THE UNIVERSITY
OF TEXAS AT DALLAS

App Review Insights: A Heuristic Mining Approach

School of Economic, Political and Policy Sciences

EPPS 6323 Knowledge Mining

Oliver Jack Myers

Instructor: Dr. Karl Ho

May 12th, 2025

Table of Contents

1. Introduction.. 1

2. Methods.. 3

2.1 Apple App Store Web-Scraping.. 3

2.2 Heuristic sorting analysis of user comments.. 6

3. Shiny App Implementation..9

4. Conclusion...16

4.2 Challenged Encountered.. 17

4.3 Future Steps.. 19

4.4 Important project links...19

References... 20

AI Appendix..22

1. Introduction

Understanding how users perceive an application and identifying usability challenges is essential for the ongoing success of any business. While comprehensive User Experience (UX) research and organized usability tests supply valuable insights, they can sometimes be time-consuming and expensive. On the flip side, app reviews that are publicly accessible provide a simple and budget-friendly alternative, delivering direct insights into user thoughts, preferences, needs, and common concerns. By looking at these reviews across various app updates, development teams can gauge whether recent changes successfully addressed previous user complaints. To maintain a competitive edge, apps need to continuously grow, often by incorporating features that improve task efficiency or boost user satisfaction. However, psychological studies reveal that a single negative review might echo the feelings of many users who faced similar issues but opted not to share feedback. Thus, recurring concerns in app reviews can serve as crucial early indicators of larger user discontent.

Software development teams and product managers can proactively address user issues by consistently tracking changes in sentiment, identifying ongoing pain points, and understanding evolving user expectations. By analyzing an app's release timeline and sorting user reviews according to the issues mentioned, teams can gain deeper insights into existing problems. This understanding leads to more thoughtful choices for future improvements. To support this initiative, the project aimed to develop and implement the "App Review UX Heuristic Analyzer," a Shiny web application intended to systematically extract and assess actionable insights from Apple App Store reviews, thereby helping to pinpoint usability issues through an automated heuristic assessment aligned with well-established UX principles.

A key advantage of employing usability heuristics is that there are well-recognized solutions available for addressing any violations in the field of human-computer interaction. The ten usability heuristics utilized in this study, which classify the collected user comments, are derived from foundational research by the Nielsen Norman Group (Nielsen 2024) . These heuristics provide a solid framework for leveraging keywords from user comments to tag and filter reviews in the "App Review UX Heuristic Analyzer," spotlighting frequently encountered usability challenges and their perceived seriousness based on related sentiment.

This leads us to the primary research question of the study: How effectively can automated analysis of app store reviews, categorized by recognized usability heuristics and associated sentiment, reveal common UX pain points and their variations across different app versions?

The "App Review UX Heuristic Analyzer" is pivotal in addressing this question. It enables the user to: (1) collect app metadata and user reviews from the Apple App Store; (2) preprocess and tidy the textual data; (3) break down the reviews and compare them against a set list of keywords tied to Nielsen's 10 Usability Heuristics; (4) perform sentiment analysis on the review segments to evaluate the emotional tone connected to specific heuristic references; and (5) showcase these insights through interactive tables and visualizations, allowing users to filter by app version and date. This delivers a structured analysis of the most commonly reported heuristic violations by users, shifts in these trends with new app launches, and the overall sentiment surrounding these usability concerns, providing a clear and data-driven view of the user experience.

2. Methods

2.1 Apple App Store Web-Scraping

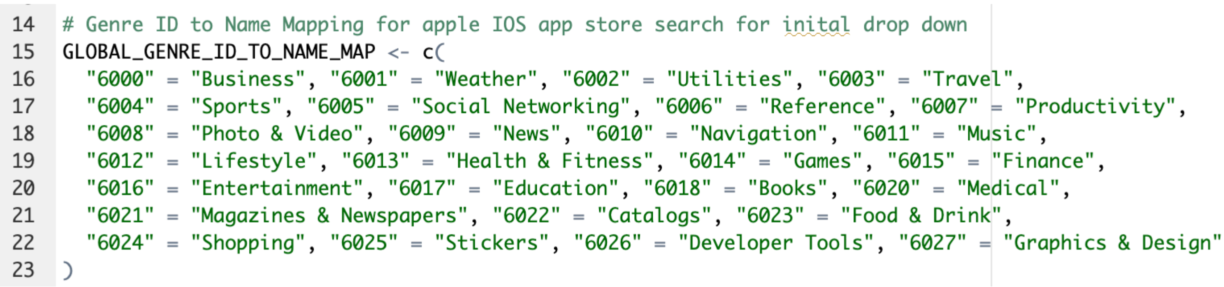
The initial challenge in this project was to devise an effective method for retrieving user reviews from the Apple App Store. Preliminary research indicated that while Apple provides APIs for developers with published applications, accessing reviews for a broader range of apps required alternative approaches. During this exploration, a Python-based framework developed by Facundo Olano (facundoolano 2025) for interacting with the App Store proved insightful. Although the original implementation was in Python, Olano's work demonstrated how the Apple RSS feed URL structure for app reviews could be leveraged.

This project drew inspiration from Olano's approach, particularly in understanding the RSS feed URL format and the types of data that could be extracted, such as version history and genre IDs. While the subsequent data processing, heuristic analysis, and the entire "App Review UX Heuristic Analyzer" application were developed independently in R for this study, Olano's publicly shared code significantly informed the foundational understanding of the review scraping mechanism. For instance, the concept of maintaining a mapping of genre names to Apple's internal genre IDs, though implemented with a custom list in this R application, was a strategy observed in Olano's framework. This acknowledgment is made to ensure due credit for the foundational insights gained.

The first stage, app identification, allows users to search for and select a target Apple iOS application. This is accomplished by interfacing with Apple's iTunes Search API. User-entered search terms (e.g., app name, keywords) and optional genre filters are used to construct a query URL. The httr package (Wickham 2023) in R is employed to send an HTTP GET request to the iTunes Search API, using the base URL https://itunes.apple.com/search. Query parameters are appended to this base URL to specify the search term (term=), country (e.g., country=US), media type (media=software), entity type (entity=software), and a limit on the number of results (e.g., limit=50). If a user selects a genre from a predefined list (e.g., "Business" corresponding to genreId=6000), this ID is also included as a query parameter (e.g., &genreId=[genre_id]). The API response is returned in JSON format, which is then parsed by the jsonlite package (Ooms 2014) into an R data frame. This data frame contains metadata for each app found, including its name (trackCensoredName), a unique track ID (trackId), primary genre (primaryGenreName), average user rating (averageUserRating), total user ratings (userRatingCount), and an icon URL (artworkUrl60). This app metadata is subsequently presented to the user in an interactive table, rendered by the DT package (Xie, Cheng, and Tan 2024), from which the user selects a single application. The trackId of the chosen app is then stored for the review scraping phase.

