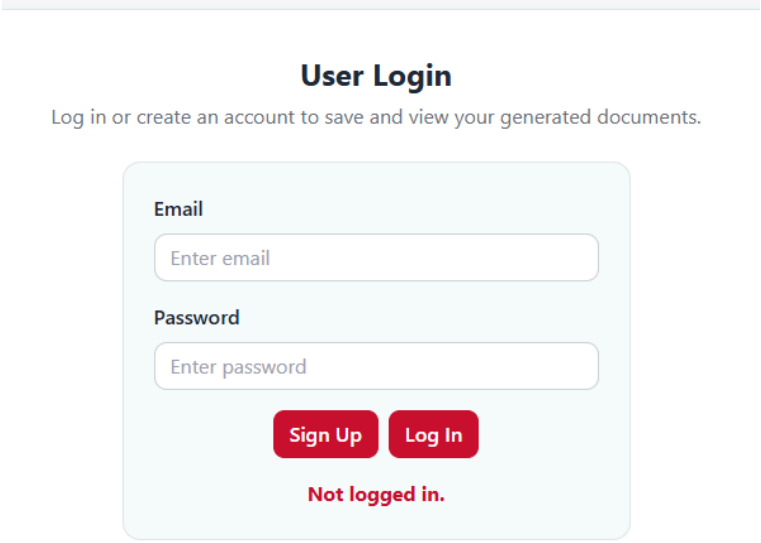
Documents Logged-In View

Assuming the user is logged in, this is an example of what they could see. They will see all of their generated documents, and have the option to either view, download, or delete the document.

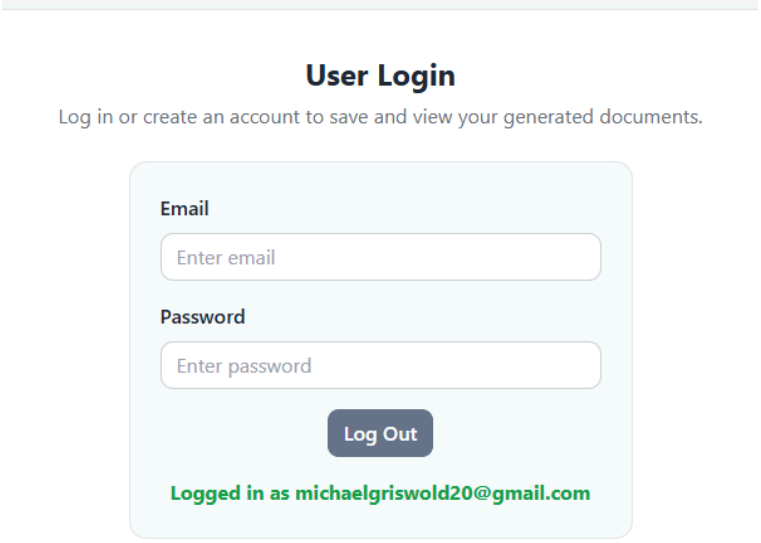


If not Logged-In View

If not logged in, the interface will prompt the user to login. You cannot view or generate documents if you are not logged in.

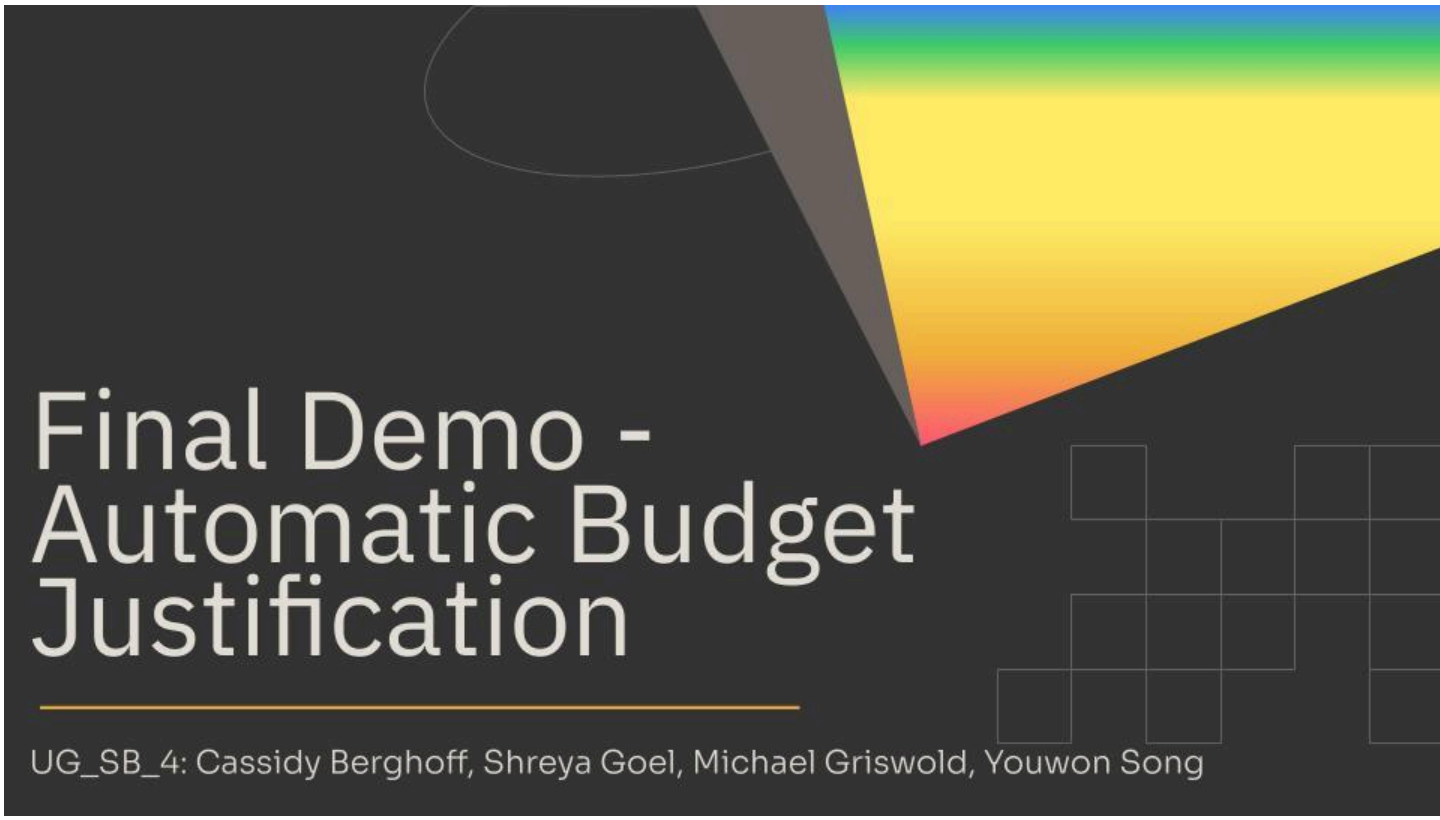


Login Page not Logged-In
This is the login page if the user has not logged in. They have the option to either sign up or log in.



