Instructlab SDG Review Prototype

Instructlab x Graphite Digital

November 2024

About Us

1

Graphite Digital

A student organization providing high-quality R&D and tech consulting services for companies.

Previous semester projects include:

- Creating websites for startups
- Designing a data auditing platform from zero-to-one
- Researching and advising on AI model selection and implementation
- Optimizing logistical supply chain data sets
- LLM benchmarking (in progress!)

Emily Gao
Director of Graphite Digital
Pomona College
Economics and Cognitive Science/Design

Sadhvi Narayanan
Co-Team Lead
Harvey Mudd College
Computer Science

Andy Xu
Co-Team Lead
Harvey Mudd College
Computer Science and Math

AJ Matheson-Lieber
Team Member
Claremont McKenna College
Philosophy, Math, and Computer Science

Ellie Lian
Team Member
Pomona College
Economics and Data Science

Felix Peng
Team Member
Harvey Mudd College
Engineering

Current Problem Space

How do users currently conduct synthetic data review, and how can we help to alleviate any pain points?

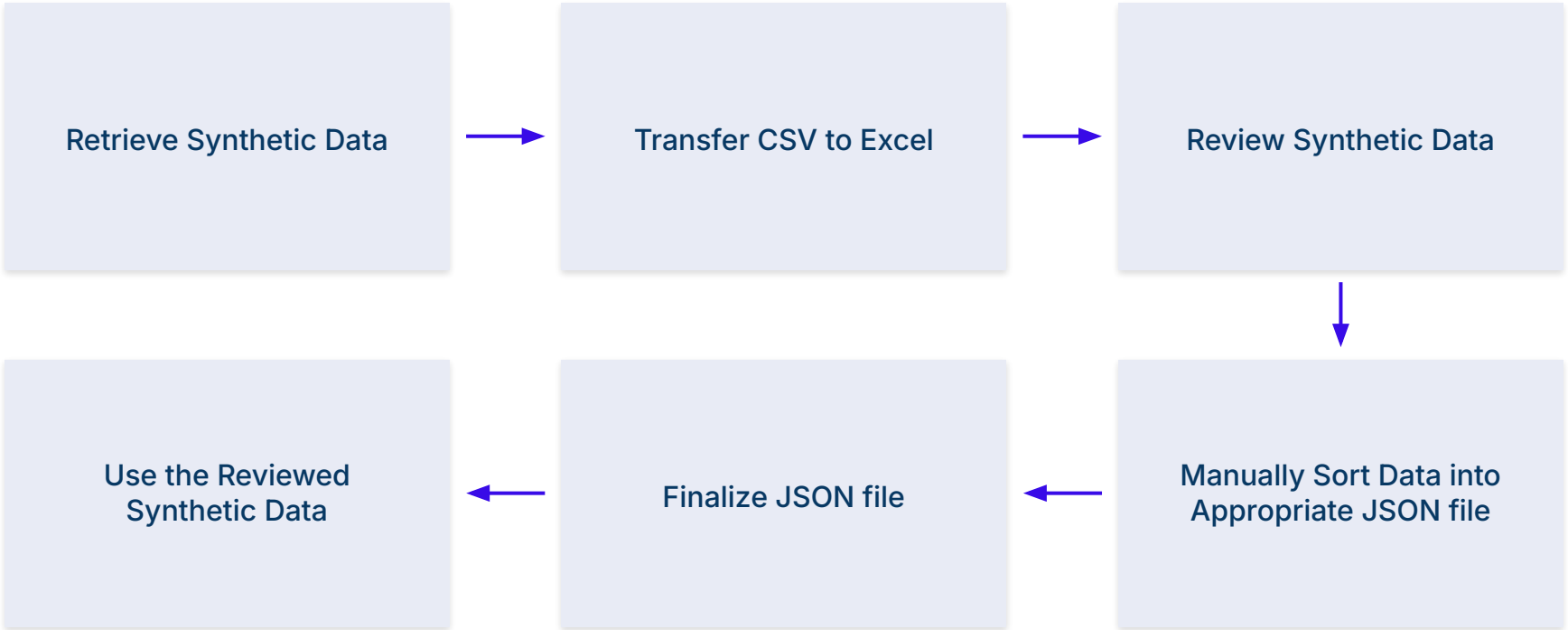
2

The Problem

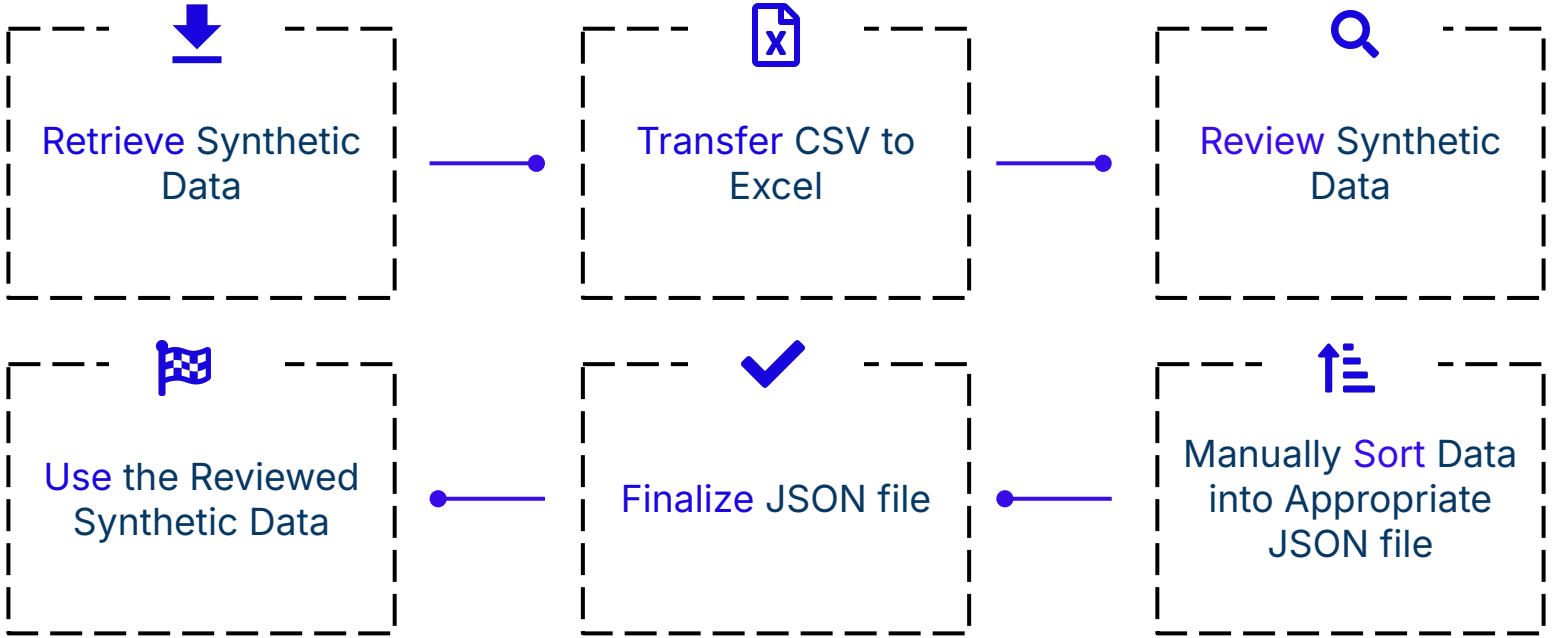
> "If I'm unsure about the process or answer, I just hope that someone gets around to reviewing the question."
>
> *– IBM Developer*

Users currently **lack a standardized process** or location to review, edit, and approve synthetically generated data.

