LAST EDIT 11/29/2022

Writing 3

Miles Grant, Olivia Legault, Rubin Roy

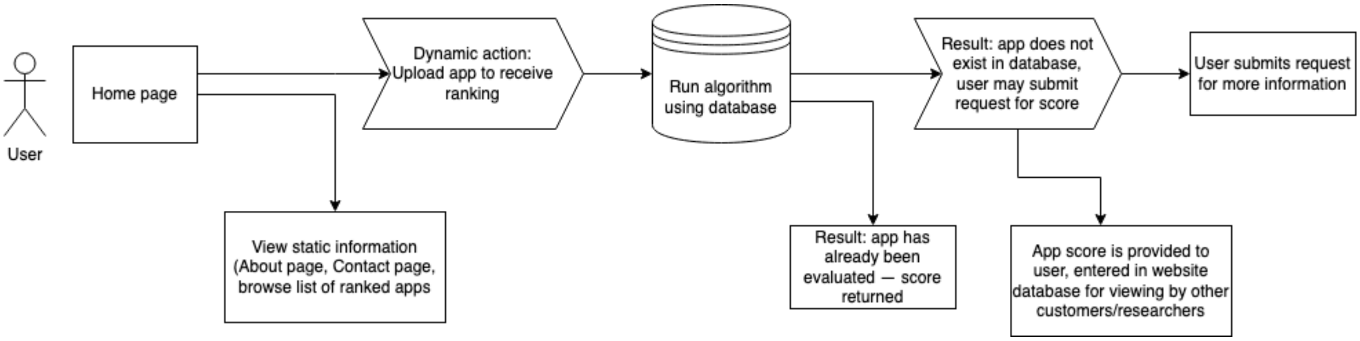
I. Product Specifications

A. User Stories

1. As a user of reproductive health app(s), I would like to know how well these apps are preserving my privacy and sensitive health data, so that I can make decisions about which apps most accurately match my needs and desires on data privacy.
2. As a researcher or journalist, I would like to have a "one-stop shop" for analyzing reproductive health app privacy to inform future studies and allow comparison between different applications.
3. As a pro-choice advocate, I would like to have a platform for evaluating app privacy to inform my recommendations and policy positions.

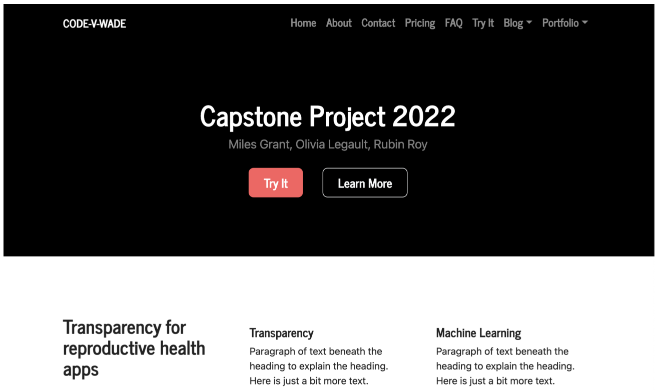
B. Flow Diagrams

1. User Interaction


User
Home page
Dynamic action: Upload app to receive ranking
Run algorithm using database
Result: app does not exist in database, user may submit request for score
User submits request for more information
View static information (About page, Contact page, browse list of ranked apps
Result: app has already been evaluated — score returned
App score is provided to user, entered in website database for viewing by other customers/researchers

C. Mockups/Wireframes

1. Basic UI Components:


CODE-V-WADE Home About Contact Pricing FAQ Try It Blog Portfolio

Capstone Project 2022

Miles Grant, Olivia Legault, Rubin Roy

Try It Learn More

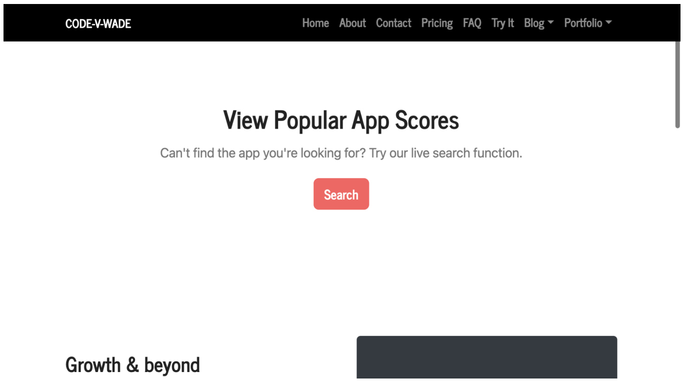
Transparency for reproductive health apps

Transparency

Paragraph of text beneath the heading to explain the heading. Here is just a bit more text.