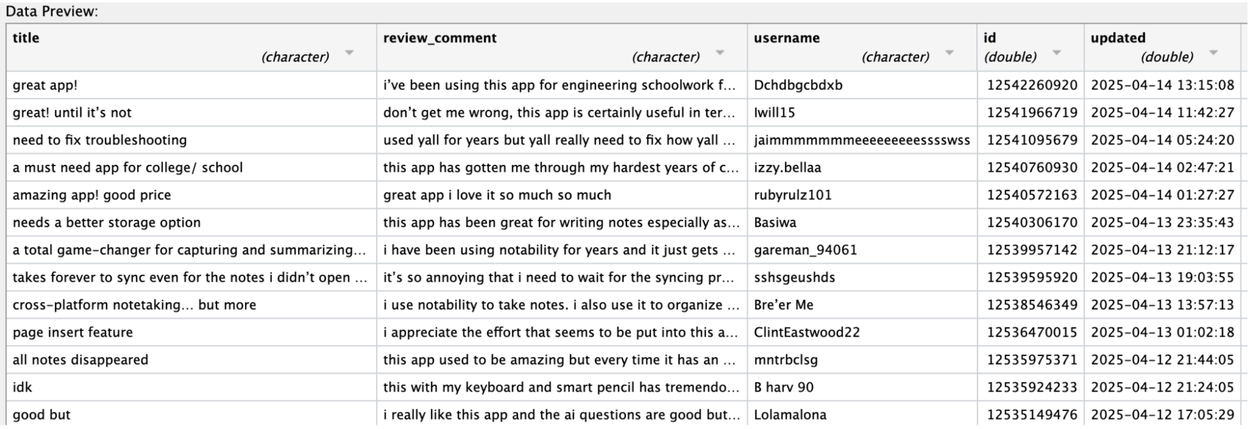
Once a target application is selected, the second stage, user review scraping, commences. The application iteratively fetches reviews by constructing URLs for specific pages of the app's customer review RSS feed. The URL structure for accessing these reviews is: https://itunes.apple.com/us/rss/customerreviews/page=[page_index]/id=[target_app_id]/sortBy=mostRecent/json. In this structure, [target_app_id] represents the trackId of the selected app, and [page_index] is an integer ranging from 1 to 10, designed to retrieve up to 500 of the most recent reviews, as each page typically contains 50 entries. Like the app search, the review feed returns data in JSON format for each page. The jsonlite package parses these responses, and core review data, such as author name, review title, review content, rating, app version associated with the review, and timestamp, are extracted from the feed$entry component of the JSON structure. Finally, data from all successfully fetched pages are combined into a single data frame using functions from the dplyr package (Wickham et al. 2023). Robust extraction logic is implemented within the application to handle potential inconsistencies in JSON field names or structure across different review entries.

(Fig.1 - Code for app Genre dropdown)



```r
# Genre ID to Name Mapping for apple IOS app store search for inital drop down
GLOBAL_GENRE_ID_TO_NAME_MAP <- c(
  "6000" = "Business", "6001" = "Weather", "6002" = "Utilities", "6003" = "Travel",
  "6004" = "Sports", "6005" = "Social Networking", "6006" = "Reference", "6007" = "Productivity",
  "6008" = "Photo & Video", "6009" = "News", "6010" = "Navigation", "6011" = "Music",
  "6012" = "Lifestyle", "6013" = "Health & Fitness", "6014" = "Games", "6015" = "Finance",
  "6016" = "Entertainment", "6017" = "Education", "6018" = "Books", "6020" = "Medical",
  "6021" = "Magazines & Newspapers", "6022" = "Catalogs", "6023" = "Food & Drink",
  "6024" = "Shopping", "6025" = "Stickers", "6026" = "Developer Tools", "6027" = "Graphics & Design"
)
```

(Fig.2 - Cleaned up comment data CSV example)



Data Preview:

title (character)	review_comment (character)	username (character)	id (double)	updated (double)
great app!	i’ve been using this app for engineering schoolwork f…	Dchdbgcbdxb	12542260920	2025-04-14 13:15:08
great! until it’s not	don’t get me wrong, this app is certainly useful in ter…	Iwill15	12541966719	2025-04-14 11:42:27
need to fix troubleshooting	used yall for years but yall really need to fix how yall …	jaimmmmmmmeeeeeeeeessssswss	12541095679	2025-04-14 05:24:20
a must need app for college/ school	this app has gotten me through my hardest years of c…	izzy.bellaa	12540760930	2025-04-14 02:47:21
amazing app! good price	great app i love it so much so much	rubyrulz101	12540572163	2025-04-14 01:27:27
needs a better storage option	this app has been great for writing notes especially as…	Basiwa	12540306170	2025-04-13 23:35:43
a total game-changer for capturing and summarizing…	i have been using notability for years and it just gets …	gareman_94061	12539957142	2025-04-13 21:12:17
takes forever to sync even for the notes i didn’t open …	it’s so annoying that i need to wait for the syncing pr…	sshsgeushds	12539595920	2025-04-13 19:03:55
cross-platform notetaking… but more	i use notability to take notes. i also use it to organize …	Bre’er Me	12538546349	2025-04-13 13:57:13
page insert feature	i appreciate the effort that seems to be put into this a…	ClintEastwood22	12536470015	2025-04-13 01:02:18
all notes disappeared	this app used to be amazing but every time it has an …	mntrbclsg	12535975371	2025-04-12 21:44:05
idk	this with my keyboard and smart pencil has tremendo…	B harv 90	12535924233	2025-04-12 21:24:05
good but	i really like this app and the ai questions are good but…	Lolamalona	12535149476	2025-04-12 17:05:29

2.2 Heuristic sorting analysis of user comments

To process user comments, I created a friendly automated system that connects segments of reviews with specific usability heuristics. This classification relies on Nielsen's 10 Usability Heuristics (Nielsen 2024), a well-respected framework in interaction design. Each heuristic comes with its own set of keywords and phrases; for example, some terms for "Visibility of System Status" include "loading," "sync," and "slow," while "Error Prevention" features terms like "bug," "crash," and "doesn't work."

