Executive Pitch

Motivations:

Our innovation aims to solve the ongoing issue of ineffective communication between application developers and users in regards to privacy concerns. Although documentation of privacy related concerns exists, the accessibility and readability of the documentation has been brought into question since the majority of users lack understanding of privacy policies.

Google's attempt to ethically combat this problem was to require developers to declare how they handle user data. As of July 20, 2022 developers are required to provide details about data protection in the form of a "data safety label" [1]. However, as this requirement of data safety labels is new, questions arise on the effectiveness of these labels. This includes whether users understand the condensed descriptions of the functionality and services of the application and whether users are using these data safety labels to make privacy decisions on their devices. The use of our research tool will assist in providing users an in-depth comprehension of data privacy labels. In addition, it can provide researchers with feedback for improving privacy documentation.

Users:

The first users our application is designed to help are the end users of the application, those who actively make informed decisions regarding their digital privacy. The first user group provides the data set that we will analyze and use to evaluate the effectiveness of the privacy labels; and the general interpretation of these labels; the first users are those who will answer the survey questions. The second set of users, of our final product, are the Google developers that will use the results provided by the first users, taking the surveys, to implement and design new nudging tools. The nudging tools are designed with the goal of providing users with the most helpful information regarding their privacy at the most effective time. Additionally, the third and final users are other researchers interested in the effectiveness and understanding of data safety labels. The goal of our application is to be able to expand it in order to run future research studies on the effectiveness of other elements of data safety.

Technical Summary

The key objectives of this project are to generate a survey that is not only dynamic by nature, but hosts the survey on a back-end where the least number of updates would have to be made to the front-end of the application. If updates to the survey are necessary, the goal is that they occur on the server side and the physical application will not have to be reinstalled for users participating in the study. Another objective of the project is to have a simple and engaging front-end display.

Currently, the entire front-end is built in Android Studio. The application, Android Studio, allows us to not only build the application, but test and see the front-end interactions on a google device. We plan to host a docker container on a server with multiple different systems within the container. Mainly, our back-end will consist of an NGINX web server, an MYSQL DB, a MongoDB, Django, an updater, and a worker that will scrape the google play store. Each of these pieces connects to different aspects of the project.

An example of the flow of data could be described as follows:

- Package name sent from UI to Django
- Django calls the worker to scrape the play store for a data safety label
- Worker stores JSON of data safety label in MongoDB
- Updater recognizes the addition of the label in MongoDB and calls the web server
- The web server generates the HTML for the survey
- MySQL DB stores the results from the front-end

Project Specifications

User Stories

As a user participating in the study, I would like to download the Data Safety Research Tool application from the Google Play Store.

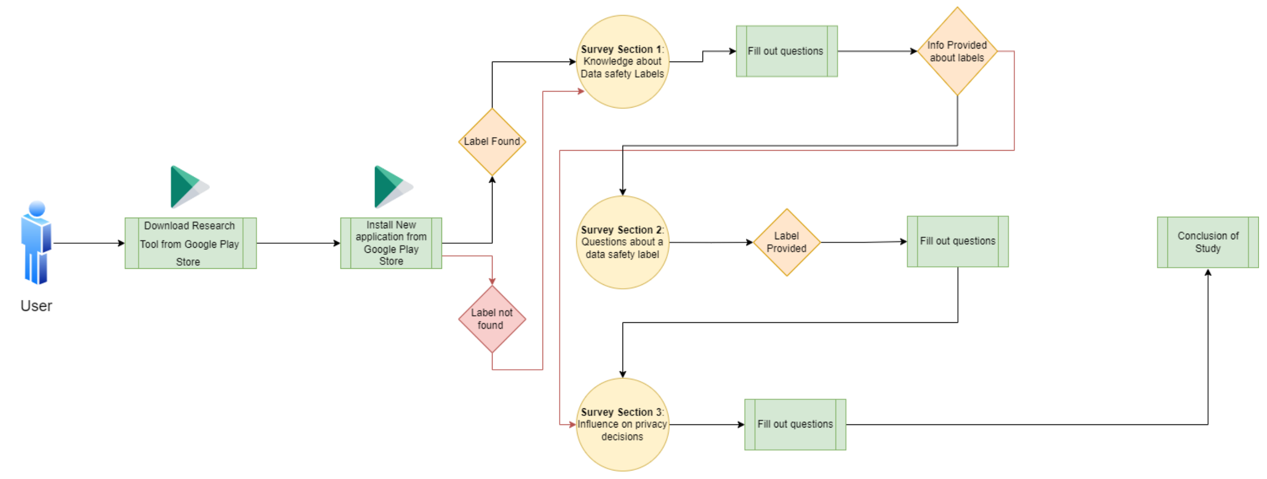
As a user participating in the study, I would like to see the notification prompt to start the survey when downloading a new application from the Google Play Store. When clicking the notification it should redirect me to a survey.

