

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 15, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Lauren Xie
David Truong
Tino Theodoropoulos
Jason Vu

ENTITLED

**Developing an Open-Source Tool for Systematic App Reviews for
Non-Technical Researchers**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Kai Lukoff

Thesis Advisor

John Fagan

Department Chair

Developing an Open-Source Tool for Systematic App Reviews for Non-Technical Researchers

by

Lauren Xie
David Truong
Tino Theodoropoulos
Jason Vu

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 15, 2023

Developing an Open-Source Tool for Systematic App Reviews for Non-Technical Researchers

Lauren Xie
David Truong
Tino Theodoropoulos
Jason Vu

Department of Computer Science and Engineering
Santa Clara University
June 15, 2023

ABSTRACT

Researchers in many fields, from psychology to privacy studies, want to understand the mobile app ecosystem. The process of reviewing mobile apps is called a 'systematic app review' (SAR). In a systematic app review, researchers analyze app metadata (e.g., the description, user reviews of the app, the app's permissions list) and/or the experience of the app itself (e.g., the user interface, the content it contains). Unfortunately, this process does not yet have clearly defined best practices and the process of scraping data from app stores is costly, challenging, and time-consuming for researchers, especially ones without a technical background. Researchers lack a well-documented open-source tool that they can use to scrape mobile app stores for app data.

The goal of this project is to develop an open-source tool that non-technical researchers can use to easily scrape the world's two most popular mobile app stores: Google Play and Apple App Store. This will empower non-technical researchers to easily conduct systematic app reviews by reducing the time it takes to scrape app stores and eliminating the need to find a third-party technical partner. To do this, we will meet and understand the needs of researchers who wish to conduct systematic app reviews but find it costly to do so, and identify (a) what researchers are looking for in an open-source tool and (b) best practices for conducting systematic app reviews. We then plan to develop our own prototype of an open-source tool based on these needs. Our project will provide non-technical researchers with tools and practices to better understand the current app ecosystem and to identify ways to improve it.

Table of Contents

1	Introduction	2
1.1	The problem	2
1.2	Limitations of existing systems	2
1.3	Our approach	4
2	User Research	5
2.1	Stakeholder needs	5
2.2	User stories	5
2.3	Methodology	6
2.4	Formative User Interview Results	7
3	Design and Rationale	8
3.1	Design	8
3.2	Functional requirements	9
3.3	Non-functional requirements	9
3.4	Rationale	9
4	Technologies	10
4.1	System Components	10
4.2	Improved Components	11
5	System Evaluation	13
5.1	Internal Testing	13
5.2	External Testing	13
5.3	Testing Results	14
6	Implementation	16
6.1	Timeline	16
6.2	Agile software development	17
6.3	Project Risks	17
7	Conclusion	18
7.1	Summary	18
7.2	Lessons Learned	18
7.3	Next Steps	19
8	References	20

List of Figures

1.1	Google Play Store displaying only 30 apps using the search word 'magic'	3
1.2	facundoolano Google Play scraper's single line of installation instruction	4
2.1	Systematic App Review user stories diagram	6
3.1	Systematic App Review C4 software architecture diagram	8
4.1	App data output in spreadsheet format	11
4.2	Scraper graphical user interface	12
6.1	SAR team timeline using the agile development method	16

Acknowledgements

We would like to thank our advisor Dr. Kai Lukoff for his support, guidance, and encouragement throughout project. His guidance and advice supported us throughout all stages of this project - ideation, research, development, presenting, and review.

Chapter 1

Introduction

1.1 The problem

Researchers in many fields, from psychology to privacy studies, want to understand the mobile app ecosystem. For example, researchers in psychology might examine adherence to clinical guidelines in apps that contain the keyword “cognitive behavioral therapy” in the description. Or privacy researchers might check how many children’s mobile games use third-party trackers, which is illegal in the U.S. and EU.

This process of reviewing mobile apps is called a ‘systematic app review.’ In a systematic app review, researchers analyze app metadata (e.g., the description, user reviews of the app, the app’s permissions list) and/or the experience of the app itself (e.g., the user interface, the content it contains).

1.2 Limitations of existing systems

In general, the systematic app review process, unfortunately, does not yet have clearly defined best practices and the process of scraping data from app stores is costly, challenging, and time-consuming for researchers, especially ones without a technical background. Currently, researchers lack a well-documented open-source tool that they can use to scrape mobile app stores for app data.

A popular scraper is the facundoolano Google Play scraper, which is an open-source scraper using Node.js to scrape app data on the Google Play store. This scraper has two thousand stars on GitHub (similar to likes on social media, two thousand people like this scraper) and has been used and cited in at least 10 different studies. Downloadable from GitHub, this program allows the user to implement different functions (list, suggest, search, etc.) that retrieve data from the given parameters outlined in the README.md. While this scraper successfully gathers data from the Google Play store, it does have limitations that hinder its performance for non-technical researchers. For one, the Google Play scraper, without modifications, can only scrape up to 30 apps, as seen in Figure 1.1. In Figure 1.1, we see the lack of a next-page button on the Google Play Store indicating the 30-app limit when a user searches for an app. While this may seem to be a reasonable amount of data, for researchers conducting a wide search of apps over

the entire Google Play store, the breadth of the search needs to be much wider, without having to modify the code of the program itself.