This comprehensive list, consistently updated in the application, showcases our expertise by translating user-friendly language into well-established HCI principles.

The analysis begins with breaking down the cleaned full review text into individual sentences using the tidytext package (Silge and Robinson 2016) and its unnest_tokens function with the "sentences" tokenizer. Each sentence is then converted to lowercase for effortless matching. The stringr package (Wickham 2023) helps us spot predefined heuristic keywords in each sentence. A sentence can connect to multiple heuristics if it contains keywords from various categories. If any sentences lack predefined keywords, I initially classify them as "Other," making sure I account for all review content while keeping our main analysis focused on identifiable heuristic violations. This thoughtful keyword-based tagging allows us to transform qualitative user feedback into a semi-quantitative framework that aligns with established UX best practices.

For every sentence linked to a specific heuristic, I calculate an overall_sentiment_score. This score results from adding up numerical values assigned to sentiment-rich words in the sentence (for instance, -1 for each "negative" word and +1 for each "positive" word). This score gives us a numeric representation of the emotional tone connected to comments relevant to a specific heuristic. From this score, I assign an issue_severity_level (like "High," "Medium," or "Low") to each sentence tagged with a heuristic, which enriches our quantitative sentiment score with a qualitative perspective. So, if a sentence has a strongly negative sentiment score (e.g.,

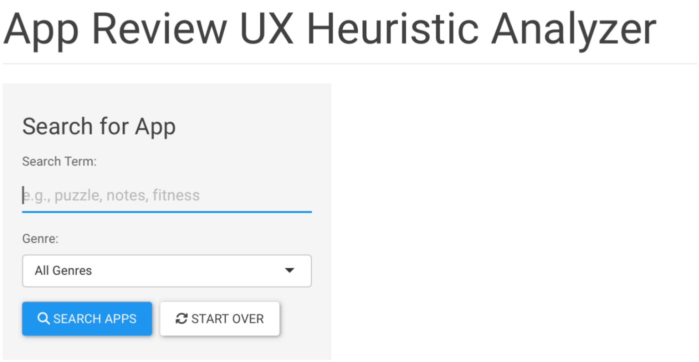
<= -3) and includes a keyword related to "Error Prevention," we'd classify it as a high-severity issue within that heuristic category. I extensively utilize the dplyr package (Wickham et al., 2023) for these grouping, joining, and summarization tasks.

By merging heuristic categorization and sentiment analysis, our "App Review UX Heuristic Analyzer" not only identifies the types of usability issues users discuss but also measures their emotional intensity, providing us with richer, more actionable insights into the user experience.

3. Shiny App Implementation

The "App Review UX Heuristic Analyzer" is an interactive web application built with the R Shiny framework. It thoughtfully guides users through an engaging workflow, starting from app selection and leading to the insightful heuristic analysis of user reviews. In the upcoming subsections, I will explore the key user interface (UI) components and the functionalities behind them.

(Fig.3 - Search for app user interface on shiny)



App Review UX Heuristic Analyzer

Search for App

Search Term:

e.g., puzzle, notes, fitness

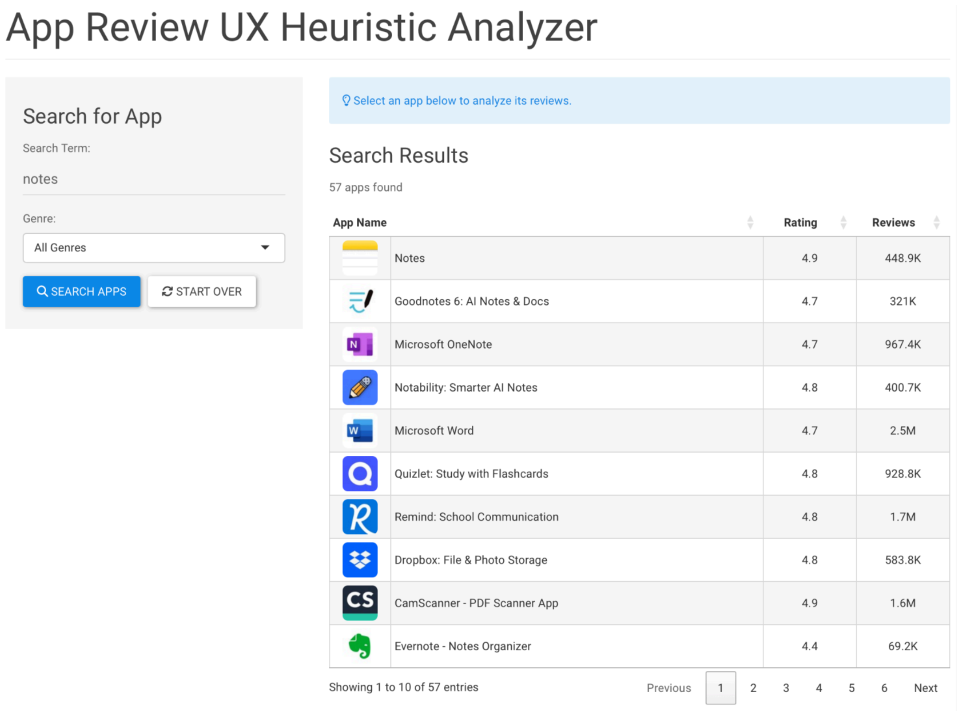
Genre:

All Genres

In this initial view, users can easily enter a search term like "notes" or "fitness tracker" to find a target iOS application. This search feature taps into Apple's iTunes Search API, mimicking how the official App Store operates. The application builds a dynamic URL request using the base API endpoint (https://itunes.apple.com/search) and adds in the user's search terms along with helpful parameters like country (country=US) and media type (media=software). Users can also make their search even better by choosing a specific app genre from a dropdown menu. This menu showcases user-friendly genre names like "Business" or "Games," which are cleverly matched to the appropriate Apple genre IDs (for instance, genreId=6000 corresponds to "Business"). This genre ID is then included in the API request to refine the search results.

When the API responds, I get a JSON object packed with a list of matching applications, which is processed using the jsonlite package (Ooms 2014). I then display the important details for each app, including its icon (pulled from a URL), name, average user rating, and total number of reviews in an engaging interactive table made possible by the DT package (Xie, Cheng, and Tan 2024), just like you see in Figure 4.