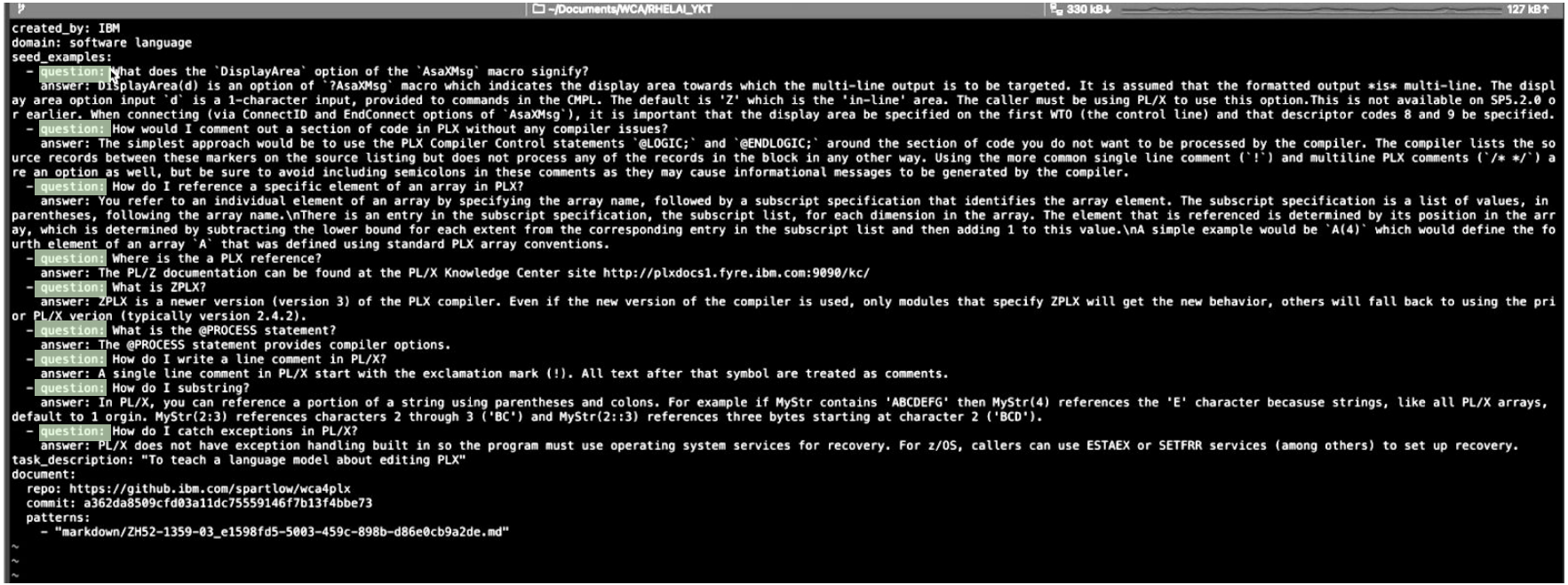
Example Reviewal Process



Example Reviewal Process



Example Reviewal Process



Source: IBM

Example Reviewal Process

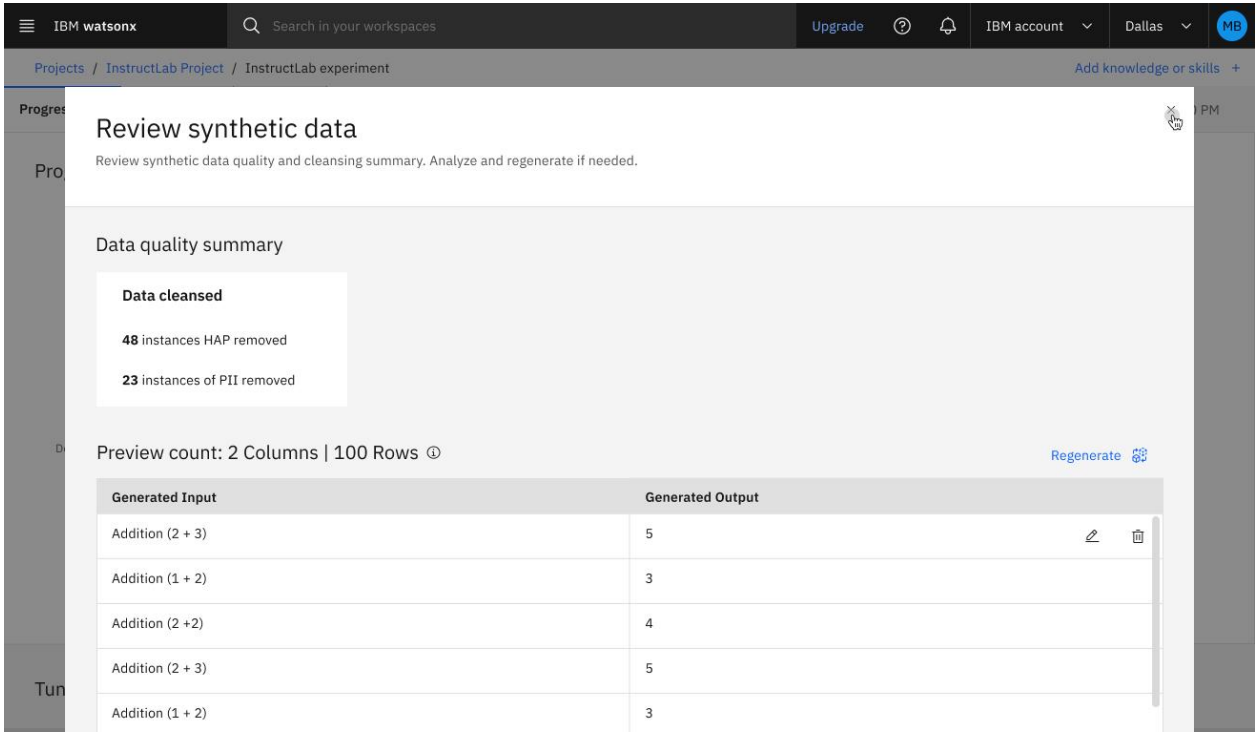
	A	B	C	D	
	Prompt	answerA (GROUND TRUTH)	AnswerB	LLMasJ	Christoph
	it looks like it's having an issue with the space in the \"name\" field\n\n+ for CURRENT_LIBRARY in $(echo \"$CURRENT_LIBRARIES\" | jq -c '[]?\n+ echo '{\"artifactId\": \"aho-corasick\",\"version\":\"1.1.2\",\"licenses\":[{\"name\":\"Unlicense\"\n{\"artifactId\":\"aho-corasick\", \"version\":\"1.1.2\",\"licenses\":[{\"name\":\"Unlicense\n++ echo '{\"artifactId\":\"aho-corasick\",\"version\":\"1.1.2\", \"licenses\":[{\"name\":\"Unlicense\n++ jq -r .artifactId\nparse error: Unfinished string at EOF at line 2, column 0\n	You can fix this by adding a space after the opening quote of the \"name\" field in the JSON file:\n\n```json\n// Assisted by WCA@IBM\n// Latest GenAI contribution: ibm/granite-20b-code-instruct-v2\n{\n \"libraries\": {\n {\n \"artifactId\": \"aho-corasick\",\n \"version\": \"1.1.3\",\n \"licenses\": [\n {\n \"name\": \"Unlicense\"\n },\n {\n \"name\": \"BSD-3-Clause\"\n }\n]\n },\n ...\n }\n}\n```\n	That error message is indicating that there is an issue with the syntax of the JSON string being printed to the console. This typically occurs when there is an unmatched quote or escape character in the string.\n\nLooking at the code snippet you provided, it seems that there might be an issue with the escape sequences used in the JSON string. Specifically, the escape sequence for a single quote within a double-quoted string should be `\\'`, but it looks like the string is being printed with a single quote at the end, which is causing the error.\n\nTo fix this issue, you can try enclosing the JSON string in single quotes (`'`), as follows:\n\n```bash\n+ for CURRENT_LIBRARY in $(echo '$CURRENT_LIBRARIES' | jq -c '[]?\n+ echo '{\"artifactId\": \"aho-corasick\",\"version\":\"1.1.2\",\"licenses\":[{\"name\":\"Unlicense\n++ echo '{\"artifactId\":\"aho-corasick\", \"version\":\"1.1.2\",\"licenses\":[{\"name\":\"Unlicense\n++ jq -r .artifactId\naho-corasick\n```\n\nBy enclosing the JSON string in single quotes, we can avoid any issues with escape sequences and the JSON string should be properly printed to the console.\n\nI hope this helps! Let me know if you have any further questions.	nAfter comparing the two responses, I find that Assistant A provides a more helpful, relevant, and accurate answer, by pointing out the exact issue and offering a straightforward fix. They highlight the missing space in the \"name\" field of the licenses array and present the corrected JSON object. Assistant B, on the other hand, focuses on the incorrect escape sequence as the problem, the assistant offers an alternative solution using single quotes - but fails to address the missing space. Although the suggested solution from Assistant B might help in some cases, it does not provide a definitive fix for the issue at hand. In conclusion, I deem [[A]] as the better assistant in this scenario.	Disagree
	Create a C++ ro[utine that finds the IP address of the local host on Linux\n	Sure! Here's an example C++ routine that uses the `gethostbyname` function from the `sys/socket.h` and `netdb.h` header files to retrieve the IP address of the local host on Linux:\n\n```\n// Assisted by WCA@IBM\n// Latent GenAI contribution: ibm/granite-20b-code-instruct-v2\n#include <sys/socket.h>\n#include <netdb.h>\n#include <iostream>\n\nint main() {\n // Retrieve the local host's IP address\n struct hostent *pHost = gethostbyname(\"localhost\");\n if (pHost == NULL) {\n std::cerr << \"Error retrieving localhost's IP address: \" << hstrerror(h_errno) << std::endl;\n return 1;\n }\n\n // Print the IP address in dotted decimal format\n for (int i = 0; pHost->h_addr_list[i] != 0; ++i) {\n std::cout << \"IP address #\" << (i + 1) << \": \";\n char szAddress[INET_ADDRSTRLEN];\n inet_ntop(AF_INET, pHost->h_addr_list[i], szAddress, sizeof(szAddress));\n std::cout << szAddress << std::endl;\n }\n\n return 0;\n}\n```\n\nThis code first includes the necessary headers for using the `gethostbyname` function, including `sys/socket.h`, `netdb.h`, and `iostream`. It then defines a main function that retrieves the local host's IP address using the `gethostbyname` function, which takes a string containing the host name as its argument. Finally, the code prints the IP address in dotted decimal format using the `inet_ntop` function.	