Logged-in View
And this is what the login page looks like if the user is logged in.

Appendix


Final Demo - Automatic Budget Justification
UG_SB_4: Cassidy Berghoff, Shreya Goel, Michael Griswold, Youwon Song

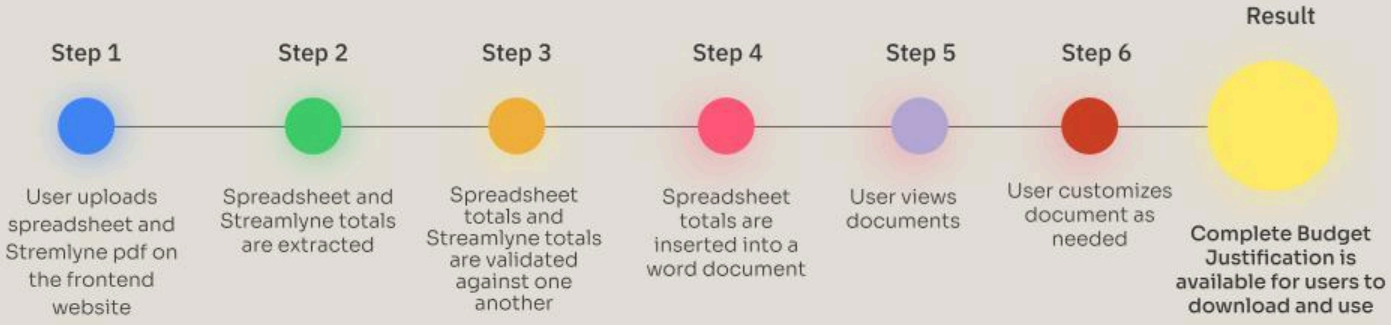

Budget Justification Overview
- Iowa State has a lot of research projects
- They must submit detailed documents that talk about the budget for their project
- Starts with spreadsheet
  - All the costs are listed
- Manual work to translate costs into word document
- Validation with Streamlyne pdf's

What is an Automatic Budget Justification?

Instantly translating raw spreadsheet data into a compliant narrative document.

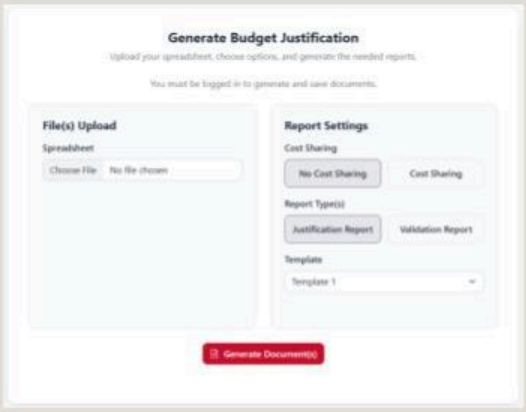
- **Zero Manual Entry**: Eliminates the traditional 30 to 60-minute copy-paste process from Excel to Word.
- **Guaranteed Consistency**: Ensures the numbers in the final narrative exactly match the source spreadsheet.
- **Automated Validation**: Instantly cross-checks figures for 100% accuracy and compliance.

How Our Tool Works



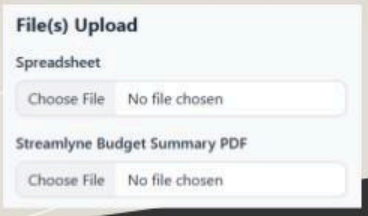
Step 1: User Interface - Uploading Document

Default

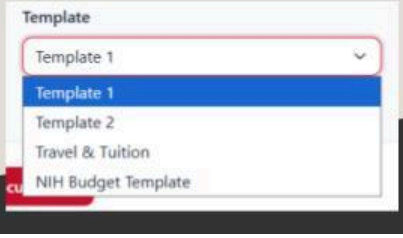


- Upload Spreadsheet
- Choose Report Options
  - Justification Report?
  - Validation Report?
    - Upload Streamlyne PDF
  - Cost Sharing?
  - Template
- Designed for Ease of Use

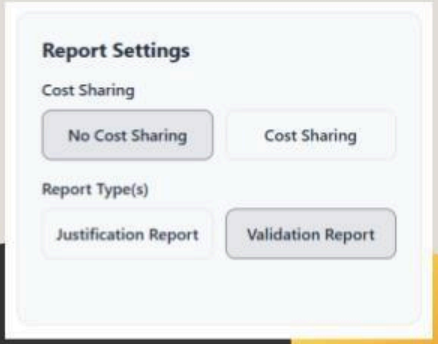
If Validation Report Selected



Template Dropdown



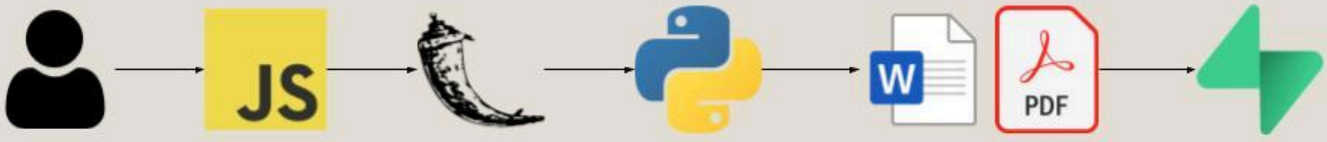
If Only Validation, no template choice



Step 1: User Interface - Uploading Document

Frontend to Backend Integration

- User actions trigger API requests to Flask
- Python Backend processes data and generates documents
- Outputs stored in Supabase for persistent access
- Seamless frontend-backend communication