(Fig.4 - Search Results for user searching keyword "notes")



App Review UX Heuristic Analyzer

Search for App

Search Term: notes

Genre: All Genres

Select an app below to analyze its reviews.

Search Results

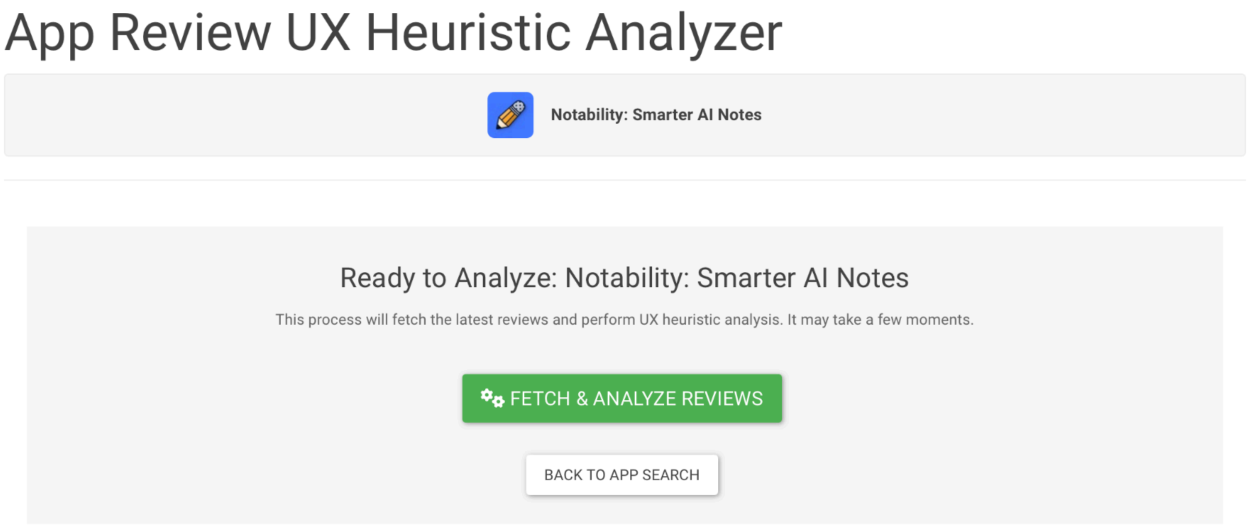
57 apps found

App Name	Rating	Reviews
Notes	4.9	448.9K
Goodnotes 6: AI Notes & Docs	4.7	321K
Microsoft OneNote	4.7	967.4K
Notability: Smarter AI Notes	4.8	400.7K
Microsoft Word	4.7	2.5M
Quizlet: Study with Flashcards	4.8	928.8K
Remind: School Communication	4.8	1.7M
Dropbox: File & Photo Storage	4.8	583.8K
CamScanner - PDF Scanner App	4.9	1.6M
Evernote - Notes Organizer	4.4	69.2K

Showing 1 to 10 of 57 entries

Whenever you select an app from the results table, the application securely captures the app's unique trackId for you. You're then greeted with a confirmation screen (Figure 5) showcasing the name of the chosen app (like "Notability") along with its icon, giving you a chance to double-check your selection before diving into the review analysis phase.

(Fig.5 - app selection confirmation screen for “notability”)



App Review UX Heuristic Analyzer

Notability: Smarter AI Notes

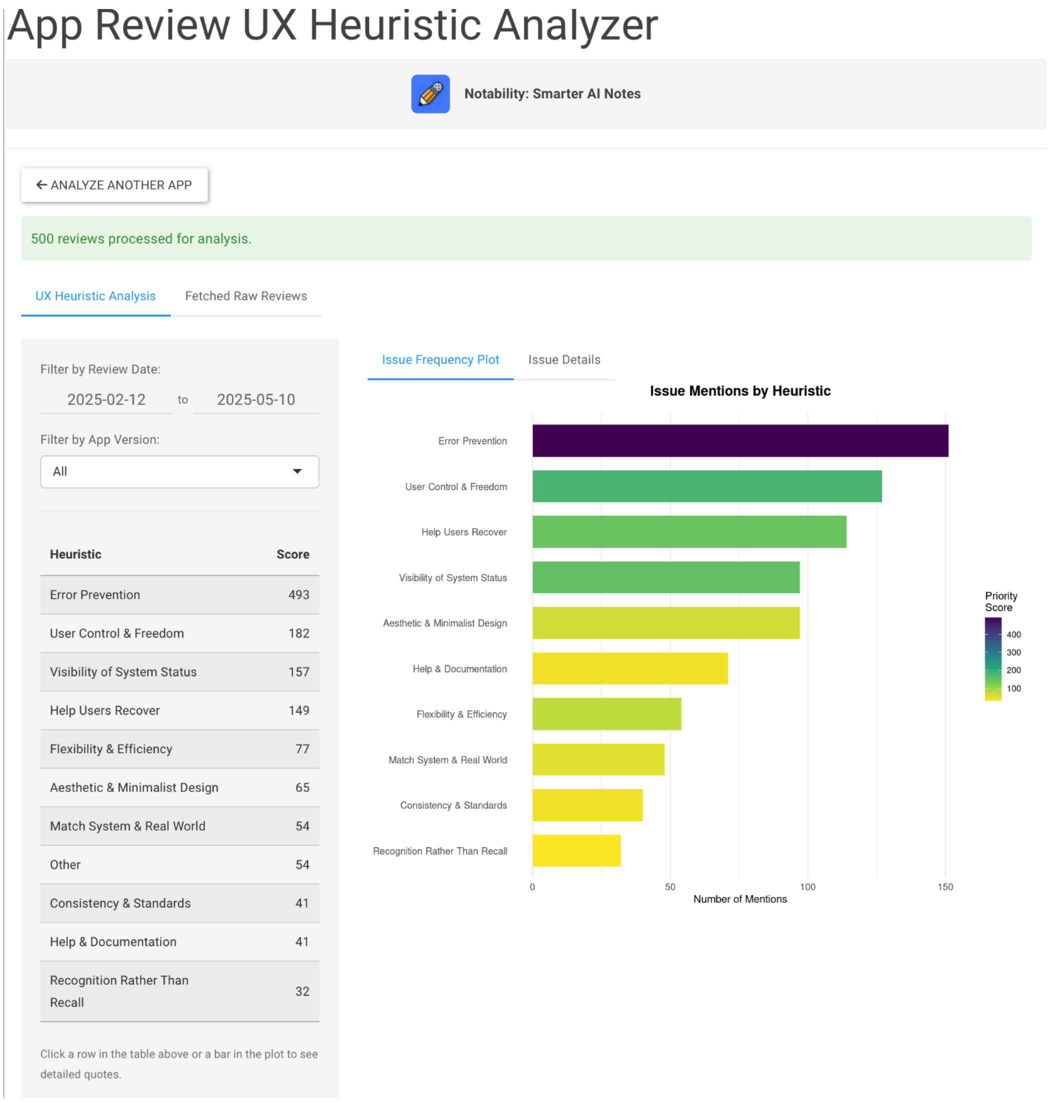
Ready to Analyze: Notability: Smarter AI Notes

This process will fetch the latest reviews and perform UX heuristic analysis. It may take a few moments.