As a user participating in the study, I would like to answer the questions on the survey without any technical difficulties. I would also like the survey to be readable and easy to understand.

As a researcher, I would like to see the user having a unique id after making a prolific account. This ID should be sent to the backend so that multiple users can use the system at once.

As a researcher, I would like to have the user results in an organized format in a database. This would allow me to analyze the data and derive results based on user input.

Flow Diagrams


User
Download Research Tool from Google Play Store
Install New application from Google Play Store
Label Found
Label not found
Survey Section 1: Knowledge about Data safety Labels
Fill out questions
Info Provided about labels
Survey Section 2: Questions about a data safety label
Label Provided
Fill out questions
Conclusion of Study
Survey Section 3: Influence on privacy decisions
Fill out questions

Summary of Flow Diagram:

- User creates a Prolific Account
- User downloads our application
- Their user ID is sent to backend
- User downloads an application. That application's id is sent to the backend and scraper will parse Google Play Store for the label.
- If no label is found, the user will answer survey section 1 and 3.
- If a label is found then the user will answer all questions of the survey.
- Results are stored in the database.
- Researchers should be able to look in the database and see data inserted by participants.
- Researchers will use data to analyze and determine results of research questions.

User Mock-Ups:

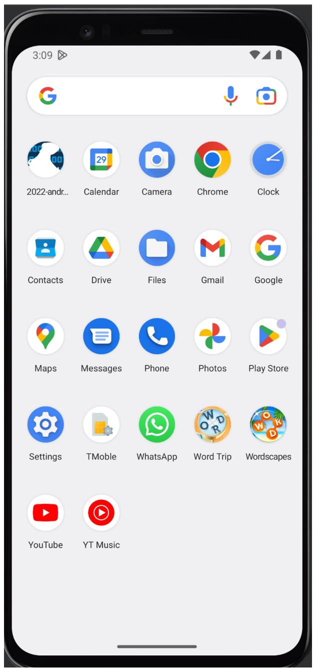
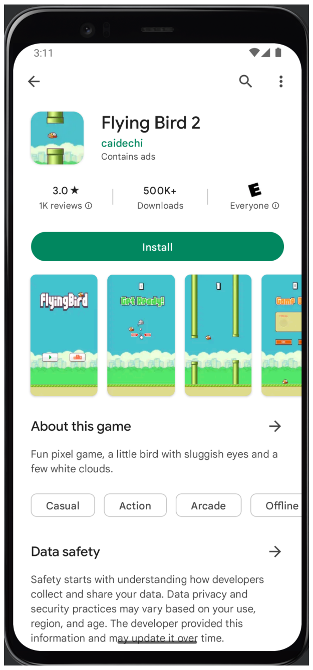


Figure 1



Fun pixel game, a little bird with sluggish eyes and a few white clouds.

Safety starts with understanding how developers collect and share your data. Data privacy and security practices may vary based on your use, region, and age. The developer provided this information and may update it over time.