User → Frontend → Flask → Backend → Output → Storage

Step 1: User Interface - Uploading Document

Deployment

- Frontend hosted on Vercel
- Backend deployed on Render
- Supabase used for cloud storage and user authentication
- Enables full-stack system to be used as a live web application

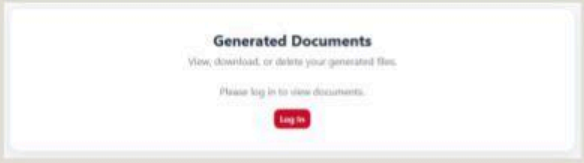
Vercel

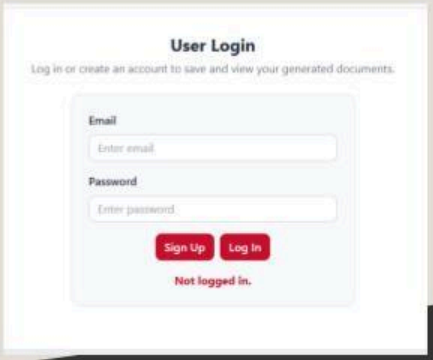
Render

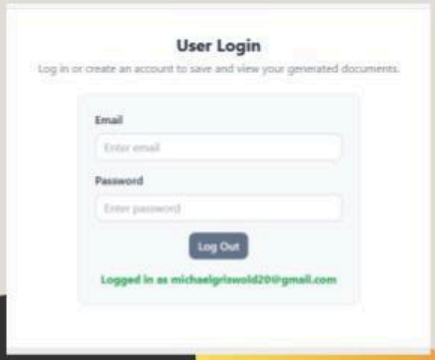
Step 1: User Interface - Uploading Document

Authentication & UI/UX

- Supabase authentication integration
- Users must be logged in to generate documents
- Prevents unauthorized access
- UI updates based on user state

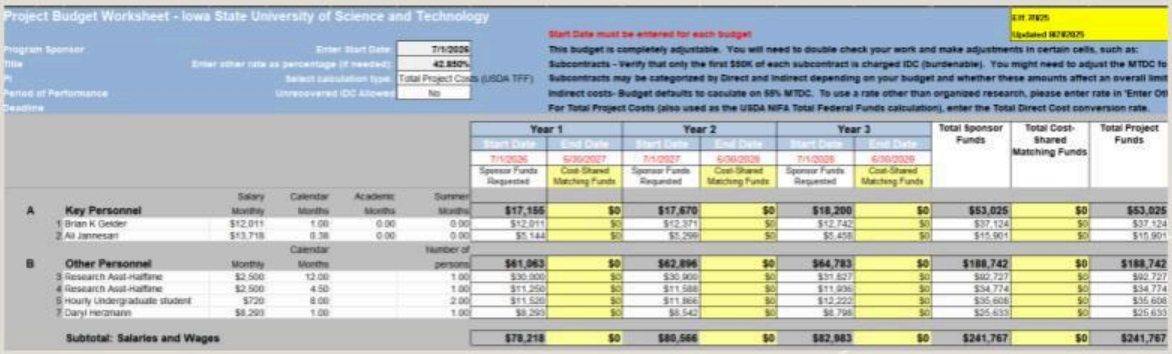






Step 2: Data Extraction

The user has uploaded a standardized
"One Page Budget Form"



Project Budget Worksheet - Iowa State University of Science and Technology

Program Sponsor
Title
PI
Period of Performance
Deadline

Enter Start Date: 7/1/2026
Enter other rate as percentage (if needed): 42.850%
Select calculation type: Total Project Costs (USDA TFF)
Unrecovered IDC Allowed: No

Start Date must be entered for each budget

This budget is completely adjustable. You will need to double check your work and make adjustments in certain cells, such as:
Subcontracts - Verify that only the first $50K of each subcontract is charged IDC (burdenable). You might need to adjust the MTDC fo
Subcontracts may be categorized by Direct and Indirect depending on your budget and whether these amounts affect an overall limit
Indirect costs- Budget defaults to caculate on 56% MTDC. To use a rate other than organized research, please enter rate in 'Enter Ot
For Total Project Costs (also used as the USDA NIFA Total Federal Funds calculation), enter the Total Direct Cost conversion rate.

						Year 1		Year 2		Year 3		Total Sponsor Funds	Total Cost-Shared Matching Funds	Total Project Funds
						Start Date: 7/1/2026	End Date: 6/30/2027	Start Date: 7/1/2027	End Date: 6/30/2028	Start Date: 7/1/2028	End Date: 6/30/2029			
		Salary	Calendar	Academic	Summer	Sponsor Funds Requested	Cost-Shared Matching Funds	Sponsor Funds Requested	Cost-Shared Matching Funds	Sponsor Funds Requested	Cost-Shared Matching Funds			
A	**Key Personnel**	Monthly	Months	Months	Months	**$17,155**	**$0**	**$17,670**	**$0**	**$18,200**	**$0**	**$53,025**	**$0**	**$53,025**
1	Brian K Gelder	$12,011	1.00	0.00	0.00	$12,011	$0	$12,371	$0	$12,742	$0	$37,124	$0	$37,124
2	Ali Jannesari	$13,718	0.38	0.00	0.00	$5,144	$0	$5,299	$0	$5,458	$0	$15,901	$0	$15,901
			Calendar		Number of									
B	**Other Personnel**	Monthly	Months		persons	**$61,063**	**$0**	**$62,896**	**$0**	**$64,783**	**$0**	**$188,742**	**$0**	**$188,742**
3	Research Asst-Halftime	$2,500	12.00		1.00	$30,000	$0	$30,900	$0	$31,827	$0	$92,727	$0	$92,727
4	Research Asst-Halftime	$2,500	4.50		1.00	$11,250	$0	$11,588	$0	$11,936	$0	$34,774	$0	$34,774
5	Hourly Undergraduate student	$720	8.00		2.00	$11,520	$0	$11,866	$0	$12,222	$0	$35,608	$0	$35,608
7	Daryl Herzmann	$8,293	1.00		1.00	$8,293	$0	$8,542	$0	$8,798	$0	$25,633	$0	$25,633
	Subtotal: Salaries and Wages					**$78,218**	**$0**	**$80,566**	**$0**	**$82,983**	**$0**	**$241,767**	**$0**	**$241,767**

Categories include: Key Personnel, Other Personnel, Fringe Benefits,
Equipment, Travel, Other Direct Costs, and Indirect Costs

Step 2: Data Extraction

We need to accurately extract information from a standardized spreadsheet...