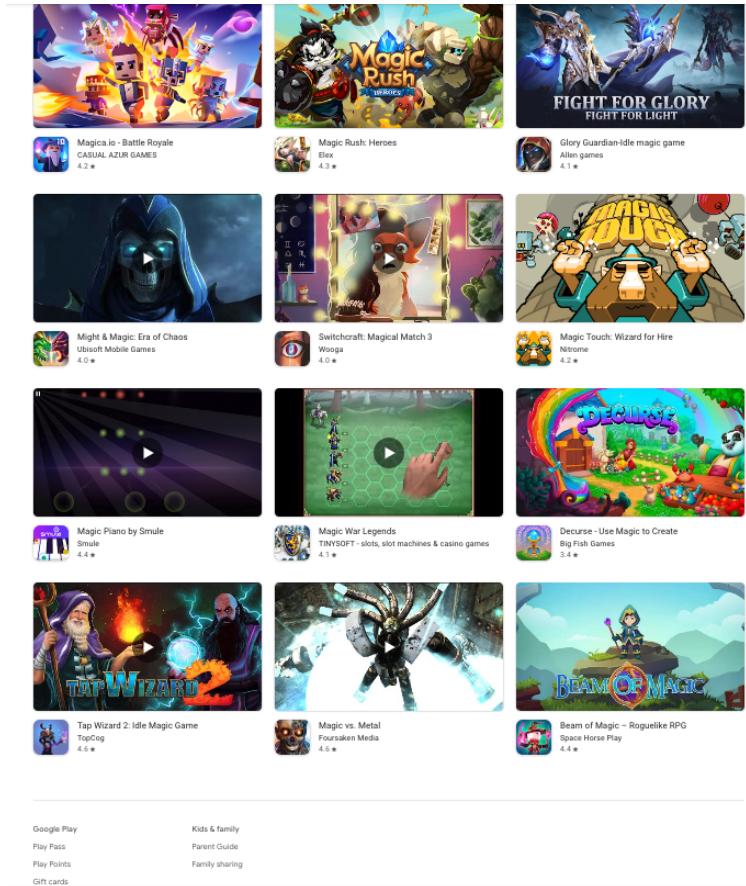
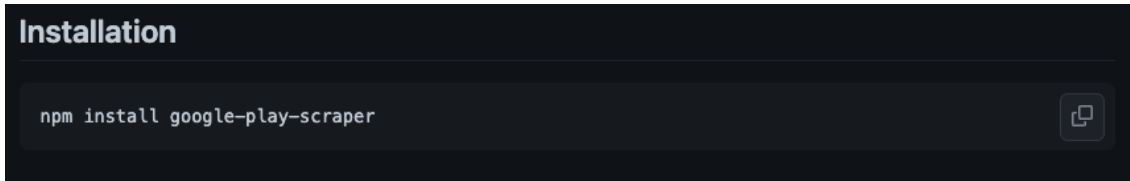


Figure 1.1: Google Play Store displaying only 30 apps using the search word 'magic'

Second, the facundoolano Google Play scraper does not have thorough documentation instructing how to install the program and simply provides a line of code to execute in the command line, shown in Figure 1.2. Even though we have technical experience, we found it difficult to install the facundoolano Google Play scraper without any outside resources. After extensive searching, we later consulted a tutorial on YouTube to correctly install the program. Furthermore, since this scraper is written in Node.js, it is implied the user's local machine requires Node.js in order for the scraper to function properly; however, there is no indication of such a dependency on the GitHub page and for users without experience with Node.js, they will not know to check or download Node.js for this tool. For non-technical researchers, the lack of clear documentation for an open-source tool will set back their research and consumer more time as they try to install and understand the tool itself.

A dark-themed terminal window with the title "Installation" in white. The command `npm install google-play-scraper` is displayed in a light gray font. To the right of the command is a small, light gray square icon with a white copy symbol inside, indicating a copy-to-clipboard function.

```
Installation  
npm install google-play-scraper
```

Figure 1.2: facundoolano Google Play scraper’s single line of installation instruction

1.3 Our approach

Our solution was to develop an open-source tool that non-technical researchers can use to easily scrape the world’s two most popular mobile app stores: Google Play and Apple App Store. The goal of our tool is to empower non-technical researchers to easily conduct systematic app reviews by reducing the time it takes to scrape app stores and eliminating the need to find a third-party technical partner.

To do this, we first understood the needs of researchers who wish to conduct systematic app reviews but find it costly to do so by conducting interviews with researchers who were conducting app studies or with researchers who scraped the app store in the past. We also provided consultation for researchers who have technical questions regarding scrapers and conducting systematic app reviews. Through this, we identified what researchers are looking for in an open-source tool. With these findings, we developed our own prototype of an open-source tool following the agile development methodology broken into 2-week sprints. While we developed our prototype, we used feedback from researchers to modify and improve iterations of our prototype to best incorporate the necessary functionalities to conduct a systematic app review.

Alongside our prototype for app store scraping, we created comprehensive and detailed documentation describing both how to install and use our tool and its functionalities in order to reduce the time and money spent on research using systematic app reviews. Our project provides non-technical researchers with tools and practices to better understand the current app ecosystem and to identify ways to improve it.

Chapter 2

User Research

2.1 Stakeholder needs

The main stakeholder we identified for our system was non-technical researchers who are interested in conducting a systematic app review. So far, we found that non-technical researchers need a comprehensive open-source scraper tool with clear documentation to easily execute data collection with our scraper. The scraper our main stakeholders wanted should be able to scraper data and return results from all the apps related to the filter in an exportable and readable format.

Another stakeholder was technical researchers; however, we decided to focus on the needs of non-technical researchers since they are most directly involved and affected by the outcomes of our project. Technical researchers will benefit from the ease and usability of our system; however, since non-technical researchers have little to no knowledge of scrapers and Node.js, it was crucial to address non-technical researchers' needs first.