Here's an example C++ routine that uses the `getaddrinfo` and `inet_ntop` functions from the `netdb.h` and `inet_ntop.h` headers file to retrieve the IP address of the local host on Linux:\n\n```\n++#include <netdb.h>\n#include <unistd.h>\n#include <iostream>\n\nint main() {\n struct addrinfohints, *pHints, *pResult;\n\n // Initialize the hints structure\n memset(&hints, 0, sizeof(hints));\n hints.ai_family = AF_UNSPEC;\n hints.ai_socktype = SOCK_STREAM;\n hints.ai_flags = AI_PASSIVE;\n\n // Retrieve the local host's IP address\n int iResult = getaddrinfo(NULL, \"http\", &hints, &pResult);\n if (iResult != 0) {\n std::cerr << \"getaddrinfo failed: \" << gai_strerror(iResult) << std::endl;\n return 1;\n }\n\n // Print the IP address of the local host\n for (struct addrinfo *piterator = pResult; piterator != NULL; piterator = piterator->ai_next) {\n char szAddress[INET6_ADDRSTRLEN];\n inet_ntop(piterator->ai_family, piterator->ai_addr->sa_data, szAddress, sizeof(szAddress));\n std::cout << \"IP address: \" << szAddress << std::endl;\n }\n\n freeaddrinfo(pResult);\n return 0;\n}\n```\n\nThis code first includes the necessary headers for using the `getaddrinfo` and `inet_ntop` functions. It then initializes the hints structure using the `memset` function to set the desired address family, socket type, and flags. The code then retrieves the local host's IP address using the `getaddrinfo` function, passing `NULL` as the host name argument to retrieve the local host's IP address. The code prints the IP address of the local host in dotted decimal format using the `inet_ntop` function. Finally, the code frees the memory allocated for the address information using the `freeaddrinfo` function.	Assistant A provides a more detailed and in-depth response compared to Assistant B. Both assistants demonstrate the use of valid C++ code to retrieve the local host's IP address, and the code itself is accurate and relevant to the user's question. However, Assistant A's response provides a more thorough explanation of the code itself, with comments describing the purpose and usage of each component. Additionally, Assistant A's code includes the use of the `memset` and `freeaddrinfo` functions, providing a more complete and robust solution. As such, Assistant A's response is more helpful and insightful, and better meets the user's needs. The final verdict is [[A]]	??? Not s
				The responses of both assistants share some common elements, such as providing sample implementations of a TypeScript function to convert an OData datetime string to an ISO date string. However, their implementation approaches differ, and so do their explanations.\n\nAssistant A's response includes an explicit function declaration, which features a brief description of the function's goal and behavior. It walks the user through the process of parsing the input string by breaking it into date and time parts based on defined separators. This explanation of string manipulation helps users with limited knowledge of	

Source: IBM

Example Reviewal Process

Prompt	answerA (ground truth)	answerB	LLMasJ

Integration Into Current Watsonx UI



Current User Pain Points

Unintuitive	Slow	Uncollaborative
• No defined process • Reliance on knowledge of JSON and CSV formats	• Manual data management • Inefficient reviewal process	• No integrated mode of collaboration • Time and expertise wasted

Users need a more practical system for retrieving synthetic data.

What primary functions did we prioritize, and what assumptions did we make?

List and Modular Views

Focused on quick scan and review, with readily available actions in list view.

Collaborative Team Tools (filtering, commenting)

Focused on cross-functional tools to see status of others' review + tag others in process.

Approving, Denying, Editing

Focused on displaying these as main functions.

Solution Considerations

List and Modular Views:	Quick scan and review, with readily available quick-actions in list view.
Collaborative Tooling:	Tools to see status of others' reviews + ability to comment and tag others.
Approving, Denying, Editing:	Displaying these as main functions, with a focus on modular view editing.