Figure 2

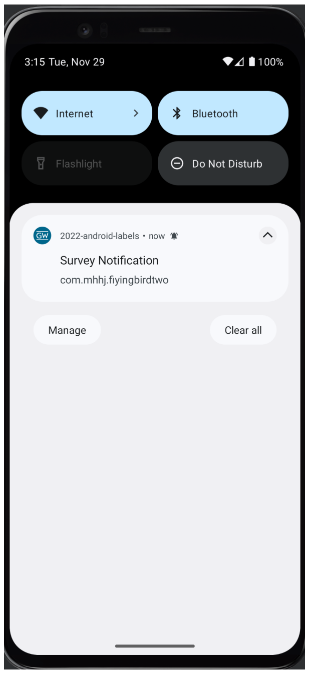
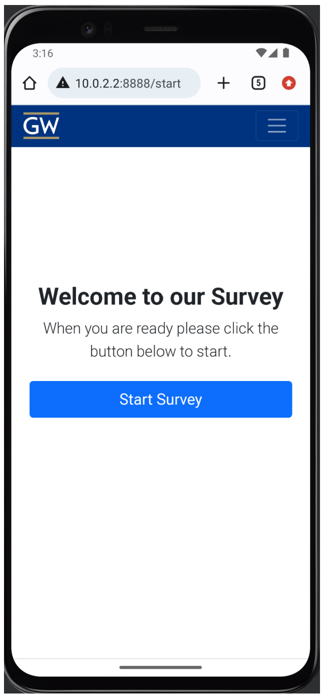


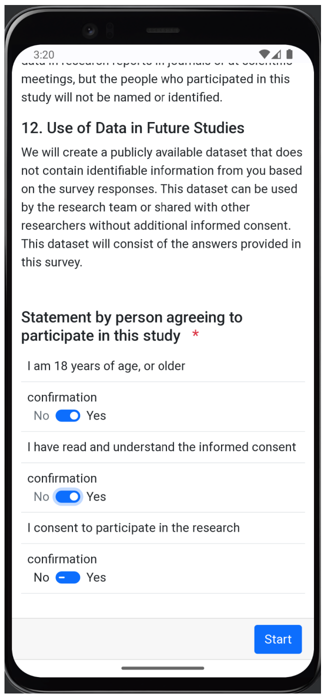
Figure 3



Welcome to our Survey

When you are ready please click the button below to start.

Figure 4



meetings, but the people who participated in this study will not be named or identified.

12. Use of Data in Future Studies

We will create a publicly available dataset that does not contain identifiable information from you based on the survey responses. This dataset can be used by the research team or shared with other researchers without additional informed consent. This dataset will consist of the answers provided in this survey.

Statement by person agreeing to participate in this study *

I am 18 years of age, or older

confirmation
No Yes

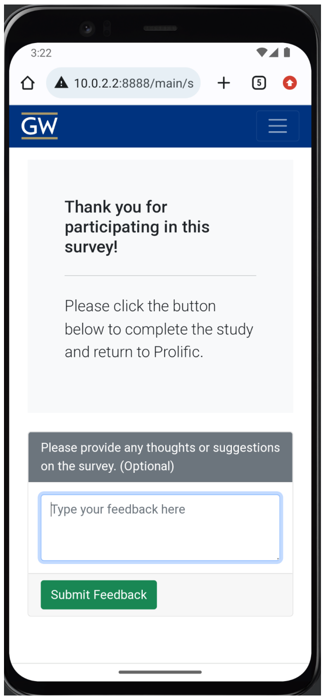
I have read and understand the informed consent

confirmation
No Yes

I consent to participate in the research

confirmation
No Yes

Figure 5



Thank you for participating in this survey!

Please click the button below to complete the study and return to Prolific.