2.2 User stories

We identified our main stakeholders to be non-technical researchers. In Figure 2.1, there are three different aspects of our system that non-technical researchers can utilize and benefit from. The first is the physical scraping tool itself. With the scraping tool, non-technical researchers will be able to easily collect data from apps that identify keywords specific to their research. For example, a mental health researcher can gather all the relevant metadata from any app that uses the keywords “mental health” to get a clearer idea of what the app ecosystem around mental health apps is. In the next branch of Figure 2.1, we compiled a “Best Practices Guide” researchers can use as a framework to conduct their own research, which in turn saves them time from looking at different resources to get them a similar result later.

Alongside our scraping tool and “Best Practices Guide,” we offered technical consulting to non-technical researchers who are currently conducting their own systematic app reviews with scrapers they are mainly unfamiliar with. Technical consulting gave non-technical researchers a quick and immediate resource to turn to when they have questions rather than having to scour the Internet for hours to find a solution. Technical consulting also provided us,

the SAR team the space to listen and understand the needs of our users, non-technical researchers.

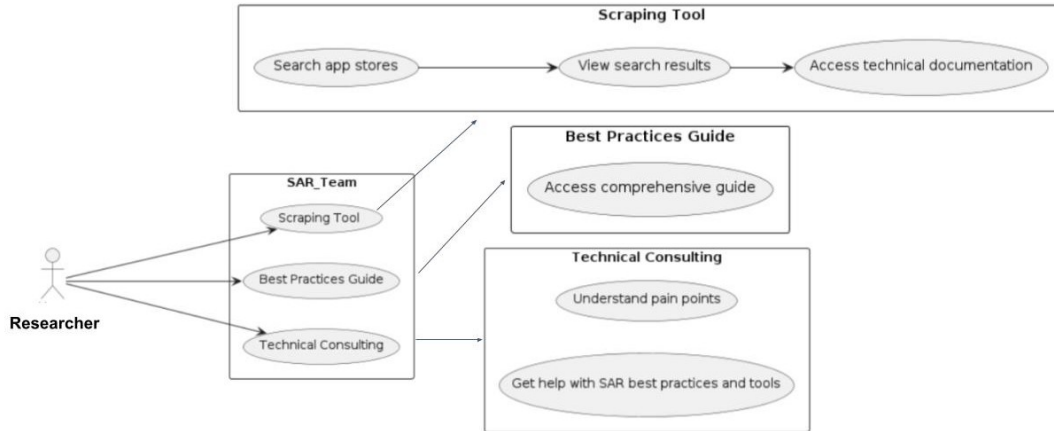


Figure 2.1: Systematic App Review user stories diagram

2.3 Methodology

For our tool, it is important to understand what the user wants in an open-source app scraper. To get a better idea of what features researchers will actually use, we decided to conduct user interviews with actual researchers who have either conducted systematic app reviews before or have done research with the Google Play Store.

To conduct user interviews, we created a user interview protocol with a list of questions to ask the researchers. The protocol allows for similar questions to be asked across different interviews and reminds the interviewer how to introduce ourselves to give the researchers a good idea of our mission.

In the protocol, we begin with an introduction section. This section is for us to introduce ourselves and our project idea to the researchers so they have a better understanding of what we are trying to do. Then we asked the researcher questions about their background like “What is your field of research?” and “Can you tell me a bit about the research in general?” We also asked about their background in web scraping and app store scraping to get a better idea of their technical and scraping experience. We then asked questions regarding the four different stages of a systematic app review: scoping, data collection, analysis, and reporting. We asked questions like “How do you store your data?” or “How do you determine the quality of the data?” to understand their process with this stage of research and to determine what works and what does not. At the end of each stage section, we ask “What is challenging about this stage?” to give the researchers an opportunity to reiterate any challenges that they have encountered for that stage of research or mention any new challenges they remembered after taking some time to reflect on their research. Overall, these questions helped us organize the researchers’ experience within these four stages of research while also identifying the researchers’ pain points with systematic app reviews.

For our user interviews, we reached out to 4 researchers: Jae Won Kim, a Ph.D. student in Information Science at

the University of Washington; Konrad Kollnig, an associate professor of Law and Technology at Maastricht University; Michael Hofer, a Ph.D. student in Computer Science at CU Boulder; and Arthur Tham, a former Master's student in Computer Science at the University of California, Irvine. The researchers we interviewed all had technical knowledge and experience with scraping, but of varying levels. Kim, Kollnig, and Hofer are researchers we met through the connection with our advisor, Dr. Kai Lukoff. Since we were using the facundoolano Google Play scraper as a basis for our tool, we wanted to interview researchers who have used the facundoolano Google Play scraper before. We contacted Tham after finding his research paper by searching the facundoolano Google Play scraper on Google Scholar.

2.4 Formative User Interview Results

From our user interviews with researchers, we found a few crucial pain points that we wanted to address in our tool. The biggest issue we learned was that app scraping is difficult and confusing for non-technical researchers, or researchers who were not familiar with the language the tool is written in.

One experience with the facundoolano Google Play scraper was Tham's difficulty using the scraper when he was conducting research on dieting, fitness, and weight apps. For some background, Tham was a Masters's student at the University of California, Irvine (UCI), studying computer science and published a research paper called *A content analysis of popular diet, fitness, and weight self-tracking mobile apps on Google Play* with his findings using the facundoolano Google Play scraper. He used the facudoolano Google Play scraper to scraper diet, fitness, and weight apps from the Google Play Store. Tham, however, found using this tool difficult because although he was a computer science student, he was more familiar with Python and had never used Node.JS before. As a result, in order to use the facundoolano Google Play scraper, Tham spent three weeks learning Node.js and how to use the scraper before getting any results. Furthermore, Tham spent additional time learning JSON in order to migrate the scraper results from the read-only command line JSON output to an editable, easy-to-use spreadsheet since the JSON output was difficult to read and understand because of how much data was outputted. Later in the interview, Tham said "We spent a lot of time learning Node.js (and JSON) more than we actually looked at the app stuff - like the data we got back" after telling us that a UI for the scraper would have been helpful when he was conducting research. This experience was just one person's; however, we can infer that many non-technical researchers are experiencing similar difficulties due to the necessary knowledge to know how to use Node.js.

