BSC APPRENTICESHIP DEGREE

DIGITAL TECHNOLOGY SOLUTIONS (DTS)

LEVEL 6

“Improving User Experience for Google Play Protect”

Final Year Work - Based Project

This report is submitted in partial fulfilment of the requirement for the degree of BSc Apprenticeship in Digital Technology Solutions by Maria Veronica Sonzini.

BSc Apprenticeship Degree in Digital & Technology Solutions

Software Engineer Pathway

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Veronica Sonzini

Abstract

“Consistency is one of the molecules of the Design DNA. Consistent design is intuitive design. It is highly useful and makes the world a better place. In short, usability and learnability improve when similar elements have consistent look and function in similar ways. When consistency is present in your design, people can transfer knowledge to new contexts and learn new things quickly without pain. This way they can focus on executing the task and not learning how the product UI works every time they switch the context.”

(Anton Nikolov, Design Principles).

Maintaining a consistent design (Fonts, sizes, buttons, labeling and similar) across Google products and systems from the same business area is of main importance. This way the user’s knowledge for one product can be reused in another. This helps eliminate a lot of the friction and provides great user experience. ***Updating all Google Play Protect’s UI and UX*** (by adding new UI components and new features to improve the user experience) is the most relevant achievement of this project, as it also was the cross team collaboration, and will be discussed in detail throughout this dissertation.

Acknowledgments

I would like to earnestly acknowledge the sincere efforts and valuable time given by Maya Tudor and Sparky, from the Google Apprenticeship team, and my teammates and Managers: Niko, Raissa, Rachael, Ioannis, Simon and Helene. Their valuable guidance and feedback has helped me in completing this Apprenticeship program.

Also, I would like to mention the support system and consideration of my husband Cris who has always been there for me.

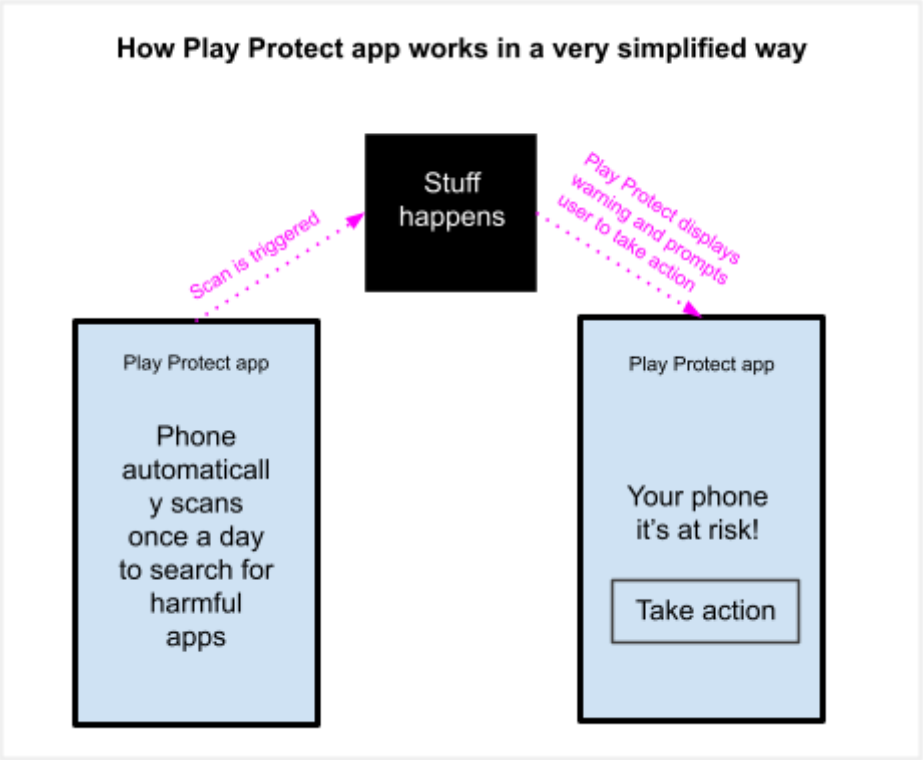
Last but not the least, my kids’ childminders Andrea and Cindy, with whom I wouldn’t have been able to take the time to work over the past years, and along these lines, thanks to Google that helped me pay for them.

Without them. I could never have completed this task.

Chapter 1

Introduction

It’s very simple: creating apps it’s hard. It’s even harder when these apps are massive. Can’t even start telling you how hard it is when these apps are not only massive, but they have sub apps inside. In companies such as the one I work for (Google) massive apps are split in smaller chunks and given to different teams to work on them.
When you have one, two, three or even more teams working on different parts of an app (see it as different teams working on creating pieces of a puzzle, that then need to go together with other team’s pieces, and create together a beautiful landscape painting) you start running into multiple complicated problems. It’s vital in these cases to work as a whole, even though you might be working with your team in the UK and another sibling team is working in California, or even Dubai! Maintaining coherence on the design of these puzzle pieces is what will give the user the feeling of unification, and perhaps they won’t even imagine it was all done by different teams working in different parts of the world.
Android’s Play Store is one of these massive apps I was talking about: it has many apps inside. Did you notice as you go into the Play Store on your Android phone or tablet, that you have the option not only to download apps, but also you can download and buy movies, books, games and even check out how your device is being protected against malware by Play Protect? Well, these are all apps that are part of the Play Store, and each belongs to a different team. My team works on Google Play Protect, and we focus on maintaining user’s devices safe from any harm through intentionally or unintentionally problematic applications. But let's not forget we are still part of a bigger picture! the picture of the Google Play Store.