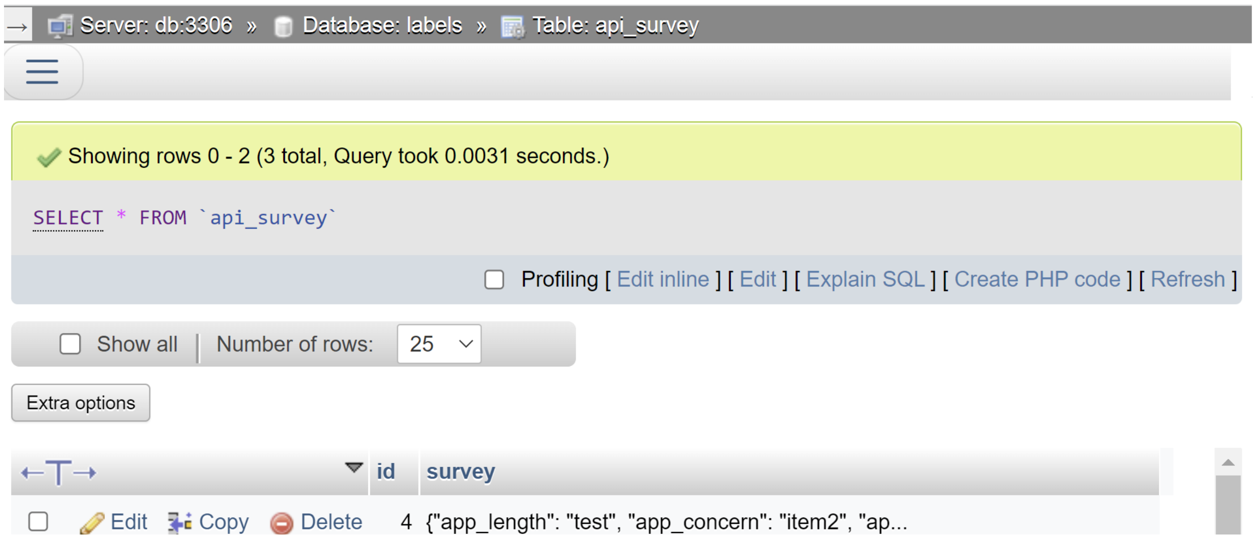
Please provide any thoughts or suggestions on the survey. (Optional)

Type your feedback here

Figure 6

The user will download our android application as shown in figure 1. The user will then download a new application as seen in figure 2. Our application will then populate a notification as shown in figure 3. The user will then be prompted to start the survey as seen in figure 4. Some of the questions structure and final feedback can be seen in figure 5 and 6.

Researcher Mock-Ups


Server: db:3306 » Database: labels » Table: api_survey

Showing rows 0 - 2 (3 total, Query took 0.0031 seconds.)

```
SELECT * FROM `api_survey`
```