When a user selects their app and clicks the "Fetch & Analyze Reviews" button, the application starts gathering user reviews right away. It uses the chosen app's trackId to create URLs for Apple's customer review RSS feed. This application efficiently works through this feed, typically collecting 50 reviews per page and going through up to 10 pages, aiming to gather approximately 500 of the latest user comments. The httr package (Wickham 2023) takes care of these HTTP requests. To prevent any potential

IP blocking or errors from sending too many requests too quickly to the RSS feed, a short delay (Sys.sleep()) is thoughtfully added between page fetches. The raw review data comes back in JSON format and is then parsed, cleaned up, and organized into a data frame, making it ready for the next round of heuristic analysis. This process includes pulling out review text, ratings, timestamps, and, of course, the app version associated with each review.

(Fig.6 - Heuristic Analysis Data for "notability")



App Review UX Heuristic Analyzer

Notability: Smarter AI Notes

500 reviews processed for analysis.

Filter by Review Date: 2025-02-12 to 2025-05-10

Filter by App Version: All

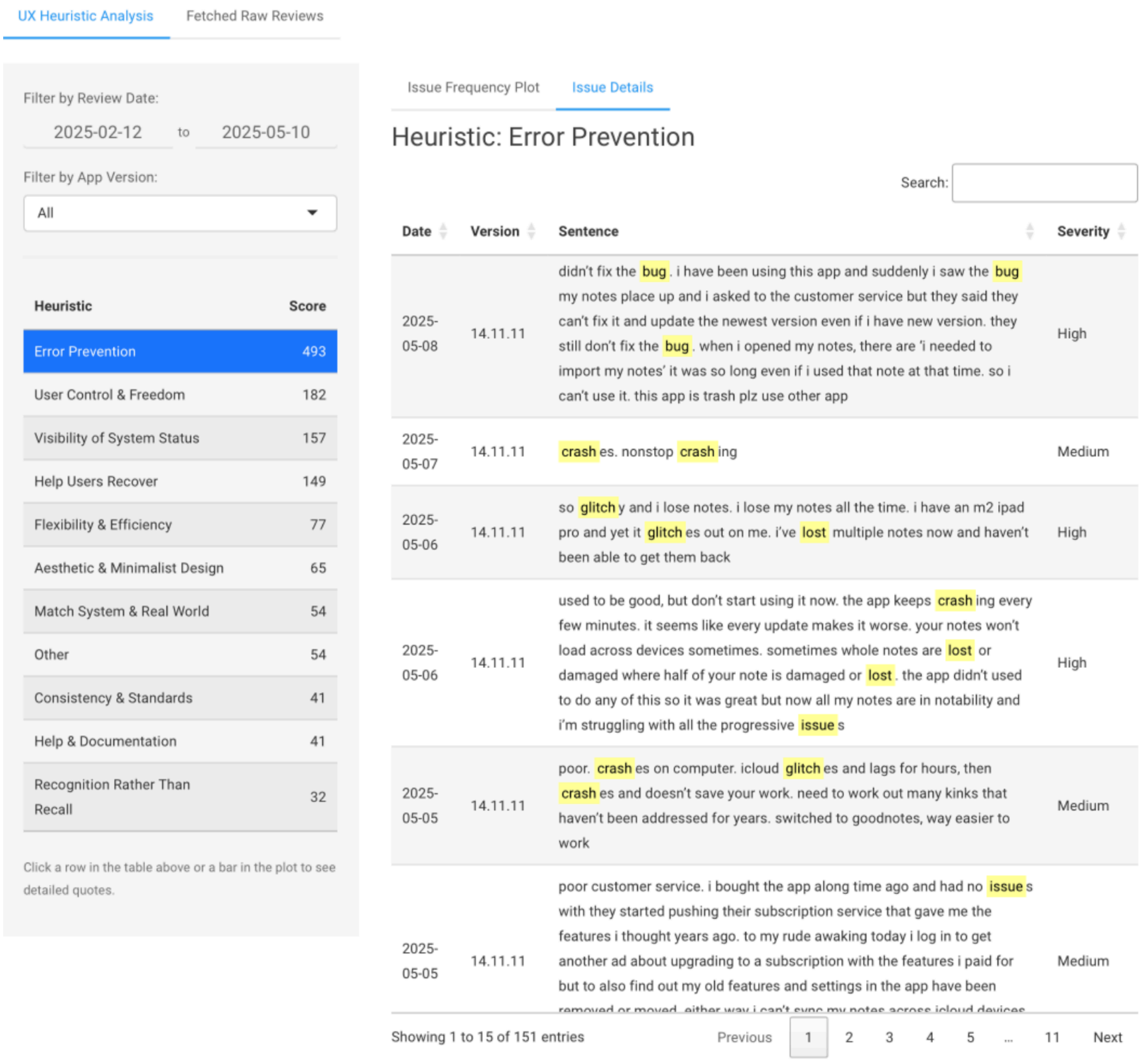
Heuristic	Score
Error Prevention	493
User Control & Freedom	182
Visibility of System Status	157
Help Users Recover	149
Flexibility & Efficiency	77
Aesthetic & Minimalist Design	65
Match System & Real World	54
Other	54
Consistency & Standards	41
Help & Documentation	41
Recognition Rather Than Recall	32

Click a row in the table above or a bar in the plot to see detailed quotes.

This dashboard provides a detailed look at the user comments analyzed. One standout feature is the "Issue Mentions by Heuristic" bar chart, which showcases the frequency of comments related to each of Nielsen's 10 Usability Heuristics (Nielsen 2024). The height of each bar indicates how many unique sentences have been associated with that particular heuristic. Plus, the bars are color-coded based on a PriorityScore, which is calculated from the overall negative sentiment of comments within that heuristic category. A higher (more negative) sentiment score is reflected by a deeper color, signaling that there's a greater priority to tackle issues tied to that heuristic.