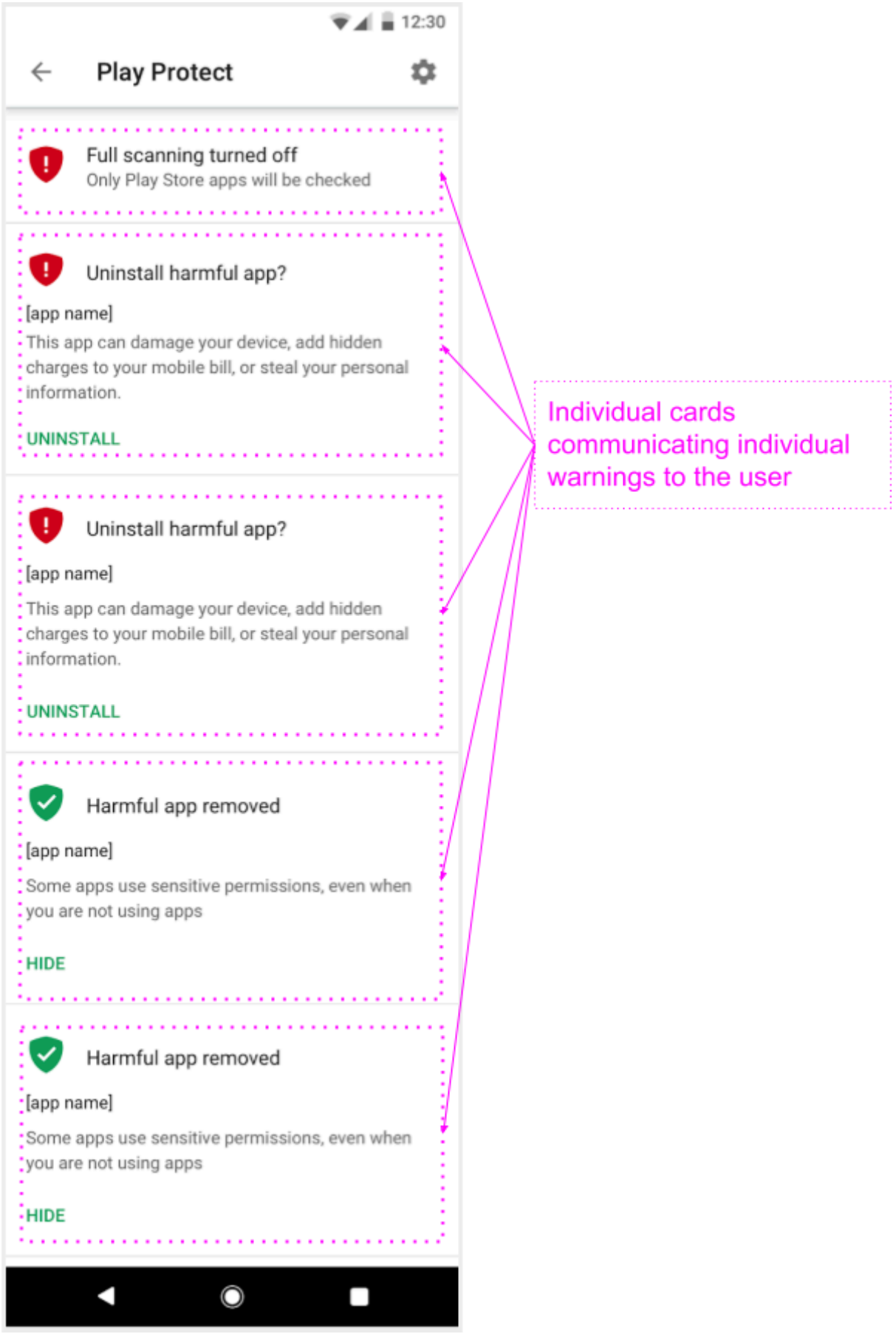


The idea on this drawing is to explain the main objective of Play Protect:

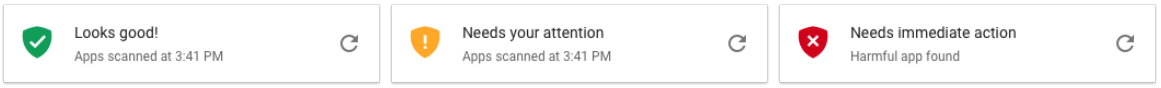
- **scan:** Once a day the phone triggers the can action, and checks one by one all apps installed on your phone against a list of harmful apps. If it finds one or multiple apps that are on the “red list” it would consider that your phone is at risk. There are other types of warnings, such as yellow ones, that could mean there is a potential risk. The green notification means that the phone is not at risk.
- **Display warnings or notifications:** If the scan found any harmful apps (or potentially harmful) it displays them on a list
- **Prompts user to take action:** a series of recommendations are given to the user on how to manage the security of their phone (uninstall app, or keep app under your risk, and others)

In 2018 the Play Store went through a massive visual reset, creating a fresher, more modern design language. The Play Store not only focused on the visual aspects of the app but also they went through a great componentization effort to improve consistency, efficiency and scalability.
For us, on the Play Protect team, this meant we needed to update our UI as well, to maintain the unified design language.

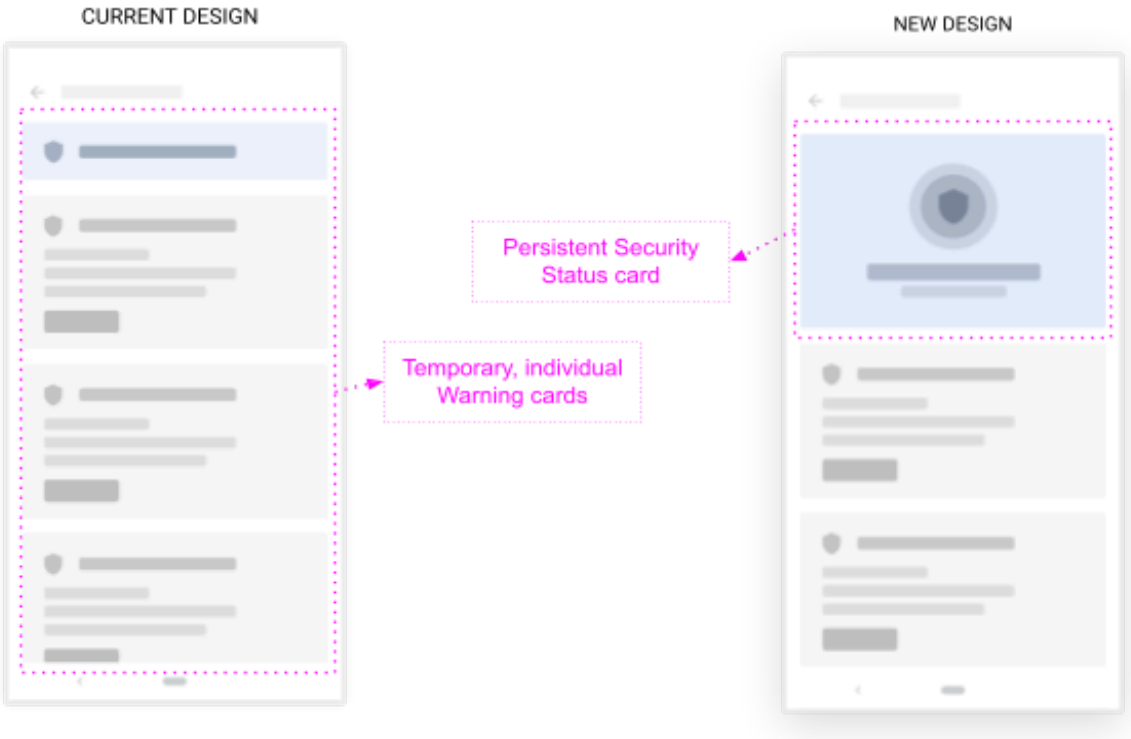
Another issue we wanted to tackle in the scope of this redesign project was that Play Protect Home displayed individual cards for each security issue found on the device, each accompanied by an icon which conveys the severity of the issue. After doing user research we discovered that by showing individual cards with warnings wasn’t enough for the user to clearly understand the overall security risk of their device.