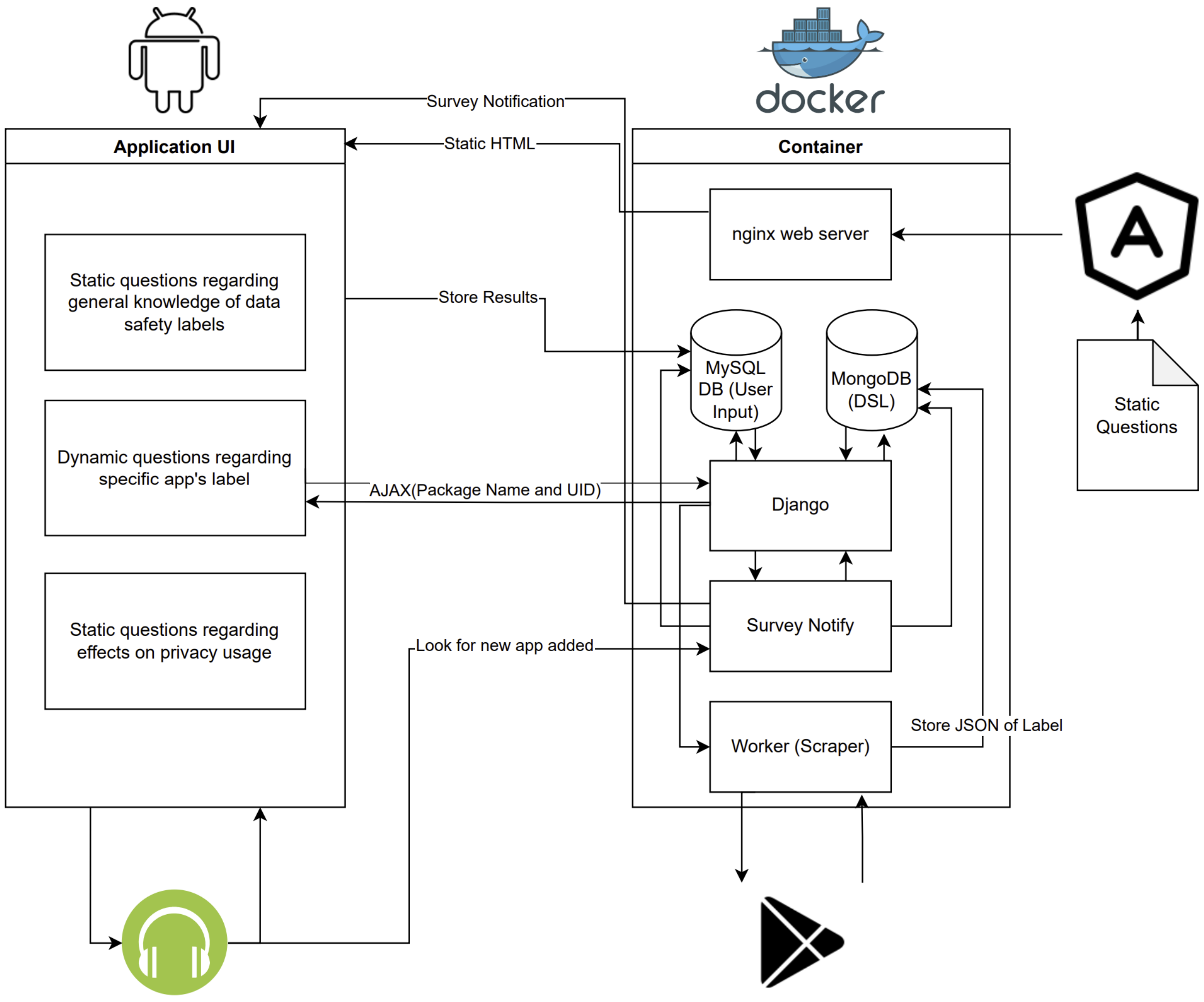
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Extra options
Edit Copy Delete

id	survey
4	{"app_length": "test", "app_concern": "item2", "ap...

Results will be stored in a MySQL Database for analysis that the research can access.

Technical Specifications

Architecture/System Diagrams



External APIs and Frameworks

Backend APIs:

Promise API:
Goal: Use this API to parse Google Play Store for labels and map them into array objects. This will allow us to easily manipulate the data into a format we wish to use such a json file that is broken up by category.
Description: This API is used in the scraper.js file in our web scraper.
Endpoints used: This web scraper will access:
${BASE_URL}/store/apps/data safety

Where BASE_URL is the playstore link plus the unique ID for the app that was downloaded.
https://play.google.com/store/apps/details?id=com.whatsapp

Ramda API:
Goal: Placeholder value used to specify "gaps" within current functions.
Description: This API is used in the scraper.js file in our web scraper.

Express API:
Goal: Use this as a built-in middleware function. It parses incoming requests with JSON payloads.
Description: Using this to communicate between docker containers in the backend. The backend consists of multiple docker containers that need to trigger certain actions when something is completed. For example, when a user downloads an application, the json format of the label will be sent to MongoDB. Then, when the user clicks on the notification to take the survey, this JSON file will be sent through the backend to create a list of dynamic questions. Once the user fills out these questions a JSON string will be sent to a database where it will be stored. This will also be helpful when threading. We will need to thread each of these processes as multiple users may be using our app or doing the research study at once. This framework can also allow us to interact and obtain data from the web framework as well.