Machine Learning

Paragraph of text beneath the heading to explain the heading. Here is just a bit more text.


CODE-V-WADE Home About Contact Pricing FAQ Try It Blog Portfolio

View Popular App Scores

Can't find the app you're looking for? Try our live search function.

Search

Growth & beyond

a) The website is hosted via AWS EC2. The user can either: 1.) look up existing results for app privacy scores or 2.) upload information for a new app to be evaluated.

b) The website will be backed by a SQL RDS database. This database will store all apps that have been ranked and their corresponding scores. If a user searches existing rankings, they will be interacting with this database. Similarly, if a user uploads information for an app to be analyzed, the RDS database will be queried with the inputted information to see if the app has already been ranked. If so, the user will be directed to the ranking page for the app in question.

c) If the user instead uploads an app that has not yet been analyzed, then we will be using DocumentDB, a NoSQL database. This will be used for a few purposes. First, to store the training dataset for the NLP, and to store the information that we gather about the app (including app store reviews, privacy policies). Furthermore, the logs generated when we run dynamic code analysis are also stored in DocumentDB.

d) These two are not currently integrated in a visual UI component, but they will be as work on the project continues. The main UI will be focusing on user accessibility through two components: easy navigation and transparent information.

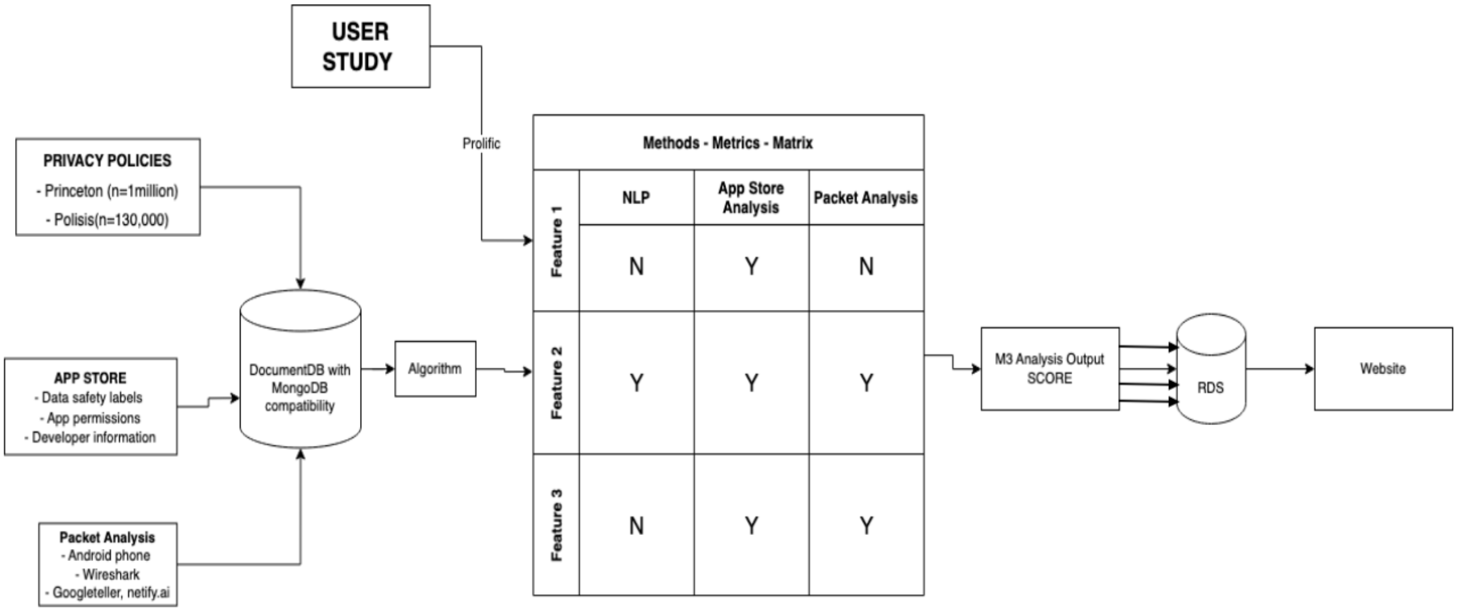
1. API components

   a) Bootstrap for CSS web page design.