To summarize, from our formative user research, we found that the existing facundoolano Google Play scraper is difficult for non-technical researchers to use because it requires the user to learn a completely new language in order to use simple commands, time they could dedicate to more important aspects of their research, and that the JSON object as an output is complicated to use since you can't edit the command line output.

Chapter 3

Design and Rationale

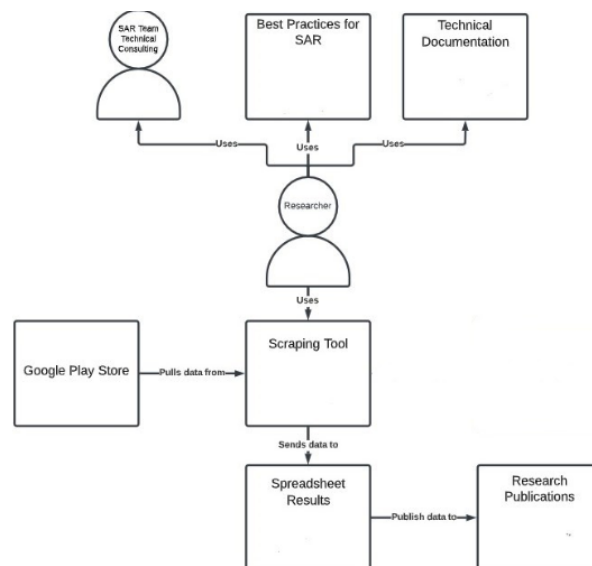


Figure 3.1: Systematic App Review C4 software architecture diagram

3.1 Design

We developed a scraping tool, by modifying an already existing open source tool, in order to allow non-technical researchers to easily get all the data they need without any technical knowledge. This scraping tool pulls data from the google play store, based on key search terms provided by the researchers, and returns the scraped data in csv format, which is easy for non-technical researchers to use with spreadsheets. We included technical documentation describing in detail how to use the SAR tool and its specifications, in a way that is accessible to non-technical users.

Additionally, we came up with best practices for conducting systematic app reviews (SAR), similar to those already existing for systematic literature reviews (SLR), for researchers to be able to use together with our scraping tool to collect the data that they need for their research in the most efficient way possible.

Finally, we provided technical consulting to researchers who had to perform systematic app reviews for their research but did not have the technical skills to do it on their own, in order to understand their pain points and what kind of tools and guidance they needed.

3.2 Functional requirements

The scraper must be able to take in a key search term from the user, then return all the apps on the app store or google play store that contain that keyword in the title. Additionally, the scraper must be able to take all of these apps and store them in an organized format that is easy for non-technical users to use, such as a spreadsheet.

3.3 Non-functional requirements

The scraper tool should be able to return all of the apps that fit the user's search criteria all at once, without being held back by app store query limits. Additionally, it should be able to return all of the results in a reasonable amount of time, so that it is convenient for researchers to use.

3.4 Rationale

Originally, we were considering using an already existing open source scraping tool, and creating best practices for researchers to use along with the tool to perform systematic app reviews. However, after trying to use the tool ourselves, we realized that even with our technical backgrounds, it was very difficult to set it up and get it to work properly, which means that it would be even more cumbersome and time consuming for non-technical researchers to use. Additionally, it did not have all of the features that we were looking for in a scraping tool, such as automatically storing all of the scraped data in a well organized spreadsheet. Therefore, we decided to create our own scraping tool, along with technical documentation, that would be easy for non-technical researchers to understand and use, as well as have all of the features that we were looking for, in order to allow the researchers to painlessly and conveniently get all of the data that they need.

Chapter 4

Technologies

4.1 System Components

In terms of outlining our technologies and test plan, we focused our efforts on empowering non-technical researchers to study mobile store apps. Our contributions revolved around the development of a user-friendly tool using various technologies, including Node.js, Electron, IPC, and the Google Play API.

To create an accessible and efficient tool, we chose to utilize Node.js, a widely adopted JavaScript runtime environment. By leveraging Node.js, we could develop the tool using JavaScript, a language familiar to many developers. This allowed for streamlined data handling and seamless interactions with external APIs.

For the purpose of building a cross-platform desktop application, we turned to Electron, a framework that combines the power of Node.js and Chromium. This decision enabled us to offer a consistent user experience across different operating systems, ensuring that researchers could access the tool regardless of their device of choice.

Inter-Process Communication (IPC) was crucial for smooth communication between the user interface and backend components in the tool's development. It facilitated efficient data exchange, allowing seamless retrieval of app data for comprehensive analyses. IPC mechanisms, like those in Electron, enabled efficient communication between frontend and backend, retrieving data from the Google Play API for a user-friendly presentation.

Furthermore, we integrated the Google Play API into our tool, using its extensive collection of mobile apps available on the Google Play Store. By utilizing this API, researchers can gain access to a better wealth of information, including ratings, reviews, and download statistics. This rich dataset can hopefully enable them to conduct thorough app reviews and analyses.

Throughout the development journey, we've remained dedicated to user-oriented design principles and adhered to best practices. Our aim was to create a tool that was intuitive and easy to navigate for non-technical users. By combining the aforementioned technologies and APIs, we successfully laid the foundation for a powerful and user-friendly tool that will empower non-technical researchers to delve into the world of mobile store apps and conduct systematic app reviews effectively.