Design Walkthrough

Mid-fidelity prototype

3

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

What is the purpose of the program status word's (PSW) program mask in MVS Programming?	The PSW program mask in MVS Programming provides bits to disable program exceptions. When MVS gives control to programs, these bits are usually off, causing many program exceptions.	
What are the two MVS services that enable program exceptions and allow a user exit routine to receive control when those exceptions occur?	ESPIE and SPIE services.	
How does the SPIE macro enable program exceptions in MVS Programming?	The SPIE macro enables program exceptions by specifying an exception for which the interruption has been disabled.	
In which addressing mode does the exit routine receive control for the SPIE macro when an interrupt occurs?	The exit routine receives control in 24-bit addressing mode for the SPIE macro when an interrupt occurs.	
What is the difference in addressing mode between the	For the SPIE macro, the exit routine receives control only if	

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

Reviewer

- ■ Reviewer 1 (self)
- ☐ Reviewer 2
- ☐ Reviewer 3
- ☐ Reviewer 4
- ☐ Reviewer 5
- ☐ Reviewer 6

What is the purpose of the prog (PSW) program mask in MVS Pro	rogram mask in MVS Programming provides bits to ogram exceptions. When MVS gives control to these bits are usually off, causing many program exceptions.	☑ ▬ 💬
What are the two MVS services that enable program exceptions and allow a user exit routine to receive control when those exceptions occur?	ESPIE and SPIE services.	☑ ▬ 💬
How does the SPIE macro enable program exceptions in MVS Programming?	The SPIE macro enables program exceptions by specifying an exception for which the interruption has been disabled.	☑ ▬ 💬
In which addressing mode does the exit routine receive control for the SPIE macro when an interrupt occurs?	The exit routine receives control in 24-bit addressing mode for the SPIE macro when an interrupt occurs.	☑ ▬ 💬
What is the difference in addressing mode between the	For the SPIE macro, the exit routine receives control only if	

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

What is the purpose of the program status word's (PSW) program mask in MVS Programming?	The PSW program mask in MVS Programming provides bits to disable program exceptions. When MVS gives control to programs, these bits are usually off, causing many program exceptions.	
What are the two MVS services that enable program exceptions and allow a user exit routine to receive control when those exceptions occur?	ESPIE and SPIE services.	
How does the SPIE macro enable program exceptions in MVS Programming?	The SPIE macro enables program exceptions by specifying an exception for which the interruption has been disabled.	
In which addressing mode does the exit routine receive control for the SPIE macro when an interrupt occurs?	The exit routine receives control in 24-bit addressing mode for the SPIE macro when an interrupt occurs.	
What is the difference in addressing mode between the	For the SPIE macro, the exit routine receives control only if	

Comments

No comments...

Include any comments here...

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

What are program exceptions in MVS Programming and what is their significance?	Program exceptions in MVS Programming refer to specific conditions that cause a program interruption. These can include incorrect parameters, exceptional results, or other issues that disrupt the normal flow of a program ...	
What is the purpose of the program status word (PSW) program mask in MVS Programming?	The program status word (PSW) program mask in MVS Programming provides bits to control certain program exceptions. When MVS gives control to programs, these bits are usually off, disabling the program exceptions. By enabling specific bits in the PSW program mask, ...	
How do the ESPIE and SPIE services enable program exceptions in MVS Programming?	The ESPIE and SPIE services in MVS Programming enable program exceptions and allow a user exit routine to receive control when those exceptions occur. By issuing the SPIE or ESPIE macro, programmers can specify their own exit routine to be given control ...	
What is the difference between the SPIE and ESPIE macros in MVS Programming?	The SPIE macro allows the exit routine to receive control only if the interrupted program is in primary address space control (ASC) mode, while the ESPIE macro allows the exit routine to receive control if the interrupted program is in either primary or access register (AR) ASC mode ...	
What is the purpose of the program interruption control	The program interruption control area (PICA) in MVS Programming contains the new program mask for the	

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

What are program exceptions in MVS Programming and what is their significance?	Program exceptions in MVS Programming refer to specific conditions that cause a program interruption. These can include incorrect parameters, exceptional results, or other issues that disrupt the normal flo	
What is the purpose of the program status word (PSW) program mask in MVS Programming?	The program status word (PSW) p Programming provides bits to con exceptions. When MVS gives cont are usually off, disabling the prog specific bits in the PSW program	
How do the ESPIE and SPIE services enable program exceptions in MVS Programming?	The ESPIE and SPIE services in MVS Programming enable program exceptions and allow a user exit routine to receive control when those exceptions occur. By issuing the SPIE or ESPIE macro, programmers can specify their own exit routine to be given control ...	
What is the difference between the SPIE and ESPIE macros in MVS Programming?	The SPIE macro allows the exit routine to receive control only if the interrupted program is in primary address space control (ASC) mode, while the ESPIE macro allows the exit routine to receive control if the interrupted program is in either primary or access register (AR) ASC mode ...	
What is the purpose of the program interruption control	The program interruption control area (PICA) in MVS Programming contains the new program mask for the interruption types that can be disabled in the PSW, the	

Comments

@Reviewer 2 Does this answer look okay to you?

Review Data

View, approve, deny, edit, and comment on generated data.

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

What is the purpose of the program status word's (PSW) program mask in MVS Programming?	The PSW program mask in MVS Programming provides bits to disable program exceptions. When MVS gives control to programs, these bits are usually off, causing many program exceptions.	
What are the two MVS services that enable program exceptions and allow a user exit routine to receive control when those exceptions occur?	ESPIE and SPIE services.	
How does the SPIE macro enable program exceptions in MVS Programming?	The SPIE macro enables program exceptions by specifying an exception for which the interruption has been disabled.	
In which addressing mode does the exit routine receive control for the SPIE macro when an interrupt occurs?	The exit routine receives control in 24-bit addressing mode for the SPIE macro when an interrupt occurs.	
What is the difference in addressing mode between the	For the SPIE macro, the exit routine receives control only if	

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

Previous 1/9 Next

What is the purpose of the program status word's (PSW) program mask in MVS Programming?

The PSW program mask in MVS Programming provides bits to disable program exceptions. When MVS gives control to programs, these bits are usually off, causing many program exceptions.

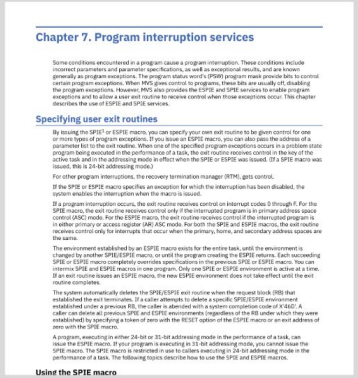
Comments

No comments...

Include any comments here...

Approve | Deny

Reference.pdf



Chapter 7. Program interruption services

Some conditions encountered in a program cause a program interruption. These conditions include incorrect parameters and parameter specifications, as well as exceptional results, and are known generally as program exceptions. The program status word's (PSW) program mask provide bits to control certain program exceptions. When MVS gives control to programs, these bits are usually off, disabling the program exceptions. However, MVS also provides the ESPIE and SPIE services to enable program exceptions and to allow a user exit routine to receive control when those exceptions occur. This chapter describes the use of ESPIE and SPIE services.