II. Technical Specifications

A. Architecture/Systems Diagrams

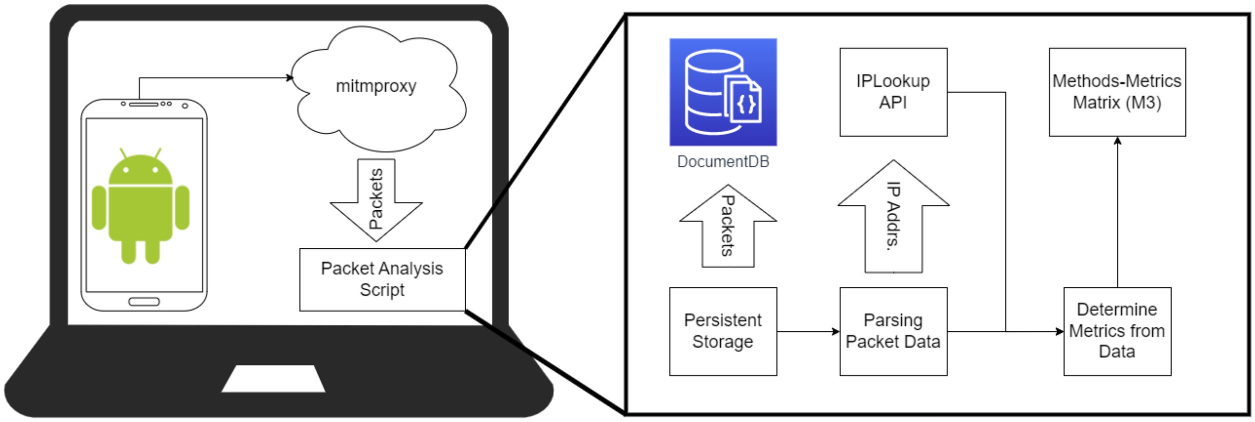
1. Overall Project Architecture:



Methods - Metrics - Matrix

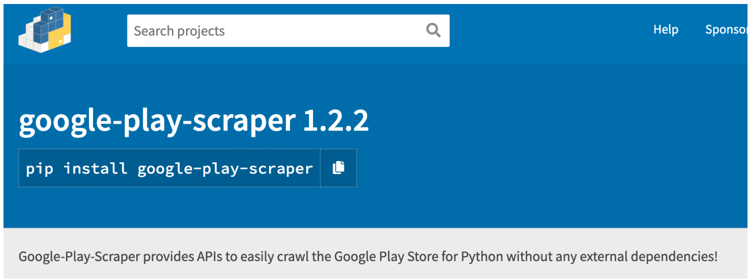
	NLP	App Store Analysis	Packet Analysis
Feature 1	N	Y	N
Feature 2	Y	Y	Y
Feature 3	N	Y	Y

2. Packet Analysis Architecture:



B. External APIs and Frameworks

1. For the natural language processing (NLP) that we are applying to the app privacy policies, we will be using some external APIs. To store the corpus of text that will be used as the training dataset, we will be using DocumentDB, a NoSQL database specifically designed for document storage for NLP. The datasets we will be using are the Princeton University dataset of privacy agreements and the Polisis dataset.
2. For the analysis itself, we will be using the open-source version of Polisis, which is designed for analysis and classification of privacy policies. The algorithm itself will be run on our server backend (i.e. at this time we do not intend for the NLP to be an API call especially due to the algorithmic component discussed below).
3. To analyze the network traffic on the apps we will be performing packet analysis. We will use an Android Virtual Device (AVD) to emulate a smartphone running each RHA, and an application called *mitmproxy* to intercept and decrypt the packets in transit. The packet data will be stored on our DocumentDB and we will write an algorithm to parse the information and compare the destination IP addresses to known third parties using APIs such as IP-API or ipstack.
4. The google-play-scraper API will be used to extract basic app store information and load it into the database for processing.



google-play-scraper 1.2.2

```
pip install google-play-scraper
```

Google-Play-Scraper provides APIs to easily crawl the Google Play Store for Python without any external dependencies!

C. Algorithms

1. The key algorithmic component of the NLP that we are using is the use of a new classification category for app privacy policies. This is because the current Polisis framework only supports general app privacy labels (e.g. shares user data first-party, etc.) We intend to classify policies with reproductive health-specific labels such as “tracks period cycles”. We hope to do so by introducing the Princeton University dataset for training purposes. During training, we will be minimizing the error of the NLP on these new classification categories using a training set of hand-annotated policies.
2. We also introduce an algorithmic component in the production of the final score itself. Based on the user study, we will have a series of features that customers have indicated are important (e.g. does not share with law enforcement, etc). For each of these questions, if any of the methods we use (NLP, dynamic analysis, app store labels) indicate a positive answer, then the feature is marked as a yes. The features are negative (i.e. an optimal app would answer no to each feature) so once a feature is determined to be “yes”, it is subtracted from the initial score of “10”. Thus, the lower the score, the less private the app itself is.