		Salary	Calendar	Academic	Summer						
A	**Key Personnel**	Monthly	Months	Months	Months	**$17,155**	**$0**	**$17,670**	**$0**	**$18,200**	**$0**
1	Brian K Gelder	$12,011	1.00	0.00	0.00	$12,011	$0	$12,371	$0	$12,742	$0
2	Ali Jannesari	$13,718	0.38	0.00	0.00	$5,144	$0	$5,299	$0	$5,458	$0

... and transform it into a computational friendly data structure

```
key_personnel------------------------------------------------------------------
        Full Name Monthly Percentage  Year 1 Total Year 2 Total Year 3 Total
0  Brian K Gelder                 1  12010.914167        12371        12742
1    Ali Jannesari             0.375       5144.25         5299         5458
```

This is done with a python library called pandas

Step 2: Data Extraction

The challenge with spreadsheet extraction is that on top of the Summary sheet, there is also the Tuition and the Travel sheet.

	Ag & Biotech Engineering (ABE) Students Only				
	Year 1	Year 2	Year 3	Year 4	Year 5
Masters Students (enter no. of students per term)					
	Year 1	Year 2	Year 3	Year 4	Year 5
PhD Students (enter no. of students per term)	1	1	1		
	2	2	2		
	1	1	1		

Domestic Travel												
					Lodging			Meals				
Year	Purpose & Destination	# of People	Airfare/ Person	# Nights	Rate per Night	*Lodging Total*	# Meal Days	Meal Cost per Day	*Meal Total*	Ground Transportation	Registration Per Person	TOTAL
1	NeurIPS - student	1	$500	4	$180	*$720*	5	$40	*$200*	$0	$300	$1,720
1	NeurIPS - academic	1	$500	4	$180	*$720*	5	$40	*$200*	$0	$900	$2,320
1	Fieldwork	1	$0	0	$0	*$0*	2	$40	*$80*	$210	$0	$290
											$375	**$4,330**
2	AWRA GWTC - student	1	$515	4	$185	*$740*	5	$40	*$200*	$0	$200	$1,655
2	AWRA GWTC - faculty	1	$515	4	$185	*$740*	5	$40	*$200*	$0	$800	$2,255
2	Fieldwork	1	$0	0	$0	*$0*	2	$40	*$80*	$210	$0	$290
											Year 2 Total	**$4,200**
3	WRA Meeti	1	$530	4	$191	*$764*	5	$40	*$200*	$0	$200	$1,694
3		1	$530	4	$191	*$764*	5	$40	*$200*	$0	$600	$2,094
3	Fieldwork	1	$0	0	$0	*$0*	2	$40	*$80*	$210	$0	$290
											Year 3 Total	$4,078

This data also needs to be extracted from the spreadsheet. Each sheets format provided unique challenges for extraction; there was not a one-size fits all

Step 2: Data Extraction

Furthermore, Iowa State University uses Streamlyne to organize research project budgets, as it helps organize grant funding. The totals that Streamlyne has are the final, accurate numbers.

Iowa State University - Budget Page Summary

Proposal Number: 7040 — Budget Name: Full Budget Draft (#1)
Principal Investigator: Brian K Gelder — Lead Unit Name: IOWA WATER CENTER - VPR
Sponsor Name: USDA-NATIONAL INSTITUTE OF FOOD & AGRICULTURE (NIFA) — Sponsor Code: 001481

	Period 1	Period 2	Period 3	Total
PERSONNEL				
Salary	78,218.01	80,566.00	82,983.00	241,767.01
Salary/Wages - Faculty	5,144.00	5,298.99	5,458.00	15,900.99
Ali Jammeari	5,144.00 (0.37 CAL)	5,298.99 (0.37 CAL)	5,458.00 (0.37 CAL)	15,900.99
Salary/Wages - Graduate Assistants	41,250.00	42,488.00	43,763.00	127,501.00
Graduate Assistant 1 TBD	30,000.00 (11.82 CAL)	30,900.00 (11.82 CAL)	31,827.00 (11.82 CAL)	92,727.00
Graduate Assistant 2 TBD	11,250.00 (4.43 CAL)	11,588.00 (4.43 CAL)	11,936.00 (4.43 CAL)	34,774.00
Salary/Wages - Professional and Scientific	20,304.01	20,913.01	21,540.00	62,757.02
Brian K Gelder	12,011.01 (1.03 CAL)	12,371.01 (1.03 CAL)	12,742.00 (1.03 CAL)	37,124.02
Daryl E Herzmann	8,293.00 (1.01 CAL)	8,542.00 (1.01 CAL)	8,798.00 (1.01 CAL)	25,633.00

This information can only be exported to a pdf

Step 2: Data Extraction

The spreadsheet information also needs to be extracted into a computational friendly format

Salary/Wages - Professional and Scientific	20,304.01	20,913.01	21,540.00	62,757.02
Brian K Gelder	12,011.01	12,371.01	12,742.00	37,124.02
	(1.03 CAL)	(1.03 CAL)	(1.03 CAL)	
Daryl E Herzmann	8,293.00	8,542.00	8,798.00	25,633.00
	(1.01 CAL)	(1.01 CAL)	(1.01 CAL)	



```
Salaries
                        Full Name  Monthly Percentage  Year 1 Total  Year 2 Total  Year 3 Total
0                     Ali Jannesari                0.37       5144.00       5298.99       5458.0
1         Graduate Assistant 1 TBD                11.82      30000.00      30900.00      31827.0
2         Graduate Assistant 2 TBD                 4.43      11250.00      11588.00      11936.0
3                     Brian K Gelder                1.03      12011.01      12371.01      12742.0
4                    Daryl E Herzmann               1.01       8293.00       8542.00       8798.0
5  Undergrad Student Hourly 1 TBD                   15.76      11520.00      11866.00      12222.0
```

The data extraction is done using Regular Expressions. These expressions had to be hand tailored for each category.