4.2 Improved Components

While we made use of many pre-existing open-source tools and frameworks over the course of our project development, we believe that our final product offers something new that was previously unavailable in the open-source community. While various open source scraping tools already exist, our product comes in a complete, easy to use package that requires no technical or programming knowledge to be effectively used, making it accessible to users of all technical backgrounds who need to conduct systematic app reviews for their research.

Our product addressed several concerns that researchers had with existing open source tools. First of all, it gets around the Google Play Store's search limit of 30 apps per request by recursively returning recommendations of search results, allowing the user to collect data on a much larger quantity of apps at one time. Secondly, results are returned in a simple csv format, as opposed to JSON, so that users can easily view all of the data using a spreadsheet program of their choice, such as Excel or Google Sheets.

	A	B	C	D	E	F	G	H	I	J	K
1	url	appid	summary	title	developer	developerId	icon	score	scoreText	priceText	free
2	https://play.google.com	com.chic.colorlights	Color Lights Flashlight	Color Lights Flashlight	Chic Apps	N/A	https://play.google.com	4.4005604	4.4	N/A	TRUE
3	https://play.google.com	com.socialmobile	Turn your phone into	Color Flashlight	Notes	N/A	https://play.google.com	4.256873	4.3	N/A	TRUE
4	https://play.google.com	com.moonshine.color	Color Lights Flashlight	Color Lights Flashlight	Moonshine Apps	N/A	https://play.google.com	4	4	N/A	TRUE
5	https://play.google.com	com.apm.hd.flashlight	Color Flashlight : Color	Color Flashlight : Color	Art Photo Maker	N/A	https://play.google.com	3.4736843	3.5	N/A	TRUE
6	https://play.google.com	com.rvppstudios.flash	Brightest, fastest, and	Flashlight: Torch Light	RV AppStudios	N/A	https://play.google.com	4.479184	4.5	N/A	TRUE
7	https://play.google.com	com.flashlight.color	Disco party LED Light	Disco Light LED Color	TechTycoons	N/A	https://play.google.com	0	0	N/A	TRUE
8	https://play.google.com	com.pas.color.flashlight	Color Flashlight : Color	Color Flashlight : Color	Perfect Art Studio	N/A	https://play.google.com	2	2	N/A	TRUE
9	https://play.google.com	com.eduardo_roraj	Screen Flashlight	Screen Flashlight	Eduardo Rojas Sorita	N/A	https://play.google.com	4.4	4.4	N/A	TRUE
10	https://play.google.com	com.rfxdev.ColorLight	Disco Color Lights Flashlight	Disco Color Lights Flashlight	rfxdev	N/A	https://play.google.com	4.428571	4.4	N/A	TRUE
11	https://play.google.com	com.conceptual.color	flashlight flash on camera	Flashlight : Flashlight	Conceptual App IIT	N/A	https://play.google.com	3.765625	3.8	N/A	TRUE
12	https://play.google.com	com.fonfon.yihgedi	Bright LED Flashlight	Flashlight	Android-studio	N/A	https://play.google.com	4.8246207	4.8	N/A	TRUE
13	https://play.google.com	com.handysoft.ledlight	Brightest Flashlight	High-Powered Flashlight	High-Powered Flashlight	N/A	https://play.google.com	4.802504	4.8	N/A	TRUE
14	https://play.google.com	com.peacock.flashlight	It quickly and easily	Flashlight	Lighthouse, Inc.	N/A	https://play.google.com	4.2508755	4.3	N/A	TRUE
15	https://play.google.com	com.simplerobileto	A clean flashlight widget	Simple Flashlight	Simple Mobile Tools	N/A	https://play.google.com	4.4	4.4	N/A	TRUE
16	https://play.google.com	com.eco.flashlight	Flashlight - Torch Light	Flashlight - Torch Light	Eco Mobile	N/A	https://play.google.com	4.6098204	4.6	N/A	TRUE
17	https://play.google.com	com.splendapps.torch	Flashlight is free, into	Flashlight	Splend Apps	N/A	https://play.google.com	4.580645	4.6	N/A	TRUE
18	https://play.google.com	com.devuni.flashlight	Tiny Flashlight + LED	Tiny Flashlight + LED	Nikolay Ananiev	N/A	https://play.google.com	4.502587	4.5	N/A	TRUE
19	https://play.google.com	com.codverter.wearflash	Description: Wear Flashlight	Wear Flashlight	CodVerter	N/A	https://play.google.com	4.8	4.8	N/A	TRUE
20	https://play.google.com	com.digitalchemistry	Turn your mobile phone	Flashlight Plus - LED	Digitalchemistry, LLC	N/A	https://play.google.com	4.529412	4.5	N/A	TRUE
21	https://play.google.com	com.axiomatic.flash	Your smartphone's	Flashlight	Axiomatic Inc.	N/A	https://play.google.com	4.5308056	4.5	N/A	TRUE
22	https://play.google.com	artline.com.galaxy	Unique LED Flashlight	Flashlight LED - Universal	LexaUA	N/A	https://play.google.com	4.7583847	4.8	N/A	TRUE
23	https://play.google.com	ru.irkang.balsan.shot	The most simple LED	Icon Torch - Flashlight	Aleksandr Balaev	N/A	https://play.google.com	4.5423727	4.5	N/A	TRUE
24	https://play.google.com	pl.nenter.app.flashlight	Flashlight Galaxy is	Flashlight Galaxy	Szymon Dja	N/A	https://play.google.com	4.671138	4.7	N/A	TRUE
25	https://play.google.com	at.bleeding162.flash	Small and stable flashlight	Flashlight Widget	David Medenjak	N/A	https://play.google.com	4.3545456	4.4	N/A	TRUE
26	https://play.google.com	apps.r.flashlight	Simple, advertise	Flashlight	R. Apps	N/A	https://play.google.com	4.5	4.5	N/A	TRUE
27	https://play.google.com	org.nixgame.flashlight	The flashlight or photo	Flashlight	NixGame	N/A	https://play.google.com	4.354839	4.4	N/A	TRUE
28	https://play.google.com	uk.forbis.flashlight	Screen Lamp & Flash	Screen Lamp & Flash	Apps by Forbis	N/A	https://play.google.com	4.571429	4.6	N/A	TRUE
29	https://play.google.com	cz.nowi.ledflashlight	A powerful, elegant	Bright flashlight photo	NoWi Apps	N/A	https://play.google.com	4.66	4.7	N/A	TRUE
30	https://play.google.com	com.midainc.idag.fr	One of the most popular	Flash call-flashlight	MarsSofts	N/A	https://play.google.com	4.290323	4.3	N/A	TRUE
31	https://play.google.com	com.flashlightsuper	The application has	Flashlight	Vmons	N/A	https://play.google.com	4.46	4.5	N/A	TRUE
32	https://play.google.com	com.humberto.flashlight	Tired of other flashlight	Flashlight: No Permission	Humberto	N/A	https://play.google.com	4.3398695	4.3	N/A	TRUE
33	https://play.google.com	dev.dect.wear.flashlight	Features: Set the	Flashlight	Douglas Silva	Dec	https://play.google.com	0	0	N/A	TRUE
34	https://play.google.com	com.flashlight.torch	Flash App: Flash & Flash	Flash App: Flash & Flash	Govo Tech by Volio	N/A	https://play.google.com	4.6	4.6	N/A	TRUE
35	https://play.google.com	com.ktwapps.flashlight	This is one of the best	Flashlight : LED torch	KTW Apps	N/A	https://play.google.com	4.553	4.6	N/A	TRUE
36	https://play.google.com	com.mobi.freeapps.flashlight	Free, Handy and	Flashlight	Light Apps Studio	N/A	https://play.google.com	4.799045	4.8	N/A	TRUE