12:30
Play Protect
Full scanning turned off
Only Play Store apps will be checked
Uninstall harmful app?
[app name]
This app can damage your device, add hidden charges to your mobile bill, or steal your personal information.
UNINSTALL
Uninstall harmful app?
[app name]
This app can damage your device, add hidden charges to your mobile bill, or steal your personal information.
UNINSTALL
Harmful app removed
[app name]
Some apps use sensitive permissions, even when you are not using apps
HIDE
Harmful app removed
[app name]
Some apps use sensitive permissions, even when you are not using apps
HIDE
Individual cards communicating individual warnings to the user

Different types of warnings



The solution we found for this problem was to additionally convey to the user a brief overview summarising all the security risks found on the device. This holistic overview will clearly communicate to the user through a single point of information whether they need to take any actions to protect their device and provide more context on the individual warning cards displayed in Play Protect Home.



The problem that this document addresses, and that I’ll be telling you all about on the next chapters of this dissertation, is:

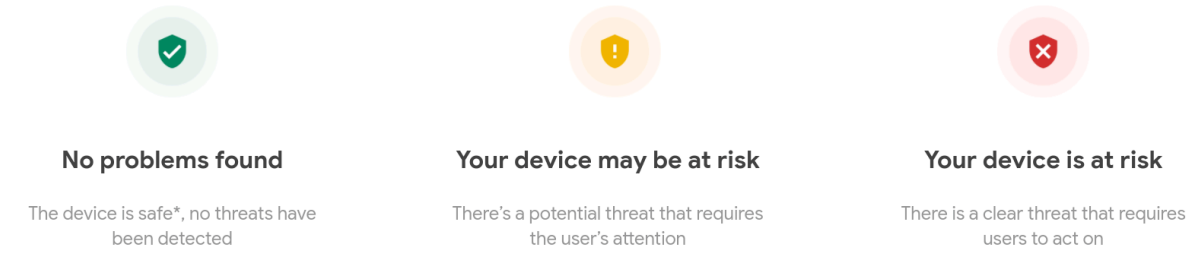
- Successfully update the UI of the Play Protect app and add new functionalities after the Play Store changed completely their design style and the way the UI is created by generating reusable UI components (that not only reduce the amount of redundant work, but also increase consistency and velocity of building said UI)

- Change the structure of code we used to build UI elements by implementing Play Store components
- Implement the Dark Mode, same as the Play Store
- Improve how users experience Google Play Protect app by:
    - Completely rethinking of the way Play Protect displays the security state of the device by introducing the new “Security State” concept:

        A device’s Security State will comprise the following:

        - Overall state -
        - “No problems found”: state when a user has no threats or recommendations. This is what the vast majority of users will always see.
        - “Your device might be at risk”: there’s a potential threat that requires the user’s attention. Used for Privacy warnings and unsafe apps.
        - “Your device is at risk”: there is a clear threat that requires users to act on. Used primarily for malware. Currently 0.35% of Android devices have malware on them.
        - Play Protect is off: used when full Play Protect consent has not been given.

    - Adding the new “Status card”: this new card will be a prominent, permanent (always visible and present) card at the top of the Google Play Protect Home screen that conveys the overall security state of the user’s device with a better, user friendly, straightforward design: A shield illustration in the card will convey, using one traffic light colour, the “overall state” of the device - Red for “Danger”, Yellow for “Warning” and Green for “Ok”. The overall state will be dictated by the most severe security warning, for example, if a device has a harmful app installed, no matter what the other risks are, the overall state will be “Danger”. The Status card will also display the Security State title, message and sometimes an action button through which a Play Protect Scan can be triggered.



Traffic light principles applied to a device’s security state

I honestly hope you find this interesting enough to read through the whole lot of chapters ahead. I won’t be taking anymore of your time, go ahead and dive into Chapter 2: Literary Review.

Chapter 2

Literature Review

In this chapter I would like to take the opportunity to share all the great resources I found while working on this project and that helped me with the production of my work. A mixture between internal, very detailed documentation, and external sources like tutorials, online articles and even YouTube videos.
I’m convinced that having internal and external points of views would give me the right unbiased perspective I need to carry this project.
For a matter of confidentiality I won’t be able to share the internal documentation I used for my work, but I will give a high level description of it.

Initially I had to properly understand core concepts such as:

- Componentization: how it was done, and how can I implement said components in my code
- Get familiarized with Play Store’s design system
- Understand best practices for writing good code

- Understand best practices for changing already existing code

Componentization

Crystal Bedel explains in a very interesting article called “Componentization (component based development) the benefits of creating components, and the history behind the concept of componentization. This helped me get started with my understanding of the project: “Component-based development offers a number of benefits. In the short-term, componentization enables software development teams to more easily collaborate. Reusing components that meet well-defined specifications also helps accelerate product development while increasing software reliability. As a result, teams can bring better quality software to market faster.”

A paper written by Kyung Su Hwang, Jian Feng Cui and Heung Seok Chae published by the IEEE called “An Automated Approach to Componentization of Java Source Code" focussed on the process of adapting source code to the newly component-based system architecture. This paper proposed an automated approach to migrating legacy software systems into component-based systems by using a tool called JCMT. Although this paper was 210 pages long and detailed at a level that was far superior for my understanding, I managed to find parts of it very useful. The approach I ended up taking for my legacy code migration did not involve either automating the process nor using the JCMT (JComp) framework as suggested by the authors of the paper. Although I did a thorough research about JComp framework I finally decided that the proposed technique involved a level of understanding and experience that I didn’t possess at the time, so my technique was to do it manually.

Several internal documents created by Google engineers about other projects that had to go through componentization efforts, gave me the understanding on how the component-based system migration was done in the company. I had the chance to compare techniques and follow guidelines on how this is done internally.
Such documents were:

- UI componentization, 24/10/2018, created by Google engineers from Play Store team
- Componentization: guide to creation, adoption and consistency, 25/2/2018, created by Google UX designers

Play Store’s design system

Firstly I wanted to get familiar with Google’s design system to get an overlook or high level understanding about the UI design. There are a great number of open source resources on Google’s Material design website, but in my case I decided to go for more specific internal documentation. Unfortunately I can’t share the details of the information I found internally, but I can say it was detailed and targeted specifying Play Store’s design language.
I gathered all the documentation regarding technical specifications (Visual redesign, 15/10/2018, created by Google UX designers and Visual designers from the Play Store team) about how the UI looked before, and how it was supposed to look after, which UI I needed to replace by the new components, and the specifications about name of components:

- Button components
- Colour components
- Text components
- Padding, spacing components