Step 3: Validation

Given that the Streamlyne totals are the “correct” numbers, our tool needs to make sure that the numbers in the spreadsheet are the same as the numbers in the Streamlyne pdf.

The goal for the Validation step is to create a table that looks like this:

```
Salaries----------------------------------------------------------------
                    spreadsheet_name  Difference Year 1 Total  Difference Year 2 Total  Difference Year 3 Total
0                     Brian K Gelder                -0.095833                    -0.01                      0.0
1                      Ali Jannesari                 0.250000                     0.01                      0.0
2            Research Asst-Halftime 1                 0.000000                     0.00                      0.0
3            Research Asst-Halftime 2                 0.000000                     0.00                      0.0
4  Hourly Undergraduate student 1 - 2                 0.000000                     0.00                      0.0
5                     Daryl Herzmann                 0.000000                     0.00                      0.0
```

To do this we need to align the spreadsheet name to the Streamlyne pdf name. This is difficult as they have different categories and different names

Step 3: Validation

Consider the Salaries examples shown in Data Insertion

Spreadsheet Extraction

```
key_personnel-------------------------
        Full Name Monthly Percentage
0  Brian K Gelder                  1
1    Ali Jannesari              0.375
other_personnel-----------------------
                        Position Month
0        Research Asst-Halftime
1        Research Asst-Halftime
2  Hourly Undergraduate student
3                Daryl Herzmann
```

Streamlyne Extraction

```
-----------------------------------
Salaries
                         Full Name
0                     Ali Jannesari
1          Graduate Assistant 1 TBD
2          Graduate Assistant 2 TBD
3                   Brian K Gelder
4                  Daryl E Herzmann
5  Undergrad Student Hourly 1 TBD
```

To solve this problem, a fuzzing tool was used (rapidfuzz) to best match the names together.

Step 4: Insertion & Generation

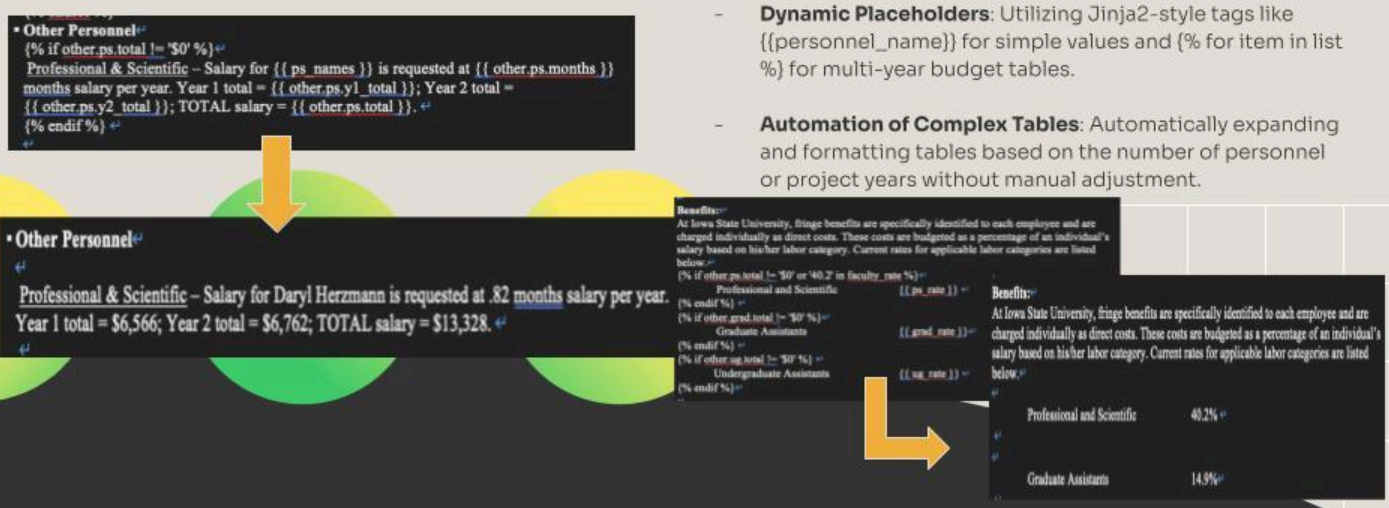
The Synthesis of Data & Template

Seamlessly merging extracted data with agency-compliant Word templates

- **Integration of Outputs**: Combines the computational data structures from our Step 2 with our pre-designed docx template.
- **Variable Mapping**: Automatically injecting specific values, such as personnel names, salaries, and indirect costs into proper sections.
- **Final Generation**: Programmatically rendering the "Output.docx", turning raw figures into a professional, audit-ready budget narrative.



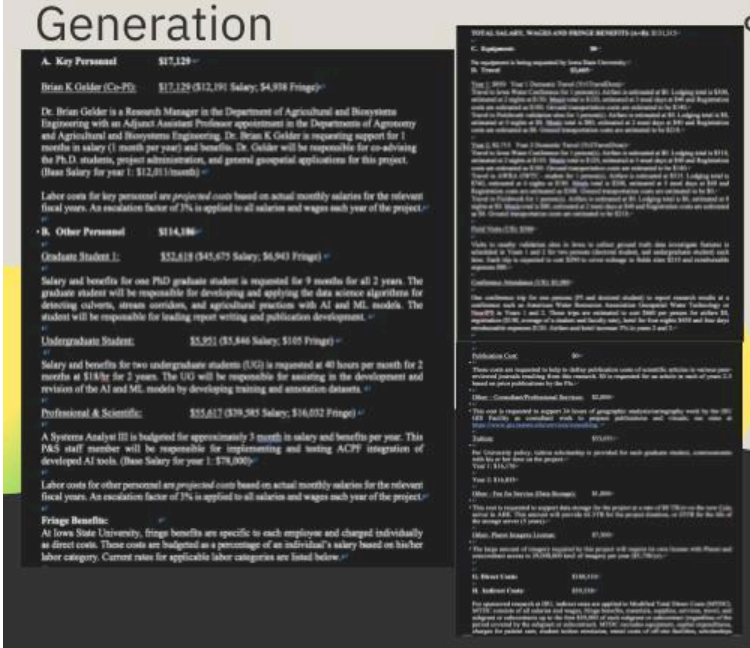
Step 4: Insertion: How code finds the “Blank”