Beside the chart, you'll find a summary table that displays each heuristic category alongside its corresponding PriorityScore, arranged so that the most critical heuristics are highlighted first. When users engage with this table by selecting a row or interact with the bar chart by clicking on a bar, the "Issue Details" view will update on the fly. This view, as shown in Figure 7, showcases the specific user sentences (quotes) connected to the selected heuristic. To make it easier to understand, the keywords in these sentences that relate to the heuristic are highlighted using <mark> HTML tags.

(Fig.7 - Issues Detailed User comments highlighted for given Heuristic)



Filter by Review Date: 2025-02-12 to 2025-05-10

Filter by App Version: All

Heuristic	Score
Error Prevention	493
User Control & Freedom	182
Visibility of System Status	157
Help Users Recover	149
Flexibility & Efficiency	77
Aesthetic & Minimalist Design	65
Match System & Real World	54
Other	54
Consistency & Standards	41
Help & Documentation	41
Recognition Rather Than Recall	32

Click a row in the table above or a bar in the plot to see detailed quotes.

Heuristic: Error Prevention

Search:

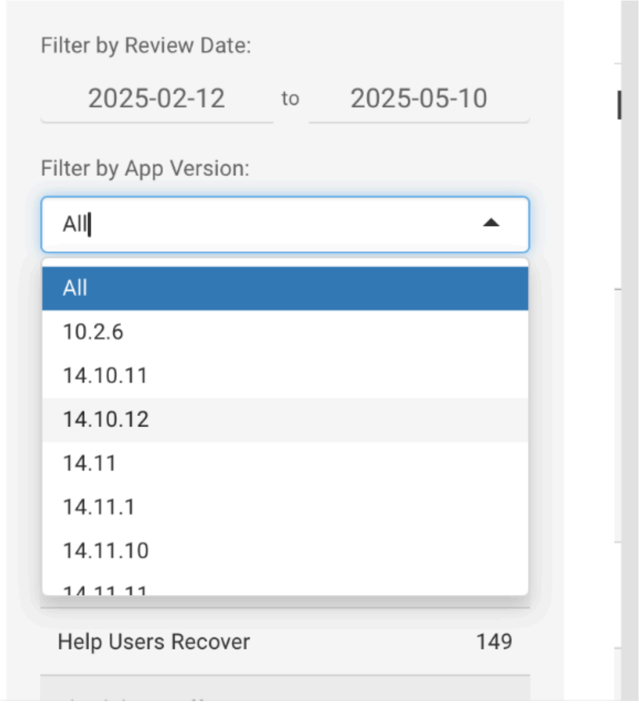
Date	Version	Sentence	Severity
2025-05-08	14.11.11	didn't fix the bug. i have been using this app and suddenly i saw the bug my notes place up and i asked to the customer service but they said they can't fix it and update the newest version even if i have new version. they still don't fix the bug. when i opened my notes, there are 'i needed to import my notes' it was so long even if i used that note at that time. so i can't use it. this app is trash plz use other app	High
2025-05-07	14.11.11	crashes. nonstop crashing	Medium
2025-05-06	14.11.11	so glitchy and i lose notes. i lose my notes all the time. i have an m2 ipad pro and yet it glitches out on me. i've lost multiple notes now and haven't been able to get them back	High
2025-05-06	14.11.11	used to be good, but don't start using it now. the app keeps crashing every few minutes. it seems like every update makes it worse. your notes won't load across devices sometimes. sometimes whole notes are lost or damaged where half of your note is damaged or lost. the app didn't used to do any of this so it was great but now all my notes are in notability and i'm struggling with all the progressive issues	High
2025-05-05	14.11.11	poor. crashes on computer. icloud glitches and lags for hours, then crashes and doesn't save your work. need to work out many kinks that haven't been addressed for years. switched to goodnotes, way easier to work	Medium
2025-05-05	14.11.11	poor customer service. i bought the app along time ago and had no issues with they started pushing their subscription service that gave me the features i thought years ago. to my rude awaking today i log in to get another ad about upgrading to a subscription with the features i paid for but to also find out my old features and settings in the app have been removed or moved. either way i can't sync my notes across icloud devices	Medium

Showing 1 to 15 of 151 entries

This interactive tool makes it easy for users, including software developers and UX researchers, to quickly identify which usability principles are commonly overlooked and to explore the specific user feedback associated with these concerns. For instance, as illustrated in Figure 7, by selecting the "Error Prevention" heuristic, you can find comments like "crashed nonstop," providing clear insight into particular issues that require attention.

A particularly useful feature for improving iterative development and tracking issues is the ability to filter the heuristic analysis by app version. The application automatically collects all unique app versions mentioned in the reviews and creates a friendly dropdown menu (Figure 8) featuring these versions.

(Fig.8 - Versions dropdown)



Filter by Review Date:

2025-02-12 to 2025-05-10

Filter by App Version:

All

Help Users Recover 149

When users choose a specific version, they can delve deeply into reviews that are directly related to that release. This feature is crucial for keeping track of how usability issues evolve over time. For example, a development team can look at whether the various concerns regarding "Error Prevention" in version 1.0, which had a high

PriorityScore, were successfully resolved in version 1.1. They can do this by seeing a reduction in those complaints and noticing a lower PriorityScore for that metric in the reviews of the newer version. This method makes it easier to assess how design updates and bug fixes impact user experience based on data.

4. Conclusion

4.1 Discussion

Developing this app involved many twists and turns throughout the semester; however, I gained significant insights not only in creating a lightweight data-scraping tool but also in delving into the details necessary for the app's functionality. Initially, I aimed to analyze user sentiment through reviews on the Apple App Store to predict future sentiment shifts when the app's UI undergoes changes. Yet, this approach proved challenging since users update their apps at varying times. Moreover, when analyzing the data to identify common words, I found no consistent pattern of negative sentiment following an app update, which was contrary to my expectations. I envision this app as a means to extract knowledge from data, offering insights into user comments. As a user experience researcher, this type of data is critical for me and other software developers. My unique contribution to this project involved applying my domain expertise using the Nielsen Norman 10 heuristics and creating categories to filter user comments by heuristic issues. This will provide developers or researchers with clearer insights into issues and their frequency, making the data easier to analyze and enabling the tracking of heuristic problems over time by linking versions of the app to user comments sourced from Apple's RSS feed. Although I would like to incorporate Android data in the future, it was too ambitious for this project. I invested considerable time and learned a great deal over the semester. This tool will be further iterated and improved. I believe my approach, which integrates heuristics and version control, is unique and hasn't been previously documented online. This is beneficial because users can search for an app just like they would in the App Store without requiring a large database, as all the data is pulled directly via the RSS feed. I also plan to add functionalities for downloading CSV data to assist users in scraping, analyzing, and compiling the data for their purposes. The goal of this project was to derive knowledge and insights from user comments using heuristic analysis, and I believe it has achieved that and more.