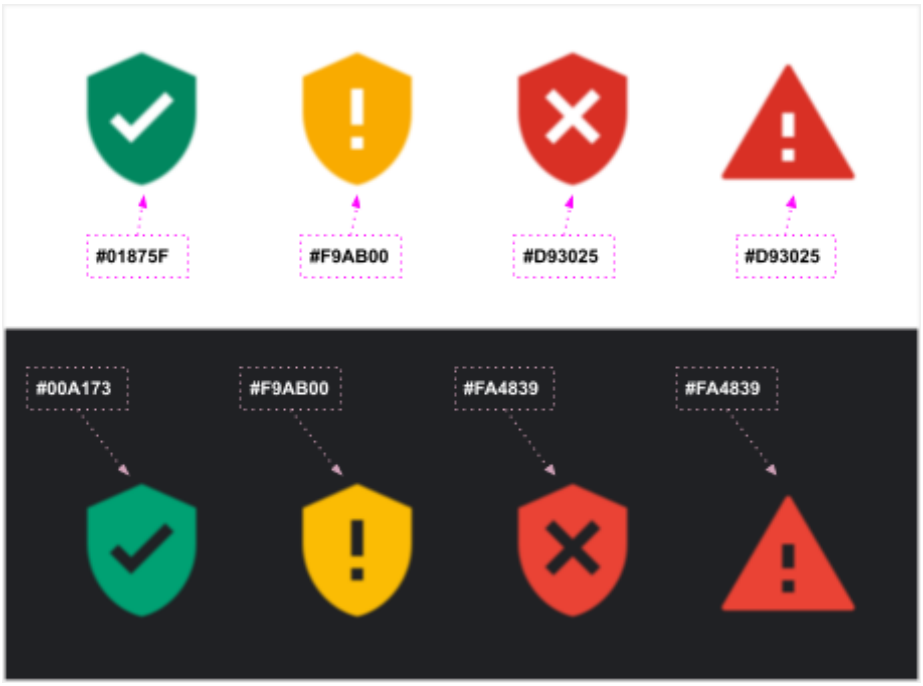
The documentation specified all technical specs for all devices: mobile, tablet and Chrome. It’s very important to notice that not all device types use the same components, because you need larger fonts and paddings, for instance, on a tablet than you need for a phone.

Among the documentation mentioned above I also had to understand how Dark Mode would be implemented on Play Protect, and for this another set of documents were provided by the visual designers of my team:

- Visual redesign & Dark Mode, September 2019, created by the Visual designer lead of Play Protect.

This document compared Light Mode and Dark Mode:

- Same colours with different shades to suit better the light or dark background
- Same assets with different shades
- Play Colour Palette
- Store components
- Play Store Dark Mode deck



Clean code

Clean Code is one of the books from the “Robert C. Martin Series”, which focuses on coding best practices. I started reading this book before I was assigned this project, but I did go back to it at the time to read over some of the points that were relevant for my work. This book taught me about how code duplication is one of the “mortal sins” of software development.

As the book states “duplication is one of the most important rules, and you should take it very seriously. Dave Thomas and Andy Hunt[1] called it the DRY method (Don’t Repeat

[1] Dave Thomas and Andy Hunt: Writers of “The Pragmatic Programmer”. A book about computer programming and software engineering, published in 1999, creators of the DRY (Don’t Repeat Yourself) principle aimed at reducing repetition in Software development.

Yourself). Kent Beck[2] made it one of the core principles of Extreme Programming and called it : ‘Once, and only once. Ron Jeffries[3] ranks this rule second, just below getting all the tests to pass.” Although the componentization effort done by Play Store is mostly related with unifying the design language and removing repeated UI implementations, it’s also related to the concept of Duplication explained in Clean Code.

Working Effectively With Legacy Code

“Working Effectively with Legacy Code” is another book from the “Robert C. Martin Series” [4]written by Michael C. Feathers that I found helpful during my time working on this project, as it covers all about “testing”:

- Unit testing
- High level testing
- Test coverage

It also has an interesting view about improving software design quality. “Design improvement is a different kind of software change. When we want to alter software’s structure to make it more maintainable, generally we want to keep its behaviour intact also. When we drop behaviour in that process, we often call that a bug. One of the main reasons why many programmers don’t attempt to improve design often is because it’s relatively easy to lose behaviour or create bad behaviour in the process of doing it.” This is absolutely right and why I needed to be extremely careful about the changes I was making while migrating to a component-based design system. I had to make sure every step of the way, the functionality was the exact same.

This book’s main characteristic is that it’s very practical, showing steps to do certain things and showing use cases applied to those things. For example when it explains “characterization tests” it does it in a very understandable way. “Many characterization tests look like ‘sunny day’ tests. They don’t test many special conditions; they just verify that particular behaviours are present. From their presence, we can infer that refactorings that we’ve done to move or extract code have preserved behaviour.”

---

[2] Kent Beck: ent Beck is an American software engineer and the creator of extreme programming, a software development methodology that eschews rigid formal specification for a collaborative and iterative design process.

[3] Ron Jeffries: one of the three founders of the Extreme Programming software development methodology circa 1996, along with Kent Beck and Ward Cunningham. He was from 1996, an XP coach on the Chrysler Comprehensive Compensation System project, which was where XP was invented.

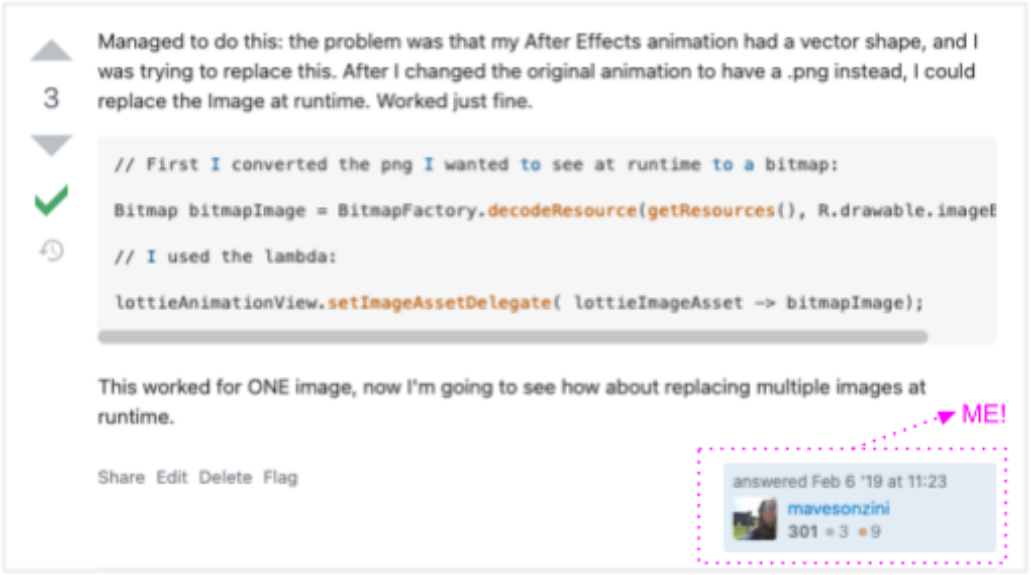
[4] Robert C. Martin: Colloquially called “Uncle Bob” is an American software engineer, instructor, and best-selling author. He is most recognized for developing many software design principles and for being a founder of the influential Agile Manifesto. Robert C. Martin Series Series

StackOverflow

Stackoverflow is sort of like a social media website where developers (mostly) upload questions about software and design (among other tech subjects) and other people can reply and give feedback or answer their concerns. Through the years I’ve done software development, this resource was definitely the one I used the most.
For this particular project I kept coming back to this website for help with best practices for writing certain pieces of code, such as:

- How to create better xml views for java
- How to replace assets on my java project
- How to create animations using java
- How to add an animation using Lottie library, etc

During this time I used Stackoverflow.com so much I ended up winning badges and many points!