Figure 4.1: App data output in spreadsheet format

Additionally, our product uses a graphical user interface, allowing users without any technical experience to easily interact with it without needing to write any code. Finally, our entire product is contained within a single executable

file, so that users can launch it simply by opening the file, and don't need to worry about having Node.js or any other dependencies installed on their computer, or having to use npm to install the right packages and set them up properly.

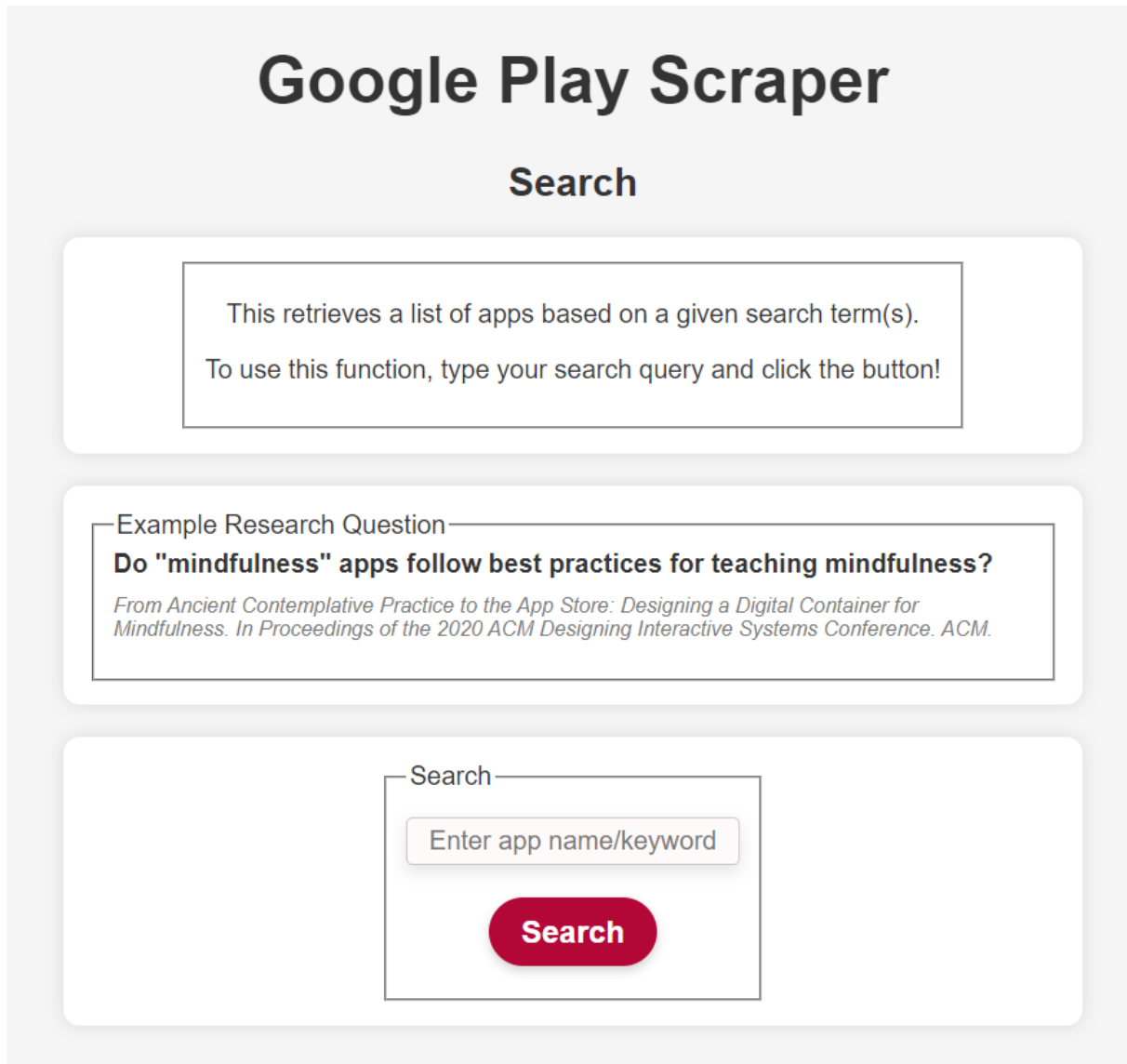


Figure 4.2: Scraper graphical user interface

Chapter 5

System Evaluation

Our system was evaluated based on the amount and quality of data that we were able to scrape, and the ease of use of the application. We evaluated the system based on the number of apps that it can scrape as well as how precise the results are to the search parameters. We also then further evaluate the number of details we are able to obtain for each app. As for ease of use, relied on feedback obtained through user testing.

5.1 Internal Testing

The app was tested internally through different iterations of the app. We ran the first set of tests after we finished the first iteration and then further improved on areas that we deemed lacking. Afterwards, our team determined the direction that we planned on heading next based on the results acquired. Each time we built a new version, we tested out the app on different parameters such as speed, quantity, and quality.

5.2 External Testing

We ran external tests by having researchers and the individuals of our intended audience test out the app's usability. To do this, we reached out to Tham, who we interviewed before in Chapter 2 - User Research. Similar to our formative user interviews, we developed a user testing interview guide to ensure all the external testing interviews are conducted similarly and that we are asking similar questions. Generally, we asked the researcher to use the tool as if they were conducting their research to understand if this tool is intuitive and can easily support the researcher in actual research. We also asked them to go through the different functions to ensure each function page and the prompt is easy to understand because we wanted all functions to be straightforward, even if it is not used at the moment. By conducting these tests, we were able to determine what aspects of the tool were comprehensive and easy to use and which aspects needed improvement. We also compared the results of external testing to the results of internal testing to determine what aspects of the program need improvement to allow users of different technological backgrounds to have the same results when using our app.

5.3 Testing Results

From the external testing, we found that our product was helpful in easily gathering the data scraped from the Google Play Store. The new UI helped users quickly scrape data without having to learn or handle Node.js, which can get messy. Furthermore, as an executable, our product eliminated the concern of not having Node.js installed on the local machine. While the product is helpful, there were some points of improvement the researchers we interviewed highlighted.

Some researchers were surprised when the tool immediately downloaded a csv of the results after clicking the 'search' button on any of the function pages. This is because the term 'search' as the button text implied an interface or interaction similar to a search engine that would show a preview of the results or show the results on the page. This was confusing at first for the researchers. It was suggested to change the text from 'search' to 'download' if we plan to have the user download the csv immediately. It was also suggested to add text to inform the user that by clicking the button, it will download a csv or that a csv is the format the data will be in. Another suggestion was to include a sample of the csv header on the page to let the user know what kind of data they will be scraping from the function.