4.2 Challenges Encountered

- The main issue I struggled with was the shiny app not being able to hold user information, such as when they added their search term into the search bar at the start of the app. Once they pressed search apps, their text input would disappear, forcing them to re-enter it if they wanted to change the category they were searching by. I used AI to help me overcome this by adding reactive values to make dynamic changes to the shiny app values while the app was in process, significantly improving the user experience and functionality of my app.

- Another issue I encountered was related to getting the BERT sentiment analysis to work properly. I continuously had problems running it. I believe it could be due to my having a Mac, but I am not sure. I tried to get it to work countless times, and eventually, I decided to part ways with it and take a more low-level approach by adding my own categories for the heuristics.
- The categories of the heuristics were an issue because I had to look through many comments and pick out words that were common in certain types of comments I viewed to fit into the heuristic category. This worked for the purpose of this project, and I believe it did derive value in my app, but I am aware of the limitations that this poses. I would like to address this by taking a more predictive approach and adding in topic modeling.
- Another issue is looking at versions over time because some people do not update their app at the same time, so some will comment on issues they are experiencing months after others have updated their app. Thus, trying to show trends graphically was changing due to many overlapping time periods, but I would like to figure out a way to overcome this in the future to improve the app.
- One of the challenges I encountered was the limitations of using the Apple iOS RSS feed. While it’s a free option, popular apps can quickly hit the 500 comment limit. This RSS feed allows us to gather data over just a week, which doesn't provide much insight for the long term. I envision a more expansive solution where users could select an app to collect data over time. Depending on the app's popularity, it could scrape data on an automated schedule and send this

information to a database, enabling us to analyze trends for thematic and sentiment analysis over time. However, implementing this would require setting up a database and adding additional code and functionality to support these features. I'm excited about the possibility of exploring this in the future!

4.3 Future Steps

Looking ahead, I'm excited to enhance the way I assign heuristic categories to various user comments. By training a topic model using thousands of labeled comments, I aim for it to predict the correct category for each user comment. This approach is much more effective than simply relying on keywords, which often miss the nuances of language. With more accurate predictions, I can ensure that our model covers a broader range of comments across different apps. I also want this app to excel in formal sentiment analysis and to gather insights on which features users wish to see. This way, developers can easily view user requests alongside heuristic issues. Such a tool would be incredibly valuable for comparing multiple apps or assessing the competition, helping to ensure your app stands out and meets user expectations. Additionally, I'd love to gather even more data, including user comments from social media or app review sites like G2 and Reddit, to widen our insights further set.

4.4 Important project links

Live APP Link: https://oliverjackmyers.shinyapps.io/App_Sentiment_Project/

Full Code Link:

https://oliverjackmyers.github.io/pages/EPPS_6323/projectproposal/EPPS_6323_projectproposal.html

References

1. Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2024. Shiny: Web Application Framework for R. R package version 1.10.0. https://CRAN.R-project.org/package=shiny.

2. Chang, Winston. 2021. Shinythemes: Themes for Shiny. R package version 1.2.0. https://CRAN.R-project.org/package=shinythemes.

3. Grolemund, Garrett, and Hadley Wickham. 2011. "Dates and Times Made Easy with lubridate." Journal of Statistical Software 40(3): 1–25. https://www.jstatsoft.org/v40/i03/.

4. facundoolano. 2025. "app-store-scraper: Scrape Data from the iTunes App Store." GitHub repository. Accessed April 10th, 2025. https://github.com/facundoolano/app-store-scraper.

5. Ooms, Jeroen. 2014. "The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects." arXiv:1403.2805 [stat.CO]. https://arxiv.org/abs/1403.2805.

6. Nielsen, Jakob. 2024. "10 Usability Heuristics for User Interface Design." Nielsen Norman Group. January 30. https://www.nngroup.com/articles/ten-usability-heuristics/

7. Silge, Julia, and David Robinson. 2016. "Tidytext: Text Mining and Analysis Using Tidy Data Principles in R." Journal of Open Source Software 1(3). https://doi.org/10.21105/joss.00037.

8. Wickham, Hadley. 2016. ggplot2: Elegant Graphics for Data Analysis. New York: Springer-Verlag.

9. Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. dplyr: A Grammar of Data Manipulation. R package version 1.1.4. https://CRAN.R-project.org/package=dplyr.

10. Wickham, Hadley. 2023. httr: Tools for Working with URLs and HTTP. R package version 1.4.7. https://CRAN.R-project.org/package=httr.

11. Wickham, Hadley. 2023. stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.5.1. https://CRAN.R-project.org/package=stringr.

12. Wickham, Hadley, and Lionel Henry. 2025. purrr: Functional Programming Tools. R package version 1.0.4. https://CRAN.R-project.org/package=purrr.

13. Wickham, Hadley, Davis Vaughan, and Maximilian Girlich. 2024. tidyr: Tidy Messy Data. R package version 1.3.1. https://CRAN.R-project.org/package=tidyr.

14. Xie, Yihui, Joe Cheng, and Xianying Tan. 2024. DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.33. https://CRAN.R-project.org/package=DT.