Other online resources

I also worked with other online resources, by searching on Google. For instance, I wanted to get more familiar with layouts in Java, so I looked for tutorials on how to make them work as I intended. The website developers.android was one I checked very often, as the content there is written by the very same people who created the Android tools or frameworks I wanted to use. I relied on this a lot.
When I researched for layouts I found all I needed to understand right there.
Let me say that layouts were a very important and big part of my project, since most changes I had to do were in reference to the views. I had to create new layouts, with new structures, to add all UI components required for the visual redesign. Here are some examples of my work related to layouts:


Everything in red is what I removed from the file
Everything in green is what I added to the file


Code removed

Chapter 3

Requirements

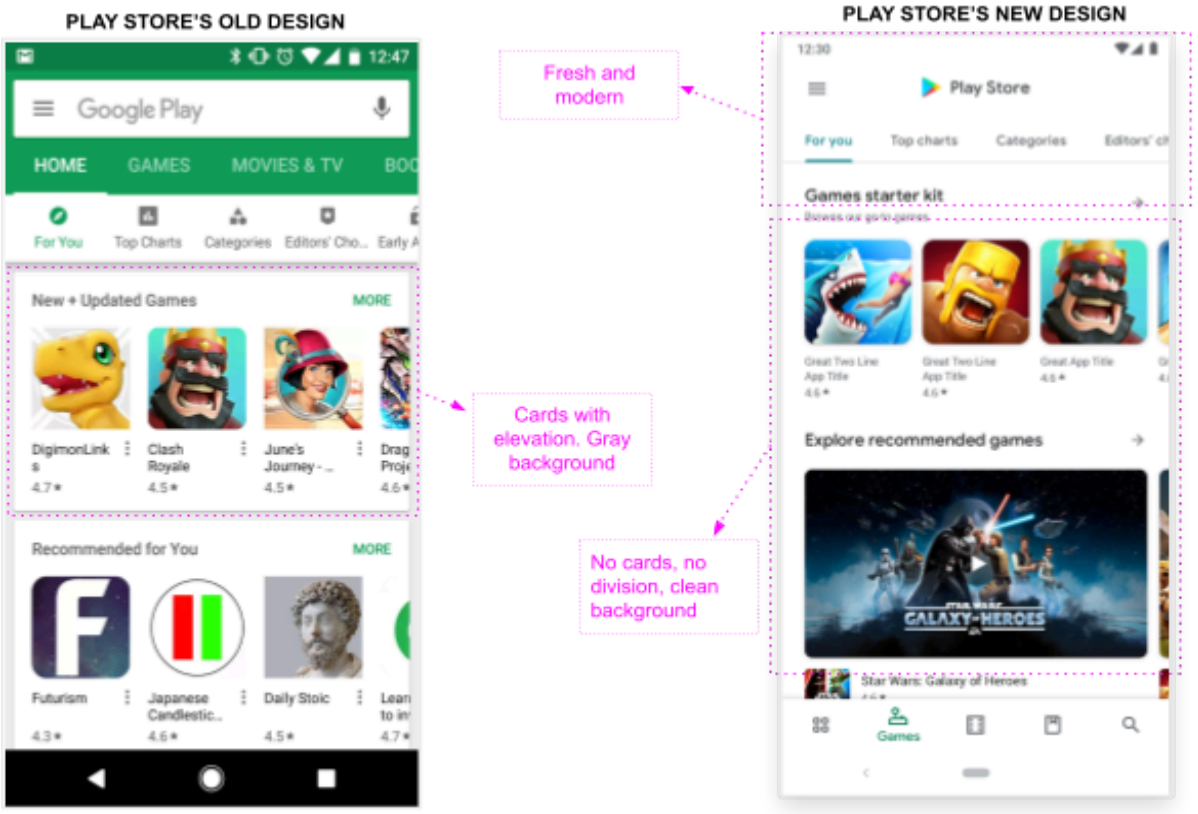
In this chapter I will share the project’s objective and its requirements in more detail, as well as breakdown the problem into manageable steps.

The Problem

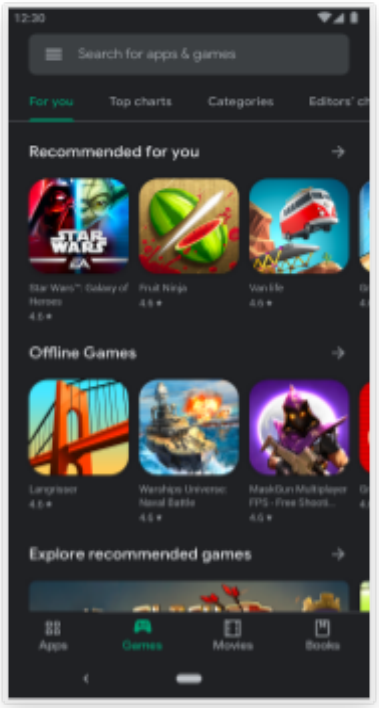
As I previously mentioned in the introduction, the Play Store has a large amount of UI code that is not shared among similar pieces, that is, the smaller apps that conform the whole Play Store (one of them being Android Play Protect, the app my team owns). This huge codebase consists of hundreds of views and thousand layouts. There are many of these views and layouts that are duplicating code (red flag here, folks!) in slightly different ways. This was not done out of malice, but for simplicity during development. Unfortunately, all those minor decisions have led to an extremely large codebase which is not easily

maintainable. This makes it more and more difficult to perform visual updates, and at the same time new feature development is becoming increasingly slow as the same UI pieces have to be reimplemented each time with minor changes. In other words, for any play store wide visual redesign, the engineering effort has become astronomically large, and will only get worse as more and more features are developed.

You may ask yourself, even after all the explanation I’ve been doing so far, what’s the reason any visual redesign is increasingly difficult. And the reason is because every view has implemented their own version of a UI spec on their own (padding, colours, etc - like the “title” example). Therefore, whenever you want to change the design language for a new visual styling, engineers would have to go to every single view and modify that piece (bad, bad practice!). A simple request such as “for all cards, change the padding around them” requires changes to many places since there are many different ways to create a card in the UI, and as I warned before, this will become a nightmare!



PLAY STORE'S NEW DARK MODE



Now that Play Protect is fully integrated into the Play Store, we want our users to experience a cohesive narrative and visual language across the Play Store. This means that Play Protect needs to adopt the principles and styling of the Play Store design system in order to feel like an integrated product.