The researchers also suggested including loading text to let the user know that the csv is downloading. This feedback was most received when testing the 'reviews' function since the reviews function usually takes more time to compile the data because there are thousands of reviews the program is scraping. While the reviews function is helpful, without loading text, the user will be confused as to whether the tool is broken or not. In fact, in a couple of the external testing interviews, researchers closed the program thinking the program had crashed since there was no indication of whether the reviews function was simply taking more time to retrieve the reviews of an app or if it stopped working.

Similarly, we received feedback on including an error message when there is no data to be retrieved. For example, if a new app had just come out on the Google Play Store, it will not have any written reviews yet. If a user were to try and scrape the reviews of the new app, the tool would simply erase the search parameters, not informing the user that there was no information to scrape. The error message implementation would be important to improving the user experience of the tool.

Another suggestion was to include an easier method to search for apps. The search function is simple since it simply uses key terms; however, many other functions like 'reviews' and 'permissions' require an app id, which is confusing and cumbersome to retrieve if the research is scraping data on multiple apps. It was suggested to implement some method to easily copy the app id from the Google Play Store or to simply input the app URL since the beginning of the app page on the website is the same.

The user experience of our scraper is crucial to the helpfulness of our product to non-technical researchers because if the product's user experience is poor, researchers will not use the product. All the feedback listed above is important

to the further development and improvement of our program for non-technical researchers. We want non-technical researchers to easily scrape app data from the Google Play Store without having to learn complicated coding languages.

Chapter 6

Implementation

6.1 Timeline

The first quarter was mainly focused on planning and information gathering. For us, this meant gathering information on the existing market of scraper tools, identifying the limitations of each tool and aspects we would like to implement in our own prototype, and understanding the goals for the project. We also interviewed one researcher who is currently working on a systematic app review.

In the second quarter, we continued to conduct user interviews to get a better understanding of the hardships of conducting systematic app reviews and what aspects of the facundoolano Google Play scraper researchers like, and what was difficult. Additionally, during the second quarter, we began the actual development of our open-source tool. During this quarter, we had our breakthrough with the 30-app limit discussed in Chapter 1 as well as product directional shifts as a result of new implementation ideas.

In our third quarter, we focused on further developing our UI of the Google Play scraper. This mainly meant designing a UI and mapping out the flow for how users would interact with the product, integrating all important features from the facundoolano Google Play scraper, and compiling the JSON results previously obtained into a csv that is easier for researchers to read and use. We also conducted user tests to receive feedback on what aspects of our tool are helpful and what needs to be improved upon for a polished user experience.