I. AI Appendix

A. Exchange Highlights

1. **Debugging Reactive Value Updates and UI Synchronization.**

    a. I wanted to make sure that when an app is chosen from the search results, its trackId is accurately saved in app_data$selected_app_id, and that the UI transitions seamlessly to the "analysis_setup_prompt" view.

    b. **Example AI Prompt:** *“In my R Shiny app, I have a reactiveValues object called 'app_data' with a field 'selected_app_id'. When a user selects a row in a DT table (output$app_search_results_table), I want to update app_data$selected_app_id with the 'AppTrackId' from the selected row of 'app_data$search_results_df'. I also need to change app_data$current_ui_view to 'analysis_setup_prompt'. My observeEvent for table row selection isn't consistently updating the reactiveValues or triggering the UI change. Can you show me the correct structure for this observeEvent, including how to access the selected row's data and update reactiveValues?”*

    c. Google AI Studio Response Example:

```
“observeEvent(input$app_search_results_table_rows_selected, {
  req(input$app_search_results_table_rows_selected) # Ensure a row is selected
  selected_row_idx <- input$app_search_results_table_rows_selected
  # Ensure search_results_df is not NULL and index is valid
  req(app_data$search_results_df,
    length(selected_row_idx) > 0,
    selected_row_idx <= nrow(app_data$search_results_df))
  chosen_app_details <- app_data$search_results_df[selected_row_idx, ]
  app_data$selected_app_id <- chosen_app_details$AppTrackId
  app_data$selected_app_name <- chosen_app_details$OriginalAppName
  # ... (other app details) ...
  app_data$current_ui_view <- "analysis_setup_prompt"
})”
```

2. Implementing Interactive Heuristic Detail Display and Keyword Highlighting.

a. When a user clicks on a heuristic in the summary table or the frequency plot, the "Issue Details" tab should display relevant quotes, and keywords specific to that heuristic should be highlighted within those quotes.

b. **Example AI Prompt:** "I have a Shiny app where users select a heuristic (e.g., 'Error Prevention') stored in 'analysis_state$active_heuristic_name'. I have a data frame 'filtered_ux_analysis_data()' with a 'sentence_text' column and a 'heuristic_category' column. I also have 'GLOBAL_HEURISTIC_KEYWORDS_DF' which maps heuristics to their keywords. How can I: 1. Filter 'filtered_ux_analysis_data()' to get sentences for the 'active_heuristic_name'? 2. For these sentences, iterate through the keywords associated with the 'active_heuristic_name' (from GLOBAL_HEURISTIC_KEYWORDS_DF) and wrap each keyword found in the 'sentence_text' with <mark> HTML tags for highlighting? 3. Display this in a DT::datatable, ensuring the HTML tags are rendered. I'm thinking of using purrr::reduce for the highlighting but am unsure of the exact implementation."

c. ***Google AI Studio Response Example:***

```
# AI might suggest:
output$heuristic_quotes_display_table <- renderDT({
  active_heuristic <- analysis_state$active_heuristic_name
  req(active_heuristic, filtered_ux_analysis_data()) # Ensure prerequisites
  quotes_data_source <- filtered_ux_analysis_data()
  keywords_to_highlight <- GLOBAL_HEURISTIC_KEYWORDS_DF %>%
    filter(heuristic == active_heuristic) %>%
    pull(keyword)
  quotes_for_selected_heuristic <- quotes_data_source %>%
    filter(heuristic_category == active_heuristic) %>%
    # ... select relevant columns ...
    mutate(
      HighlightedSentence = purrr::reduce(keywords_to_highlight,
        function(text, kw) {
          stringr::str_replace_all(text,
                                   fixed(kw, ignore_case = TRUE),
                                   ~ paste0("<mark>", .x, "</mark>"))
        },
        .init = sentence_text # or original_review_text if that's the source
      )
    ) %>%
    select(Sentence = HighlightedSentence, /* other columns */
  datatable(quotes_for_selected_heuristic, escape = FALSE, ...) # escape=FALSE is crucial

})
```

B. AI Tools Used

1. **Google AI Studio (Gemini Pro developer preview):** Used for creating code snippets, developing strategies for intricate reactive logic, and troubleshooting R Shiny-related problems.
2. **ChatGPT (OpenAI, GPT-4 via paid subscription):** Utilized to enhance code explanations, propose different programming patterns, and help clearly express methodological steps for the project paper.

C. Explanation of How AI Tools Were Used:

1. **Debugging**
    a. AI identified logical errors in R code, particularly within Shiny's reactive framework (e.g., observeEvent, reactiveValues, renderUI synchronization). It addressed issues with reactive variables not updating and UI elements not re-rendering after state changes.
2. **Exploring an Approach for Scraping Adaptation**
    a. While a public resource inspired the Apple RSS feed URL structure, AI aided in adapting this Python-based concept into robust R functions. This included error handling for HTTP requests (httr), iterative page fetching with delays (Sys.sleep()), and effective parsing of nested JSON structures (jsonlite) in R.

3. **Reactive Variable Management**
    a. AI provided examples for effectively using reactiveValues to store and update application state (e.g., search results, selected app ID, current UI view), crucial for managing data flow and UI state across interactions.
4. C**omplex UI Interactivity Logic**
    a. App Selection Workflow: AI helped refine the logic for capturing user selections from a DT table, storing the app ID, and triggering a change to a confirmation screen for data scraping, ensuring data integrity and smooth UI transitions.
5. **Heuristic Detail Display & Highlighting**
    a. AI developed the logic for dynamically filtering review sentences based on selected heuristics and programmatically highlighting relevant keywords, suggesting purrr::reduce for iterative string replacement and ensuring HTML tags for highlighting were rendered correctly in DT tables (escape = FALSE).
6. **Code Refinement and Best Practices**
    a. AI tools refactored existing R code for better readability, efficiency, and adherence to best practices.

D. Account of Why AI Tools Were Used

1. **To Overcome Technical Hurdles:**
    a. Shiny's reactivity and dynamic UI creation, particularly for interactive data-driven features, posed considerable challenges. AI assisted in filling gaps in current knowledge and offered avenues for solutions.

2. **To Accelerate Development:**
    a. AI provided quick prototyping for code snippets and debugging support, greatly cutting down on the time needed for troubleshooting and refining solutions.
3. **To Explore Alternative Solutions:**
    a. When initial approaches to a problem are inefficient or overly complex, AI can suggest alternative programming patterns or R packages that are more suitable.
4. **To Implement Complex Features:**
    a. Implementing features such as dynamic keyword highlighting based on various criteria was complex when starting from scratch. However, AI offered foundational logic that could be adapted and integrated more easily.
5. **To Learn and Validate Approaches:**
    a. Engaging with AI enhanced the understanding of various R concepts and Shiny functionalities, as it frequently offered explanations along with code suggestions. This proved to be both an educational resource and a means to verify independently developed methods.
6. **To Manage Scope and Complexity:**
    a. The project comprised several interrelated elements (API interaction, web scraping, data processing, text analysis, dynamic UI). AI helped simplify these complexities, enabling more manageable solutions.