Why does this sound like a problem? Well, because integrating the Play Store design using their UI components will be so positive in the feature, **none of our components are yet aligned with the Play Store’s, therefore some visual elements are not looking as they should. I will need to adapt the existing code to implement the requested Visual Redesign.**

The new design introduced by Play Store includes things such as:

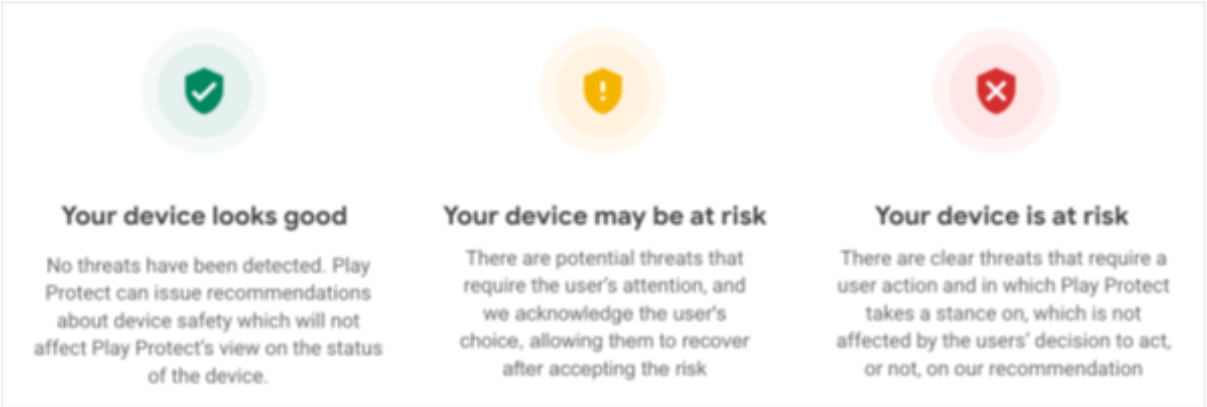
- Black mode
- Design compliant with Google Material 2 (Google’s main design language)
- Change of typography
- Global margins
- Semantic padding
- Colors
- Buttons

- Cluster headers
- Chips
- Full bleed
- Rounded corners + elevation

Some of the expected benefits:

- Provide Play Store users with a more consistent and usable experience
- Increase our brand perception
- Make it easier for future development
- Users perceived it to be "cleaner, slicker, nicer, grown up and more sophisticated"
- Improved future UX & engineering velocity

At the same time we want to address the problem of properly communicating the user the security status of their device. **Currently, we are prioritizing individual actions which do not communicate the overall risk that the user may be at any given point in time.** In other words, we show the users different cards displaying warnings, but this doesn’t clearly show to the user the level of risk their devices are. The approach we took to solve this particular issue was to create a “Security Status” to display the overall security status of the device.



Objectives and requirements

- Improve the integration of Play Protect’s UX and UI to adopt the Play Store’s new visual style
- Update the UI in all the different cards in Google Play Protect home, incorporating new predefined UI components.

- Implement new UI changes specific for Play Protect app
- Create new Status card
- Implement the new “traffic light” system for warnings.
- Complete the project during Q3 of 2019. This will give me enough time to complete all the requirements, to get everything tested and approved to be submitted and released to the users.

Breaking down the problem into stages:

1. A first visual redesign, to integrate our product with the Play Store’s UX and visual identity by implementing the new Play Store’s UI components. This stage will change the structure of code used to build UI elements and combine them. Testing is carried out throughout the whole stage: unit tests are added to make sure none of the functionality is lost. QA & testing specialists are involved in each step of the way to be certain there are no bugs or problems I might have missed. UX and Visual designers feedback is also provided after each change. Code reviews are performed by Senior engineers of the team after every change before submitting the code.
    - Update UI in Play Protect by using Play components
        - Migrate fonts to use Play components
        - Migrate all buttons to use Play components
        - Migrate all container paddings to use Play components: mobile, landscape mode, tablet, chromebook
        - Update all dividers and internal paddings
        - Migrate all cluster headers and body of text to use Play component
        - Replace old icons by new ones
        - Testing is done simultaneously with every change
        - Code review
    - Visual redesign of Play Protect app
        - Adapt existing layouts where possible, and create new layouts if needed
            - Create new layouts for new cards:
                - “Status” card
                - “Recently scanned apps” card
                - “Learn more” card
                - “Scan in progress” card

2. A following stage including some other UI changes (specific to our app design) and new features and functionality. This would be: changes in the way our app displays warnings, notifications and confirmations. Same as the previous stage, testing is done with every change introduced. Unit tests, QA and UX feedback all involved.
    - Flow for re-enabling applications that have been disabled
        - Create UI for new user warnings
        - Create UI for new user confirmation dialogs
        - Creation of unit tests to cover all changes
        - Code reviews
    - Implement new ‘traffic light’ top-level system for Play protect
        - UI:
            - Add all new assets provided by Visual designers
            - Create persistent top-level status icon
            - Redesign of look-and-feel for status icon
        - Functionality:
            - Reflect the phone security status on the “security card”:
                - Create all the logic to show the traffic light system
                    - check on the phone status and see what warnings there are. Based on that, define the status ‘colour’ to be displayed:

Show Red warning UI	Show Yellow warning UI	Show Green warning UI
• Mix of red warnings • Mix of red & yellow warnings • Mix of red warnings & green notifications	• Mix of yellow warnings • Mix of yellow warnings & green notifications	• Mix of green notifications

            - Create the “scan” functionality: when the user clicks on the “scan” button, the scan functionality will be triggered.
            - (Potentially) animations of status icon transitions
        - Creation of unit tests to cover all changes
        - Code reviews

The project is scoped so that it can be finished within the expected time frame. The changes will be periodically submitted and released, following an Agile methodology.

Chapter 4

Specification & Design

This project has been split into two different phases as mentioned in the previous chapter. The first phase is related to the implementation of new UI components created by the Play Store. This part of the project involves

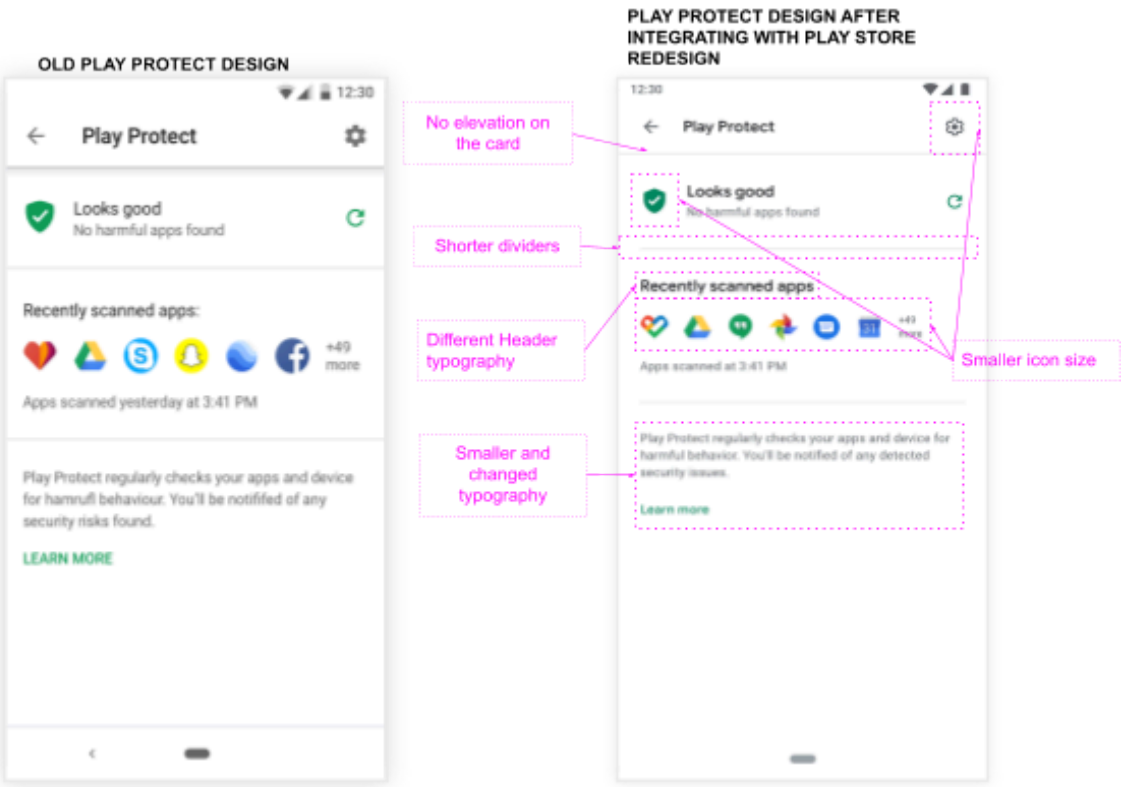
- Replacing assets such as icons by new ones corresponding to the new Play Store redesign guidelines
- Replacing all paddings, buttons, dividers and fonts by new components with the new Play Store design.