Precision Through Intelligent Mapping

- **The Context Dictionary**: Converting extracted Pandas DataFrames into a Python “Context” dictionary that matches the template’s structure.
- **Dynamic Placeholders**: Utilizing Jinja2-style tags like {{personnel_name}} for simple values and {% for item in list %} for multi-year budget tables.
- **Automation of Complex Tables**: Automatically expanding and formatting tables based on the number of personnel or project years without manual adjustment.

Step 4: Insertion & Generation

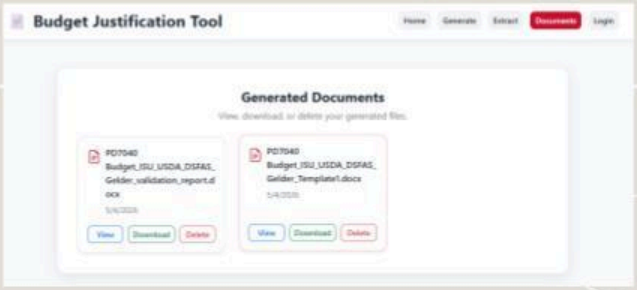


Output Quality

- **Style & Branding Preservation**: The engine inherits all font styles, margins, and institutional headers from the master template. The final document looks exactly like an official ISU research proposal
- **Intelligent Section Handing**: Using conditional logic, the tool automatically hides empty sections. If there is no “Equipment” budget, those headers and tables are removed to keep the narrative clean.
- **Automatic PDF Conversion**: We integrated a headless LibreOffice (soffice) to convert the rendered Word document into a PDF instantly.

Step 5: User Interface pt 2 - Viewing Documents

- Display generated documents
- View (.pdf), download (.docx), and delete
- Automatic .docx -> .pdf conversion
  - LibreOffice
  - Allows in-browser viewing
- Validation reports
  - As a document
  - Native



Zoomed out display



Budget Justification Document



Budget Justification

in refining the operational window of each year and for implementing the Data Management Plan for this project. (Base Salary for year 1: $99,519)

Labor costs for other personnel are *projected* costs based on actual monthly salaries for the relevant fiscal years. An escalation factor of 3% is applied to all salaries and wages each year of the project.

Fringe Benefits:
At Iowa State University, fringe benefits are specific to each employee and charged individually as direct costs. These costs are budgeted as a percentage of an individual's salary based on his/her labor category. Current rates for applicable labor categories are listed below.

Class	Rate	Personnel
Faculty	31.8%	Janssaart
Professional and Scientific	40.5%	Gelder, Herzmann
Graduate Students	15.2%	Doctoral Students
Undergraduate Students	1.8%	Undergraduate Student

TOTAL SALARY, WAGES AND FRINGE BENEFITS (A+B): $292,261

C. Equipment: **$0**

No equipment is being requested by Iowa State University.

D. Travel **$14,702**

Field Visits (US): $870

Visits to nearby validation sites in Iowa to collect ground truth data investigate features is scheduled in Years 1, 2, and 3 for two persons (doctoral student, and undergraduate student) each time. Each trip is expected to cost $290 to cover mileage to fields sites $210 and reimbursable expenses $80.

Conference Attendance (US): $13,832

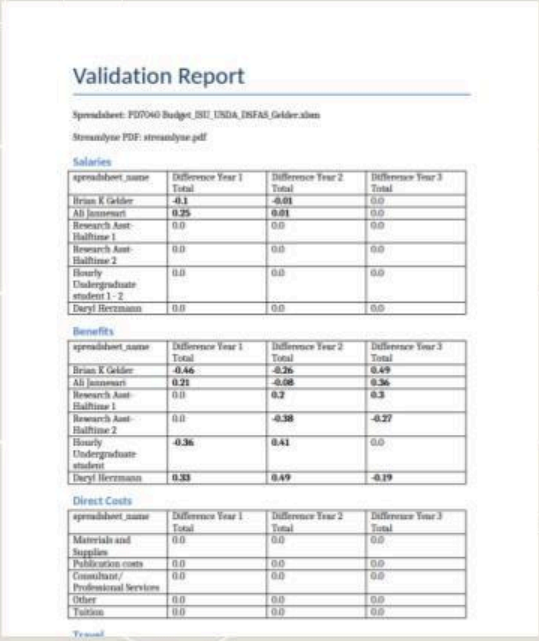
One conference trip for two persons (PI and doctoral student) to report research results at a conference such as American Water Resources Association Geospatial Water Technology or NeurIPS in Years 1, 2, and 3. These trips are estimated to cost $2,020 per person for airfare $500, registration ($600, average of a student and faculty rate), hotel for four nights $540 and four days reimbursable expenses $200. Airfare and hotel increase 3% in years 2 and 3.