Specifying user exit routines

By issuing the SPIE[1] or ESPIE macro, you can specify your own exit routine to be given control for one or more types of program exceptions. If you issue an ESPIE macro, you can also pass the address of a parameter list to the exit routine. When one of the specified program exceptions occurs in a problem state program being executed in the performance of a task, the exit routine receives control in the key of the active task and in the addressing mode in effect when the SPIE or ESPIE was issued. (If a SPIE macro was issued, this is 24-bit addressing mode.)

For other program interruptions, the recovery termination manager (RTM), gets control.

If the SPIE or ESPIE macro specifies an exception for which the interruption has been disabled, the system enables the interruption when the macro is issued.

If a program interruption occurs, the exit routine receives control on interrupt codes 0 through F. For the SPIE macro, the exit routine receives control only if the interrupted program is in primary address space control (ASC) mode. For the ESPIE macro, the exit routine receives control if the interrupted program is in either primary or access register (AR) ASC mode. For both the SPIE and ESPIE macros, the exit routine receives control only for interrupts that occur when the primary, home, and secondary address spaces are the same.

The environment established by an ESPIE macro exists for the entire task, until the environment is changed by another SPIE/ESPIE macro, or until the program creating the ESPIE returns. Each succeeding SPIE or ESPIE macro completely overrides specifications in the previous SPIE or ESPIE macro. You can intermix SPIE and ESPIE macros in one program. Only one SPIE or ESPIE environment is active at a time. If an exit routine issues an ESPIE macro, the new ESPIE environment does not take effect until the exit routine completes.

The system automatically deletes the SPIE/ESPIE exit routine when the request block (RB) that established the exit terminates. If a caller attempts to delete a specific SPIE/ESPIE environment established under a previous RB, the caller is abended with a system completion code of X'46D'. A caller can delete all previous SPIE and ESPIE environments (regardless of the RB under which they were established) by specifying a token of zero with the RESET option of the ESPIE macro or an exit address of zero with the SPIE macro.

A program, executing in either 24-bit or 31-bit addressing mode in the performance of a task, can issue the ESPIE macro. If your program is executing in 31-bit addressing mode, you cannot issue the SPIE macro. The SPIE macro is restricted in use to callers executing in 24-bit addressing mode in the performance of a task. The following topics describe how to use the SPIE and ESPIE macros.

Using the SPIE macro

Review Data

View, approve, deny, edit, and comment on generated data.

To Review (9) | Reviewed (10)

Previous 1/9 Next

What is the purpose of the program status word's (PSW) program mask in MVS Programming?

The PSW program mask in MVS Programming provides bits to control certain program exceptions. When MVS gives control to programs, these bits are usually off, disabling the program exceptions.

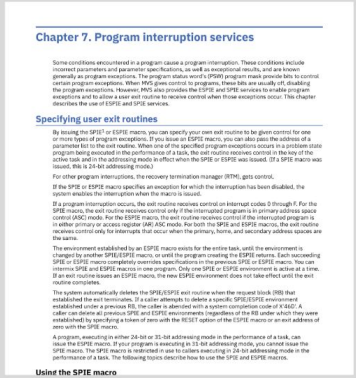
Comments

No comments...

Include any comments here...

Approve | Deny

Reference.pdf



Chapter 7. Program interruption services

Some conditions encountered in a program cause a program interruption. These conditions include incorrect parameters and parameter specifications, as well as exceptional results, and are known generally as program exceptions. The program status word's (PSW) program mask provide bits to control certain program exceptions. When MVS gives control to programs, these bits are usually off, disabling the program exceptions. However, MVS also provides the ESPIE and SPIE services to enable program exceptions and to allow a user exit routine to receive control when those exceptions occur. This chapter describes the use of ESPIE and SPIE services.

Specifying user exit routines

By issuing the SPIE[1] or ESPIE macro, you can specify your own exit routine to be given control for one or more types of program exceptions. If you issue an ESPIE macro, you can also pass the address of a parameter list to the exit routine. When one of the specified program exceptions occurs in a problem state program being executed in the performance of a task, the exit routine receives control in the key of the active task and in the addressing mode in effect when the SPIE or ESPIE was issued. (If a SPIE macro was issued, this is 24-bit addressing mode.)

For other program interruptions, the recovery termination manager (RTM), gets control.

If the SPIE or ESPIE macro specifies an exception for which the interruption has been disabled, the system enables the interruption when the macro is issued.

If a program interruption occurs, the exit routine receives control on interrupt codes 0 through F. For the SPIE macro, the exit routine receives control only if the interrupted program is in primary address space control (ASC) mode. For the ESPIE macro, the exit routine receives control if the interrupted program is in either primary or access register (AR) ASC mode. For both the SPIE and ESPIE macros, the exit routine receives control only for interrupts that occur when the primary, home, and secondary address spaces are the same.

The environment established by an ESPIE macro exists for the entire task, until the environment is changed by another SPIE/ESPIE macro, or until the program creating the ESPIE returns. Each succeeding SPIE or ESPIE macro completely overrides specifications in the previous SPIE or ESPIE macro. You can intermix SPIE and ESPIE macros in one program. Only one SPIE or ESPIE environment is active at a time. If an exit routine issues an ESPIE macro, the new ESPIE environment does not take effect until the exit routine completes.

The system automatically deletes the SPIE/ESPIE exit routine when the request block (RB) that established the exit terminates. If a caller attempts to delete a specific SPIE/ESPIE environment established under a previous RB, the caller is abended with a system completion code of X'46D'. A caller can delete all previous SPIE and ESPIE environments (regardless of the RB under which they were established) by specifying a token of zero with the RESET option of the ESPIE macro or an exit address of zero with the SPIE macro.

A program, executing in either 24-bit or 31-bit addressing mode in the performance of a task, can issue the ESPIE macro. If your program is executing in 31-bit addressing mode, you cannot issue the SPIE macro. The SPIE macro is restricted in use to callers executing in 24-bit addressing mode in the performance of a task. The following topics describe how to use the SPIE and ESPIE macros.

Using the SPIE macro

Future Iterations

Keyboard Accessibility

Hotkeys were emphasized by users for efficiency.

Customizable keybinds, such as:

- Up, down, left, right
- Delete and return

Could this be a question for accessibility experts?

Manager Dashboard

Users suggested implementing a manager dashboard with a statistical overview:

- # approved
- # denied
- Commenting frequency
- Time per question

Additional Filter

Users expressed interest in the number of approved or denied questions to assess SDG quality.