And the second part of the project involves a redesign of the Play Protect user experience

- Improve for the users the way Play Protect communicates the whole security status of the device by displaying a traffic light system
- Manual scan trigger

Integration with the Play Store & component migration

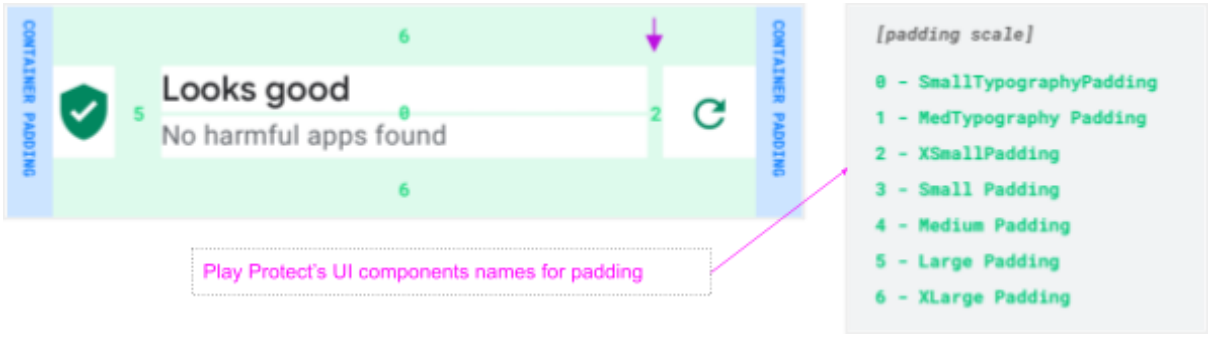
To kickstart the project, a series of guidelines and design decks were provided by the Visual designers and UX designers of the team, with all tech specifications.
This technical specifications included:

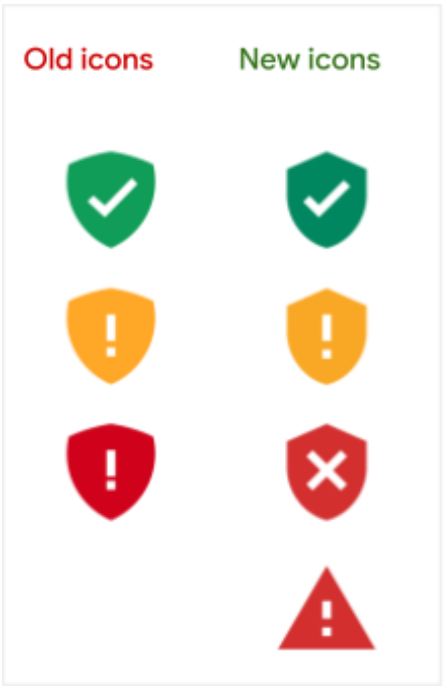


The documentation provided was very detailed and included all new component’s names for all UI components. The UX design team worked very thoroughly and left nothing up to chance.

The design specifications were given for:

- Paddings
- Typography
- Icons: new assets, colour specs for light & dark mode
- Size of containers
- Type of buttons: Colour filled buttons, outlined buttons, no outline buttons, etc
- Specs for:
    - Mobile
    - Tablet
    - Chromebook