F. Other Direct Costs: $146,969

Materials and Supplies: $5,020

Extra details needed

Validation Report Document



Validation Report

Spreadsheet: PD7040 Budget_ISU_USDA_DSFAS_Gelder.xlsm

Streamlyne PDF: streamlyne.pdf

Salaries

spreadsheet_name	Difference Year 1 Total	Difference Year 2 Total	Difference Year 3 Total
Brian K Gelder	**-0.1**	**-0.01**	0.0
Ali Janssaart	**0.25**	**0.01**	0.0
Research Asst-Halftime 1	0.0	0.0	0.0
Research Asst-Halftime 2	0.0	0.0	0.0
Hourly Undergraduate student 1 - 2	0.0	0.0	0.0
Daryl Herzmann	0.0	0.0	0.0

Benefits

spreadsheet_name	Difference Year 1 Total	Difference Year 2 Total	Difference Year 3 Total
Brian K Gelder	**-0.46**	**-0.26**	**0.49**
Ali Janssaart	**0.21**	**-0.08**	**0.36**
Research Asst-Halftime 1	0.0	**0.2**	**0.3**
Research Asst-Halftime 2	0.0	**-0.38**	**-0.27**
Hourly Undergraduate student	**-0.36**	**0.41**	0.0
Daryl Herzmann	**0.33**	**0.49**	**-0.19**

Direct Costs

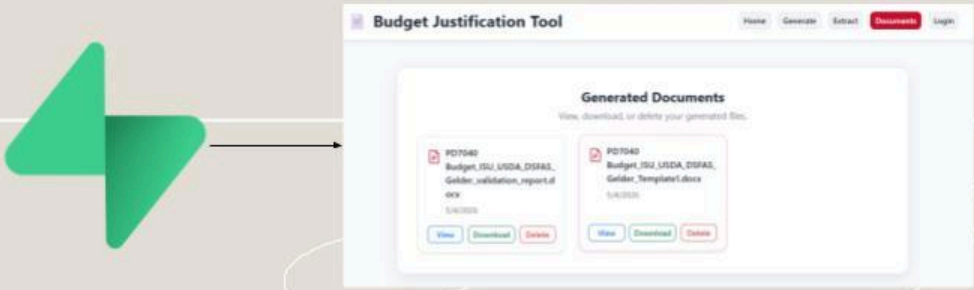
spreadsheet_name	Difference Year 1 Total	Difference Year 2 Total	Difference Year 3 Total
Materials and Supplies	0.0	0.0	0.0
Publication costs	0.0	0.0	0.0
Consultant/ Professional Services	0.0	0.0	0.0
Other	0.0	0.0	0.0
Tuition	0.0	0.0	0.0

Travel

Step 5: User Interface pt 2 - Viewing Documents

User-Specific Document Storage

- Integrated with Supabase cloud storage
- Documents tied to user when logged-in
- Supports multiple users



Step 6: User Customization

Additional Features:

- Added Section to End
- Reusability
- Templates

Step 6: Add Section

2 ways:

- Create own section name, automatically added to end of document
- Specify section and line number, text automatically added after line number
- Gemini AI used for editing

Facilities and Administrative Costs (Indirect Costs):

For sponsored research at ISU, indirect rates are applied to Modified Total Direct Costs (MTDC). MTDC consists of all salaries and wages, fringe benefits, materials, supplies, services, travel, and subgrant or subcontracts up to the first $25,000 of each subgrant or subcontract (regardless of the period covered by the sub grant or subcontract). MTDC excludes equipment, capital expenditures, charges for patient care, student tuition remission, rental costs of off-site facilities, scholarships and fellowships, as well as the portion of each subgrant and subcontract in excess of $25,000. Current rates are provided below. These rates were approved by the Department of Health and Human Services (effective 07/01/2018 – 06/30/2024, provisional until revised).

On-campus – Organized Research 42.9%

Additional Information

This is the budget we have set for this year; however, this project is set to last for at least a few years and therefore, the budget is subject to change.

B. Other Personnel **$219,144**

Graduate Student 1: $106,822 ($92,727 Salary; $14,095 Fringe)

Salary and benefits for one PhD graduate student is requested for 12 months for all 3 years. The doctoral students will be responsible for developing and applying the data science algorithms for detecting agricultural practices with AI and ML models. The student will be responsible for leading report writing and publication development.

Step 6: Reusability

Section Extraction

Budget Justification

A. Key Personnel **$73,117**

Brian K Gelder (Co-PI): $37,124 ($27,843 Salary; $9,281 Fringe)

Dr. Brian K Gelder is a Research Manager in the Department of Agricultural and Biosystems Engineering with an Adjunct Assistant Professor appointment in the Departments of Agronomy and Agricultural and Biosystems Engineering. Dr. Brian K Gelder is requesting support for 1.00 months in salary (1 month per year) and benefits. Dr. Brian K Gelder will be responsible for co-advising the Ph.D. students, project administration, and general geospatial applications for this project. (Base Salary for year 1: $12,375/month)

Labor costs for key personnel are *projected costs* based on actual monthly salaries for the relevant fiscal years. An escalation factor of 3% is applied to all salaries and wages each year of the project.

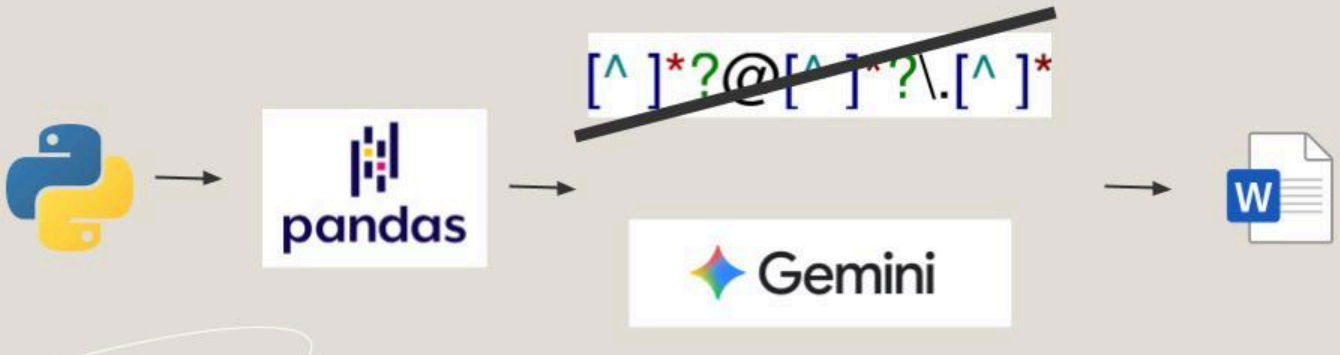
Brian K Gelder (Co-PI): $37,124 ($27,843 Salary; $9,281 Fringe)

Dr. Brian K Gelder is a Research Manager in the Department of Agricultural and Biosystems Engineering with an Adjunct Assistant Professor appointment in the Departments of Agronomy and Agricultural and Biosystems Engineering. Dr. Brian K Gelder is requesting support for 1.00 months in salary (1 month per year) and benefits. Dr. Brian K Gelder will be responsible for co-advising the Ph.D. students, project administration, and general geospatial applications for this project. (Base Salary for year 1: $12,375/month)

Labor costs for key personnel are *projected costs* based on actual monthly salaries for the relevant fiscal years. An escalation factor of 3% is applied to all salaries and wages each year of the project.