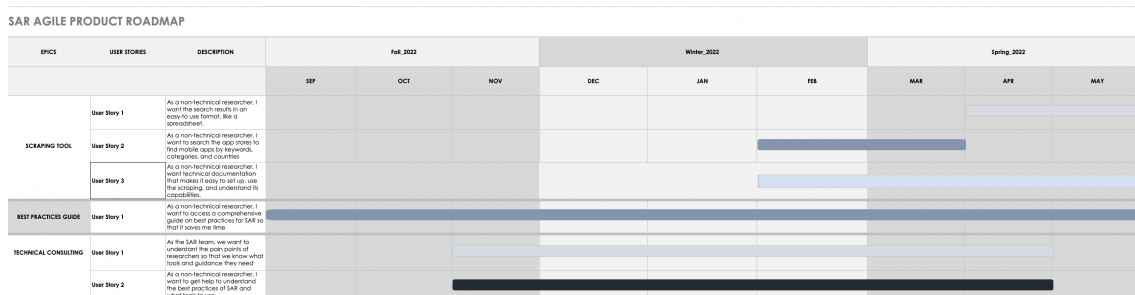


Figure 6.1: SAR team timeline using the agile development method

6.2 Agile software development

For our project, we used the agile software development methodology. This means that we divided the project into multiple different 2-week sprints and completed tasks within those given 2 weeks. We chose to work using the agile method because with this method, we can work on multiple epics (as seen in Figure 2.1) at the same time. For example, one person can work towards completing epic 3 by conducting an interview with a non-technical researcher whereas another can work on epic 1 by working on the code for the scraper.

The agile methodology allowed us to work on many aspects of the project in parallel with each other. An important aspect of the agile method is the reflection at the end of each sprint. After each sprint, as a team, we discussed what went well, what needs improvement, and our plans for improvement towards the next sprint. This allowed us to reflect not only on the actual work we produced but our process, communication, and collaboration as a whole.

6.3 Project Risks

Our tool currently has a recursive workaround for the 30-app limit the Google Play Store has on their web version. A risk to this is if Google makes any changes to their app limits that block our recursive method which would then hinder any function that uses this method from retrieving the full data. For the time being, we believe there will not be any major changes in Google's app limit policies so we consider this as a low risk, for now. Another risk is the possibility of malicious users abusing our tool's data scraping capabilities to collect app data and develop their own app. Because our program is made with the intention for research purposes only, the possibility to use our product for anything else is a risk we are taking. One way to hinder malicious users is to limit the number of times they can scrape app data and to incorporate user accounts and authentication so that users can only use the program if they have a .edu email, an email domain extension that is reserved for academic and educational institutions. Luckily, we are not scraping actual user data, like passwords and addresses, and everything our tool scrapes can be found on the Google Play Store.

Chapter 7

Conclusion

7.1 Summary

In the 21st Century, apps are increasingly integrating into our lives. Not only are there apps for games and social media, but also mental health tools and fitness; there are apps for essentially everything. With the increasing development of apps, researchers want to better understand how these apps are influencing our daily lives by studying the app ecosystem. The facundoolano Google Play scraper is a popular open-source tool that researchers used in the past to scrape app data from the Google Play Store; however, this tool is difficult to use for non-technical researchers. Our solution was to create an easy-to-use open-source scraper with a UI that researchers can easily interact with, without having to learn the complicated nuances of Node.js and JSON. Our tool allows researchers, non-technical or technical, to scrape thousands of app data on the Google Play Store with a click of a button while storing the data in a simple and clean csv format that researchers can make notes and modify at will and with ease.

7.2 Lessons Learned

Being a year-long project, developing our open-source Google Play scraper taught us many lessons. For many of us, this project was the first project we had to navigate on our own. Prior to this, in-class projects provided some level of direction and implementation plans in order to easily complete in the span of 10 weeks. This project, however, was different. We only had a prompt to help non-technical researchers better scrape app data and were to figure out how to develop a tool from the beginning. Because of this, we learned about project management, the agile software development methodology, and effective communication between team members but also for external involvement, like the researchers we interviewed. We learned how to lead a project on our own and how to decide the direction we wanted to develop in. There were moments where our vision and plans were misaligned with each other's which created confusion in our next steps. During these times, we paused development to realign ourselves, ensuring that everyone was able to voice their ideas, rationales, and concerns. After these meetings, we all had one vision and moved forward to bring that vision to life. Regarding lessons with user experience interviews, we learned how to develop an

effective interview guide that asks the questions we want the answers to while refraining from guiding the researcher to answer a certain way. This is important because we wanted unbiased answers and genuine feedback when conducting our research so it was important that we presented no bias. Because the users of our product are researchers, it was crucial to receive genuine feedback on both our formative and evaluative user interviews.

7.3 Next Steps

Looking into the future, the next steps for the open-source tool could be to implement user accounts to save their search history and data, in addition to being a method of authentication to restrict and block malicious intent. This would require more development to work with storing data like search data or user profiles on servers. This feature would also highly require the migration of the tool as a downloadable executable to a web-based application for easy access and user accounts. Furthermore, more user tests could be conducted since we weren't able to interview too many researchers due to the time constraint of the final quarter. More user feedback would provide a clearer idea of what are the most important features of the tool to be improved on. Our tool currently only allows users to select one function to retrieve app data; however, the ability to further filter the search parameters is a strong possibility for future improvements as well since researchers will want to scrape more specific queries.

Chapter 8

References

1. Tham, A., Kim, L., Victory, S., Chen, Y., Zheng, K., & Eikev, E. V. (2020, March 23). A content analysis of popular diet, fitness, and weight self-tracking mobile apps on Google Play. iConference 2020 Proceedings. <https://www.ideals.illinois.edu/items/114127>