This could look like:

- Approved (43), Denied (4)

Progress Overview

Completed Steps

Initial Exploration

↓

Lo-fi Design

↓

Mid-fi Design

↓

User Testing

Next Steps

Option 1: Project Transfer to IBM

Hand over prototype to IBM to begin development

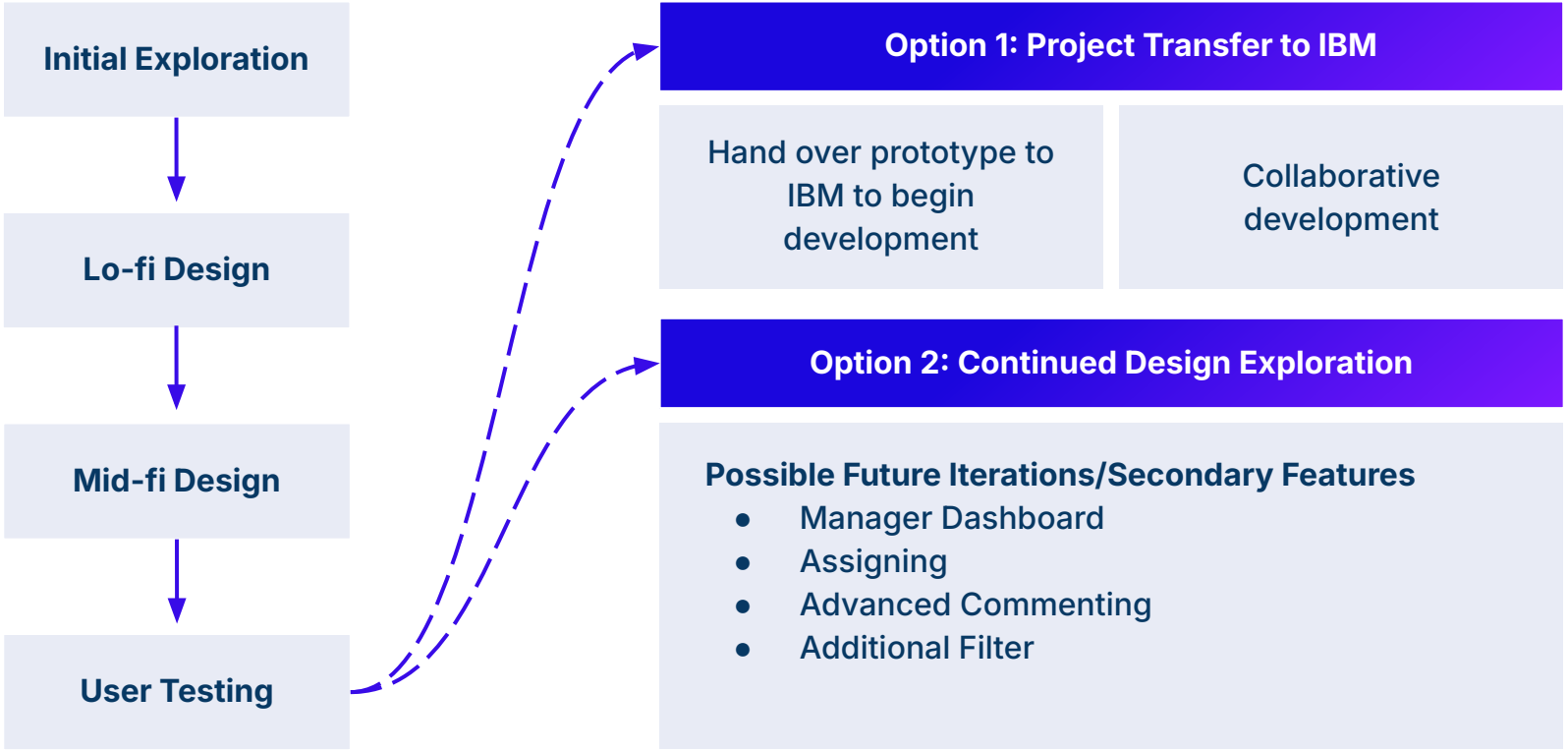
Collaborative development

Option 2: Continued Design Exploration

Possible Future Iterations/Secondary Features:

- Manager Dashboard
- Assigning
- Advanced Commenting
- Additional Filter

Next Steps


Initial Exploration
Lo-fi Design
Mid-fi Design
User Testing
Option 1: Project Transfer to IBM
Hand over prototype to IBM to begin development
Collaborative development
Option 2: Continued Design Exploration
Possible Future Iterations/Secondary Features
Manager Dashboard
Assigning
Advanced Commenting
Additional Filter

“ This is a big step forwards from what we’ve been doing in the past — a large improvement! ”

– Jacob Engelbrecht,
Backend IBM SWE

Thank you!