Endpoints:

Frontend APIs:

Intent API:
Goal: Access the package name of the new application that the user downloaded.
Description: This API is used in MainActivity.java and MyReciever.java in the App repo for our project. It connects to the internal OS of the phone and is called when the ACTION_PACKAGE_ADDED intent occurs. The application is always listening for this intent to populate within a device, and the result of the intent is sent to the notification API.

Notification API:
Goal: Populates a notification with the added package name and redirects the user to start the survey
Description: The API is used in MyReciever.java in the App repo for our project. This API connects to the internal notification center for Androids and is called after the Intent API has collected the package name. The parameters passed in to the API include the package name and the redirect link and the results are shown in the notification for the user.

Endpoints Used: GET http://10.0.2.2:8888/start?appname={appname}

<u>Frameworks/Other External Tools</u>

Angular
Goal: Use angular framework to build the Survey. Angular has features which convert it to mobile format using android studio.
Description: Using angular to create static questions for the survey. All stylistic elements on our survey will use angular.

Django
Goal: Framework for the backend web application based on python.
Description: Communicates between the frontend and the backend for accurate survey generation.
Ports: Connected on 8000:80

Android Studio
Goal: Builds basic front end application
Description: Provides a unified android based environment to independently build, test, and debug our application. Android Studio is an IntelliJ based IDE which we use for our entire APP repo.

MySQL - PhpMyAdmin
Goal: Saves all survey results from users
Description: This database connects through our Django framework and the NGINX web server to store the results the user enters into the survey.
Ports: Connected on 8080:80

NGINX
Goal: Hosts the static angular questionnaire
Description: This web server connects through the Docker Compose and the Django framework to host the static questions of section 1 and 3 of our survey.
Ports: Connected on 8888:80

MongoDB
Goal: Saves all JSON strings of data safety labels
Description: This database connects through our Docker Compose framework and the web scraper to store the results of the scraped data safety label from the Google Play Store.
Ports: Connected on 8081:8081

Docker-Compose
Goal: Hosts the numerous moving components of application backend
Description: This tool allows us to define each backend component and run multiple containers as a single service. Each of the containers can run in isolation but they interact with each other when needed for the purposes of the application. The docker container system is described in the docker-compose.yml file.

Algorithms

Web Scraper:

Our first algorithmic operation is our web scraper. This is designed to parse the Google Play Store and obtain the data safety label information of a specific application. If the label is found, a json string will be sent back to the server and stored in MongoDB.

The web scraper consists of a couple of main components. The Promise API which allows the mapping of the data into an array format. Then there is a request and throttle js class. The request class makes a request to the Google Play Store which allows us to parse. The throttle class allows a limit to be placed on the number of requests obtained at a certain time.

The scraping should occur once the user downloads a new application. So there will be a listener on the actual device itself that will send a request to the backend which will then trigger the scraping to occur. Our goal is to have this request contain the user ID obtained from prolific and then the ID of the application itself so we can scrape the store. This process should repeat itself for every application downloaded. If multiple users are downloading and using our application at once, there will be threaded processes that will perform the scraping simultaneously.

Dynamic Survey Question Generation:

This algorithm uses the data present in the JSON of the data safety label to generate questions in relation to each specific application. The Django framework will pull from the MongoDB to access the label for the particular application. Our algorithm will then make questions visible to the user depending on if the particular set of data is collected by the application. There will be a cap on the number of dynamic questions a user can be asked in order to prevent answering fatigue and provide the researchers with the most informative set of data.

Generation of User IDS:

Randomly generates a user ID using alphanumerics. It uses an 8 digit combination of numbers and letters. It is in the views.py file which should be triggered when we build the docker compose. This will assign a unique ID for every user that is using our application. So this is per device basis. Having a unique ID is important because if we have multiple users accessing our application and carrying out these processes at once we will have multiple threads that are executing the scraper, the survey question generation, and the storing of data. We need the ID to make sure that all the data stored relates back to the data entered on each device.