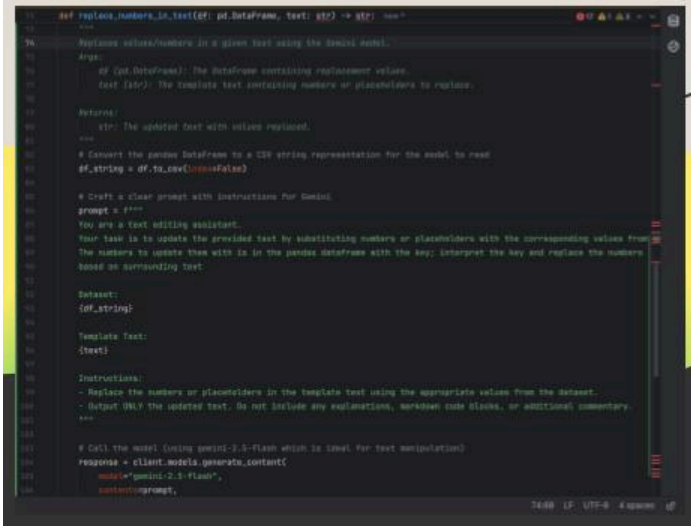
Step 6: Reusability

Switch from regex to Gemini AI



Step 6: Reusability

Switched from regex to Gemini AI

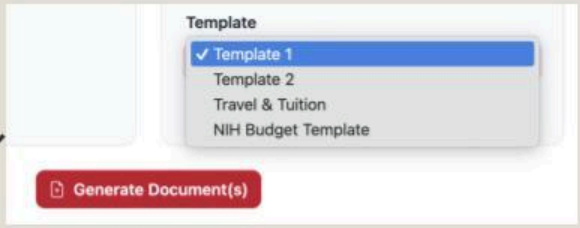


Brian Gelder (Co-PI): $6,336 ($4,752 Salary; $1,584 Fringe)

Dr. Brian Gelder is a Research Manager in the Department of Agricultural and Biosystems Engineering with an Adjunct Assistant Professor appointment in the Departments of Agronomy and Agricultural and Biosystems Engineering. Dr. Brian Gelder is requesting support for 0.27 months in salary (1 month per year) and benefits. Dr. Brian Gelder will be responsible for co-advising the Ph.D. students, project administration, and general geospatial applications for this project. (Base Salary for year 1: $11,661/month)

Labor costs for key personnel are projected costs based on actual monthly salaries for the relevant fiscal years. An escalation factor of 3% is applied to all salaries and wages each year of the project.

Step 6: Template Choice



Budget Justification

A. Key Personnel **{{ total_key_personnel_cost }}**

{{ pi_name }} (Co-PI): {{ pi_total }} ({{ pi_salary_sum }} Salary; {{ pi_fringe_sum }} Fringe)

Dr. {{ pi_name }} is a <Add Detail> in the Department of <Add Detail>. Dr. {{ pi_name }} is requesting support for {{ pi_months }} months in salary (1 month per year) and benefits. Dr. {{ pi_name }} will be responsible for <Add Detail>. (Base Salary for year 1: {{ pi_base_salary }}/month)

{%p for p in key_personnel[1:] %}

{{ p.name }} (Co-PI): {{ p.total }} ({{ p.salary }} Salary; {{ p.fringe }} Fringe)

{{ p.name }} is a <Add Detail> at <Add Detail>. Dr. {{ p.last_name }} is requesting support for {{ p.months }} months per year and benefits. Dr. {{ p.last_name }} will be responsible for <Add Detail>. (9-month Base Salary for year 1: {{ p.base_salary_9mo }})

{%p endfor %}

Labor costs for key personnel are *projected costs* based on actual monthly salaries for the relevant fiscal years. An escalation factor of 3% is applied to all salaries and wages each year of the project.

B. Other Personnel **{{ total_other_personnel_cost }}**

{% for p in other_personnel %}
{{ p.role }}: {{ p.total }} ({{ p.salary }} Salary; {{ p.fringe }} Fringe)
{% if "Graduate" in p.role %}
Salary and benefits for one PhD graduate student is requested for {{ p.months }} months for all {{ n_years }} years. The doctoral students will be responsible for developing and applying the data science algorithms for detecting agricultural practices with AI and ML models. The student will be responsible for leading report writing and publication development.
{% elif "Undergraduate" in p.role %}
Salary and benefits for two undergraduate student (UG) is requested at {{ p.hours }} hours per

8. Other Direct Costs

Tuition: {{ tuition_total }}

Per University policy, tuition scholarship is provided for each graduate student, commensurate with his or her time on the project.

{%p for line in tuition_lines %}

{{ line.text }}

{%p endfor %}

Other - Planet Imagery License: {{ planet_total }}

The large amount of imagery required by this project will require its own license with Planet and concomitant access to its PlanetScope and/or SkySat archive at {{ planet_rate }}.

G. Direct Costs: **{{ total_direct_sum }}**

Indirect Costs:

Indirect costs are included at a rate of {{ indirect_rate }} of modified total direct costs (MTDC), which is Iowa State University's on-campus negotiated rate for research. Total indirect costs: {{ total_indirect_sum }}.

MTDC consists of all salaries and wages, fringe benefits, materials, supplies, services, travel, and subgrant or subcontracts up to the first $50,000 of each subgrant or subcontract (regardless of the period covered by the sub grant or subcontract). MTDC excludes equipment, capital expenditures, charges for patient care, student tuition remission, rental costs of off-site facilities, scholarships and fellowships, as well as the portion of each subgrant and subcontract in excess of $50,000.

Total Direct and Indirect Costs: **{{ total_budget_sum }}**

Video Demo




Thank you