Instructlab x Graphite Digital

November 2024

Figma link

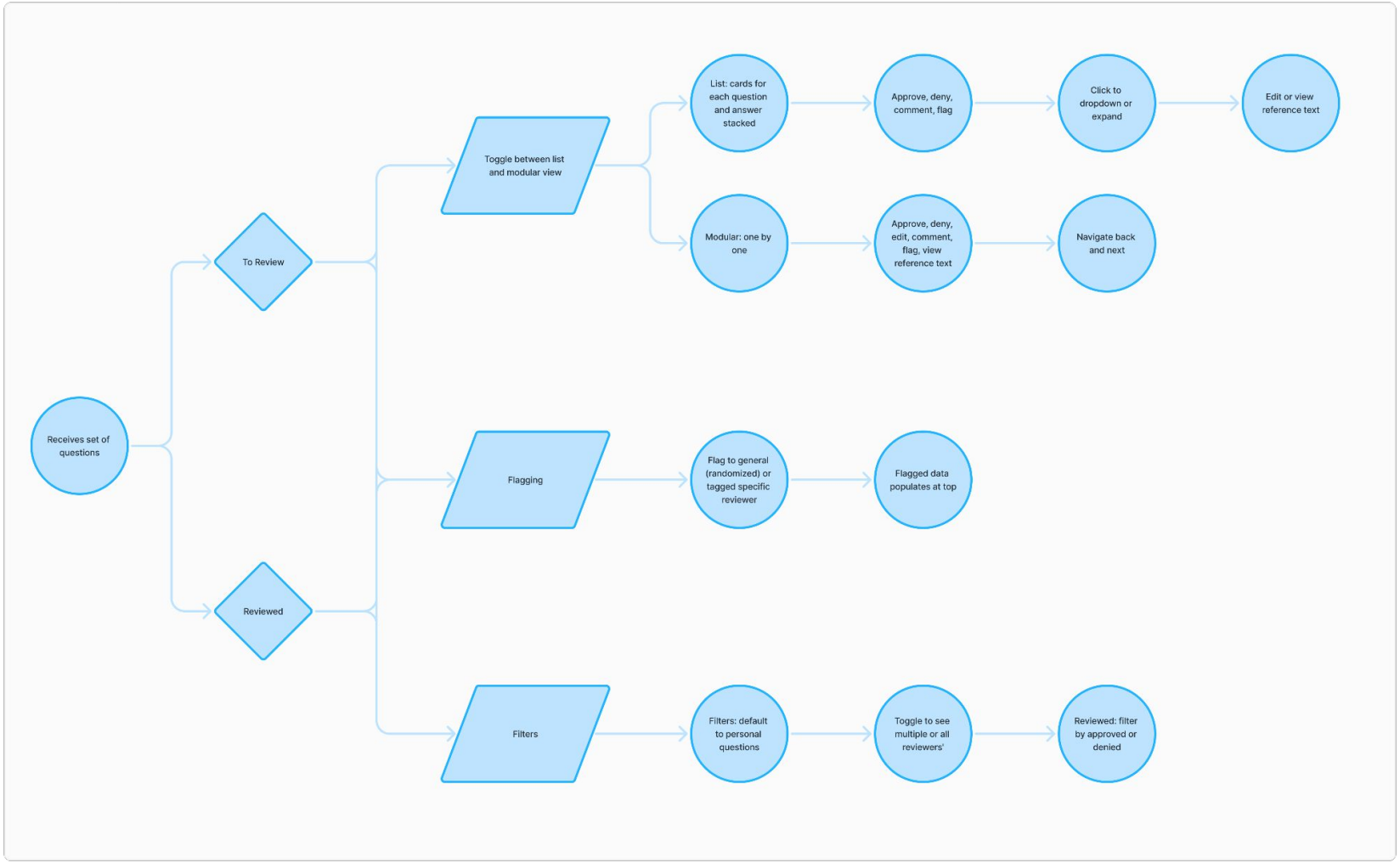
Appendix

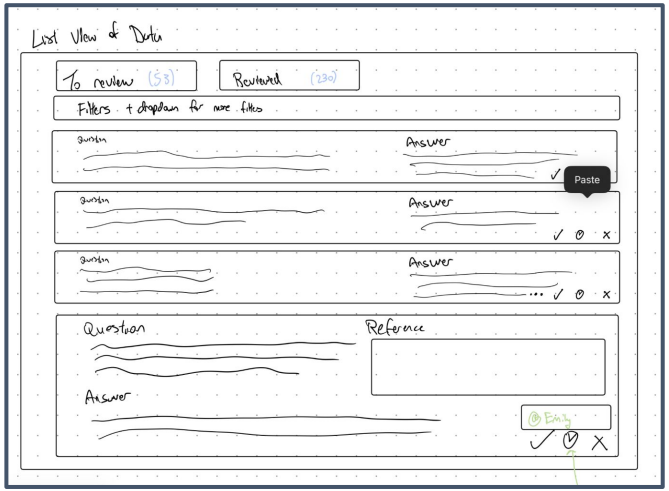
Additional Info

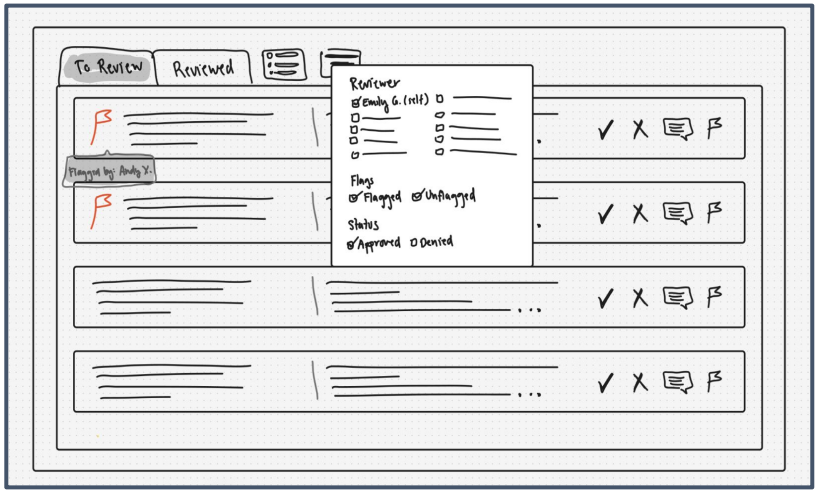
Initial Prototype Design

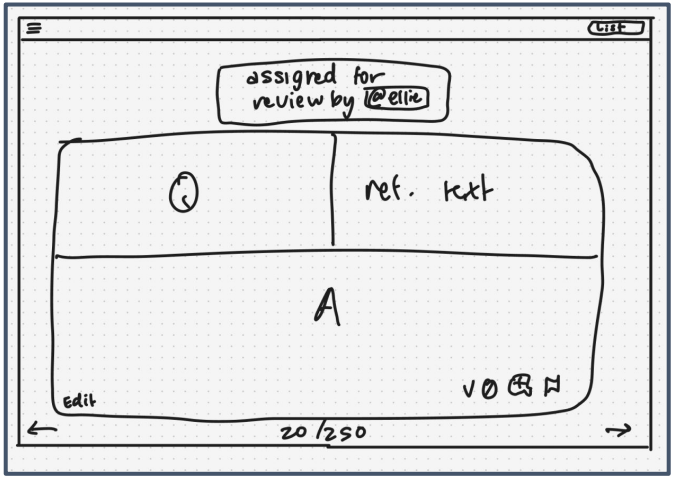
Bringing the first iteration of the tool to life

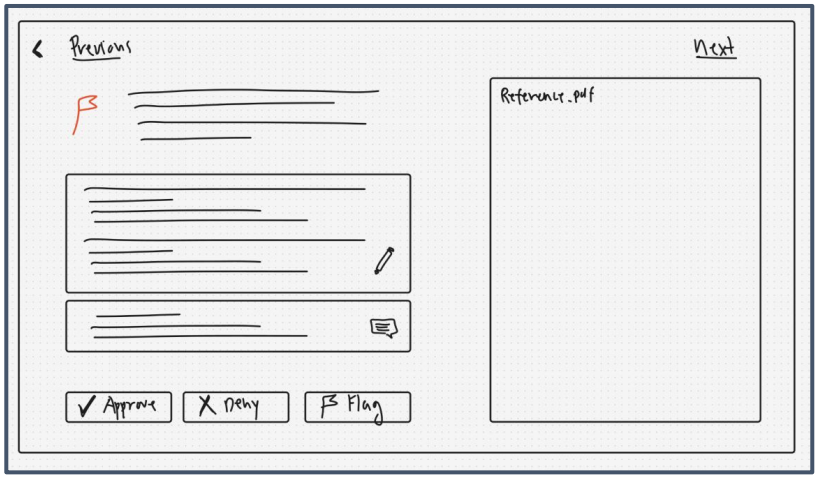
Individual Review Flow

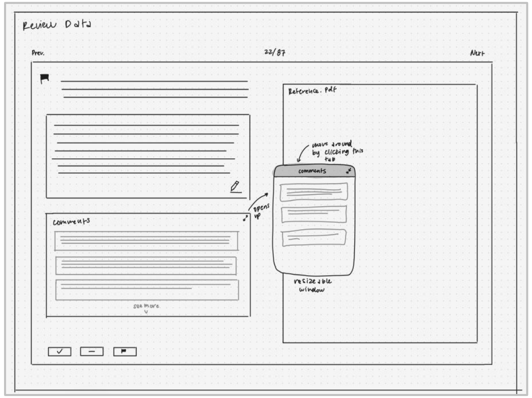


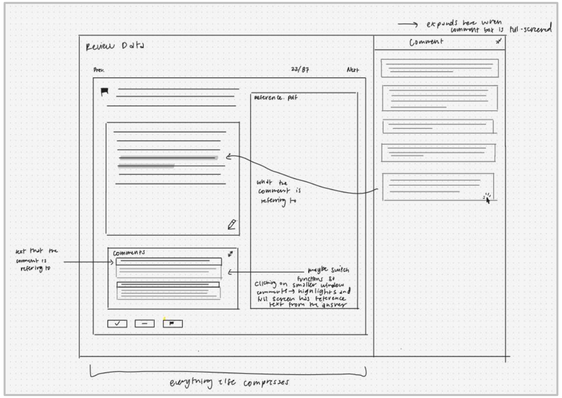

List View of Data
To review (53)
Reviewed (230)
Filters + dropdown for more filters
Question
Answer
Paste
Question
Answer
Question
Answer
Question
Reference
Answer
Emily

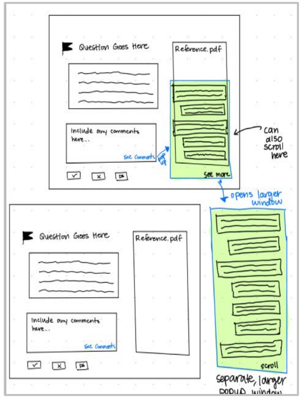

To Review
Reviewed
Reviewer
Emily G. (me)
Flagged by: Andy Y.
Flags
Flagged
Unflagged
Status
Approved
Denied

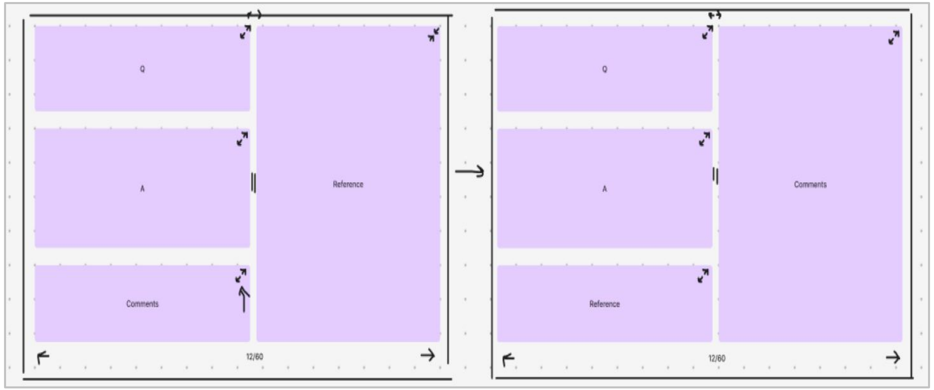

List
assigned for review by @ellie
Q
ref. text
A
Edit
V0
20/250

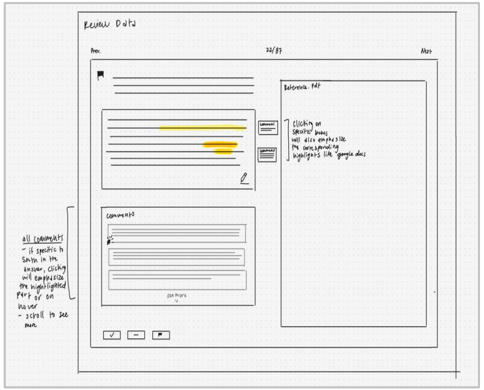

Previous
Next
Reference.pdf
Approve
Deny
Flag


Review Data
Prev.
22/87
Next
Reference. pdf
comments
comments
see more
resizable window
opens up


expands here when comment box is full-screened
Review Data
Prev.
22/87
Next
Comment
Reference. pdf
what the comment is refering to
comments
text that the comment is refering to
maybe switch functions so clicking on smaller window comments → highlights and full screen has reference text from the answer
everything else compresses


Question Goes Here
Reference.pdf
Include any comments here...
See Comments
see more
can also scroll here
opens larger window
scroll
separate, larger popup window


Q
A
Comments
Reference
12/60
Q
A
Reference
Comments
12/60


Review Data
Prev.
22/87
Next
Reference. pdf
clicking on specific boxes will also emphasize the corresponding highlights like google docs
comments
see more
all comments
- if specific to smth in the answer, clicking will emphasize the highlighted part or on hover
- scroll to see more

User Research v.1

Testing our first prototype draft with developers

What did our user testing look like?

User Testing Questions

Pre-testing questions:

- Current SDG review
- Anticipated feature uses

Testing questions:

- Users + feature interactions
- Usability of primary features
    - accept/deny, edit, comment, etc.

Post-testing questions:

- Feedback for future iterations
- Secondary functions
    - (assigning, advanced commenting)

Users

User Profile

- Software Developers at IBM
    - One developer was the main lead for project we referenced in the questions
- Time-consuming to edit questions
    - Deeply care
- Future with the product:
    - Ease of use
    - Ability to communicate with others

User-Testing Process

- Each developer → separate breakout room + review process

What were some insights/feedback we drew from user testing?

List and Modular Views

Modular view was used much more than anticipated.

- Focused on one question at a time
- More thorough + slow review process
- Difficult toggling between views

Collaborative Team Tools (filtering, commenting)

The review process was more individual than expected.

- Comment function only used when truly unsure about answer

Approving, Denying, Editing

Users would rather opt for approving, denying, or ignoring rather than editing.

- Could be attributed to the question content
- Emphasis should be more on editing question

Reliance on Reference Document

Strong reliance on reference document during review

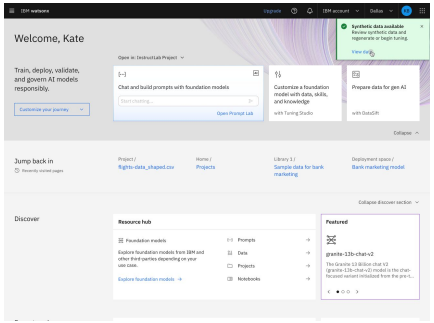
- Users struggled to find relevant information in the reference document

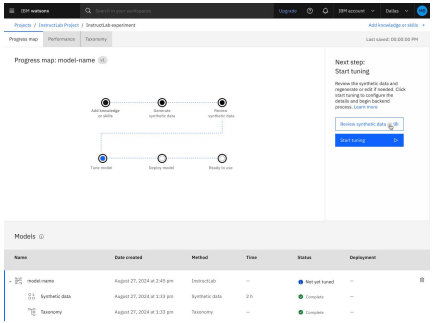
Proposed Changes

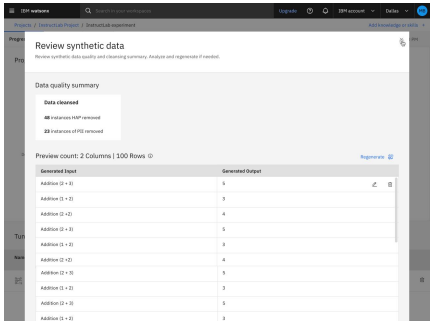
How can we use the insights gained from testing to improve our prototype?

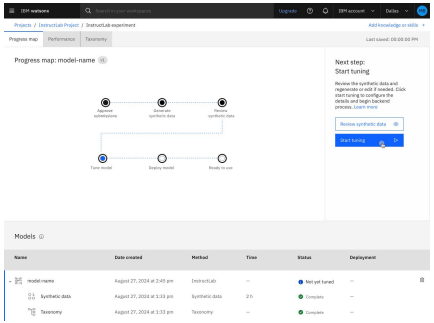
Immediate Changes

	List vs. Modular View	Commenting	PDF Reference Document
Feedback	• Users were unsure how to switch back to list from modular view • Found the icon confusing/not intuitive	• Users prioritize simplicity • Comment functions do not need to be overly complex (ex. Forums, etc.)	• Users referenced their own resource document for each question, unaware of existing placeholder • Difficult to find information
Changes	• Implemented a toggle button between list & modular views	• Added minimally invasive comment display • Updated comment icon and notifications	• Included reference doc for each question in modular view • Have a pdf search tool • Have reference document available in list view


Welcome, Kate
Train, deploy, validate, and govern AI models responsibly.
Jump back in
Discover


Progress map: model-name
Next step:
Start tuning
Models


Review synthetic data
Data quality summary
Preview count: 2 Columns | 100 Rows


Progress map: model-name
Next step:
Start tuning
Models