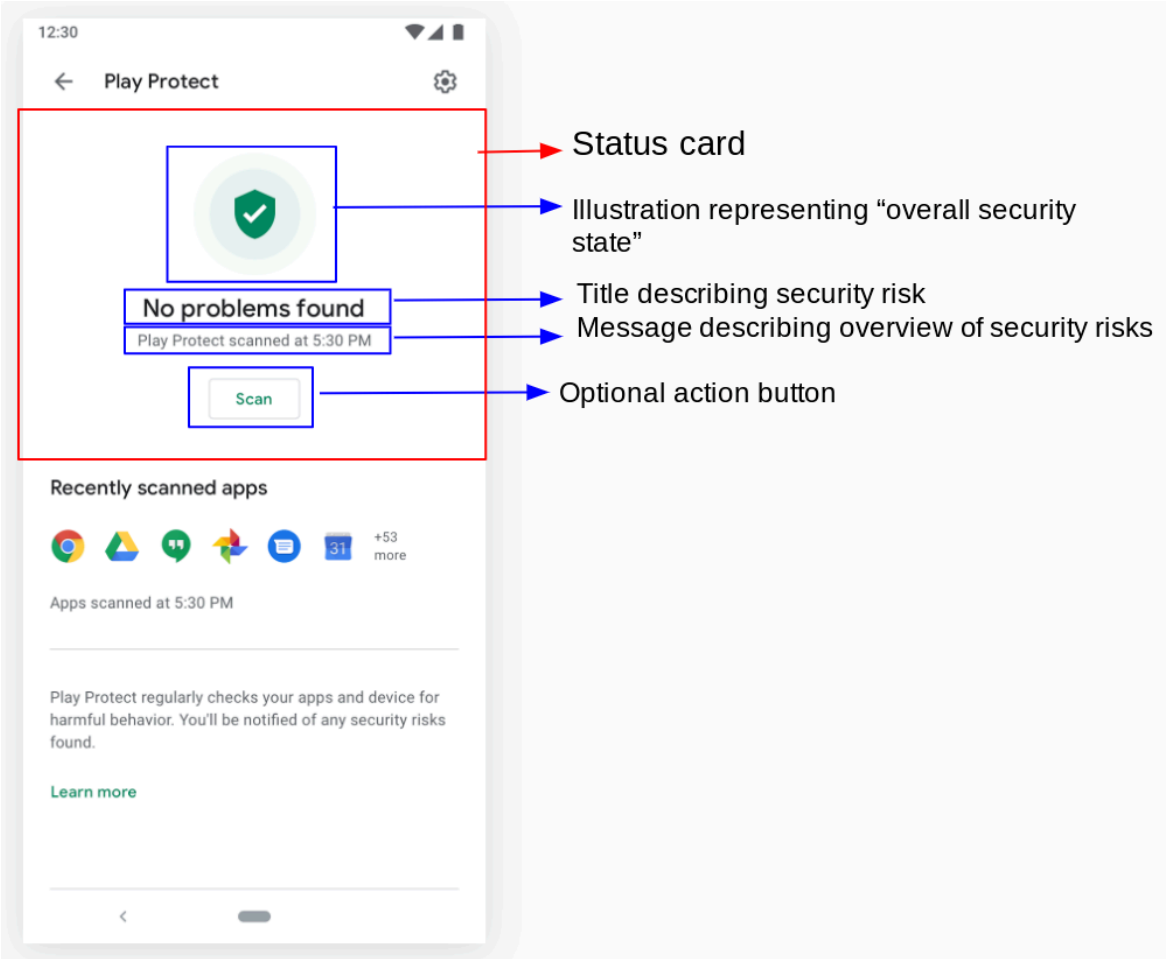


New features & functionalities

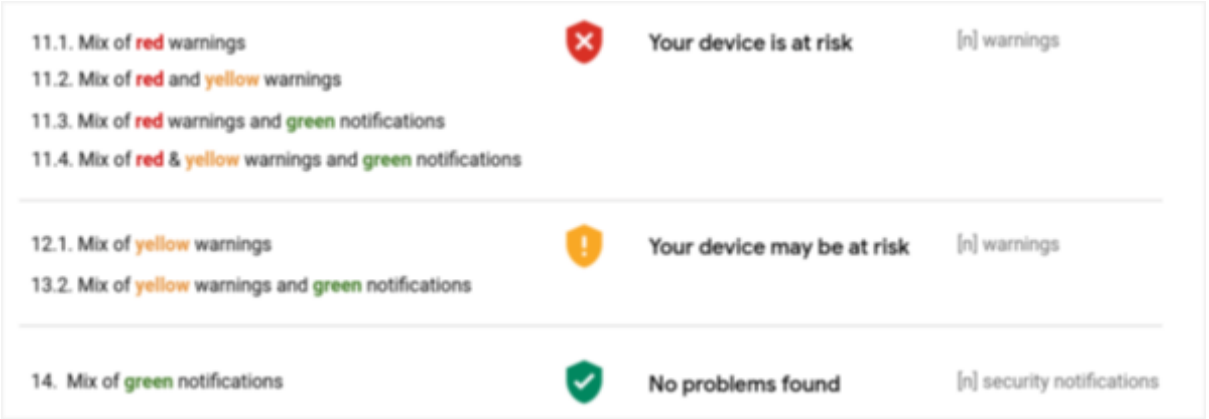
This section of the project involves fundamental changes, not only to the UI but also related to the app’s functionality.

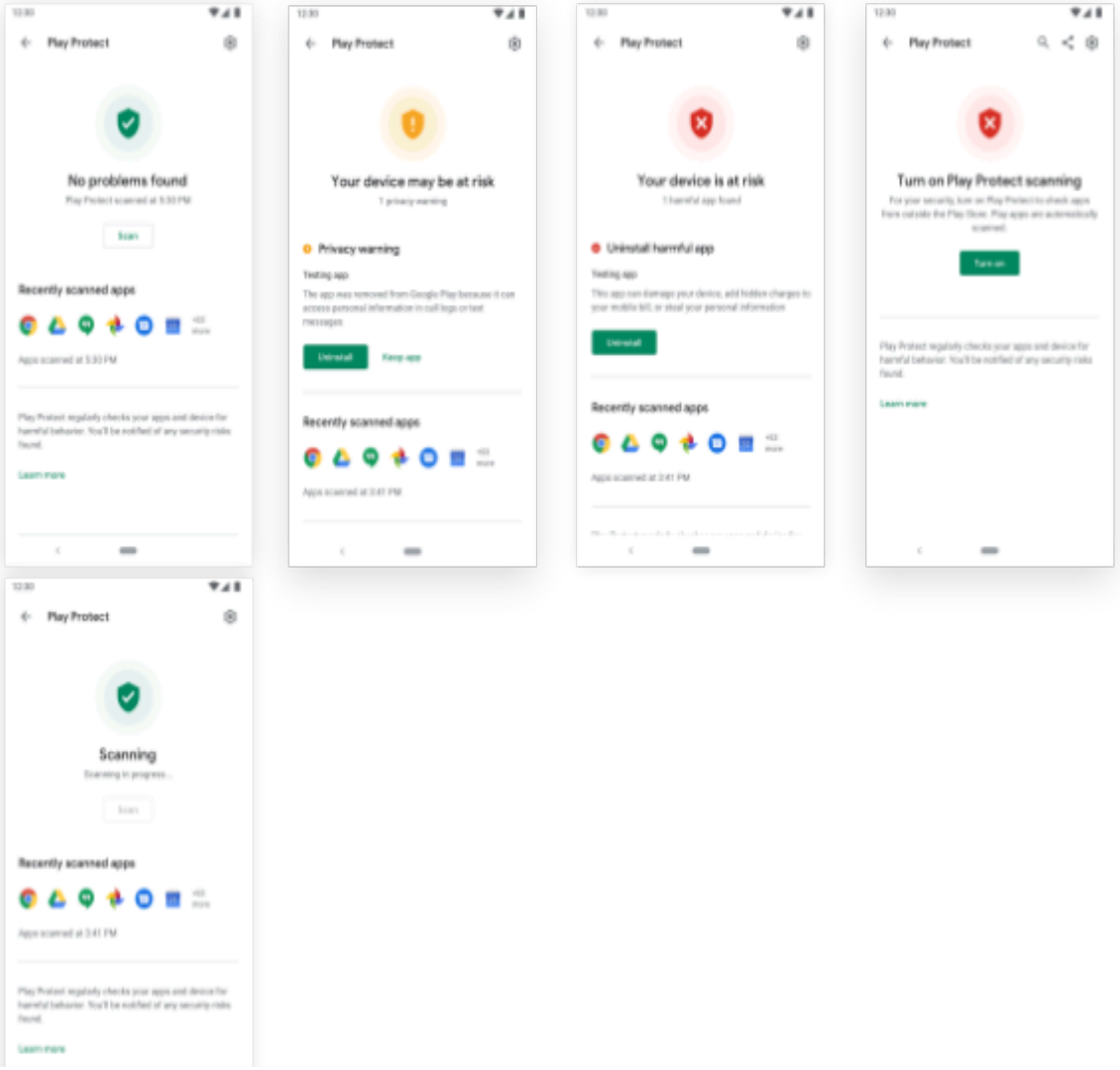
Specifications

- Creation of the Status card’s UI
- Add new functionality to the status card:
    - manual trigger for the scan function
    - Display of the device’s security state



While the overall state will be represented through the 3 traffic light colours, there are many combinations of security risks that can be present on a device, which leads to many possible Security States. In addition to the Status cards which display the Security States, we will additionally have one Status card indicating that an app scan is in progress



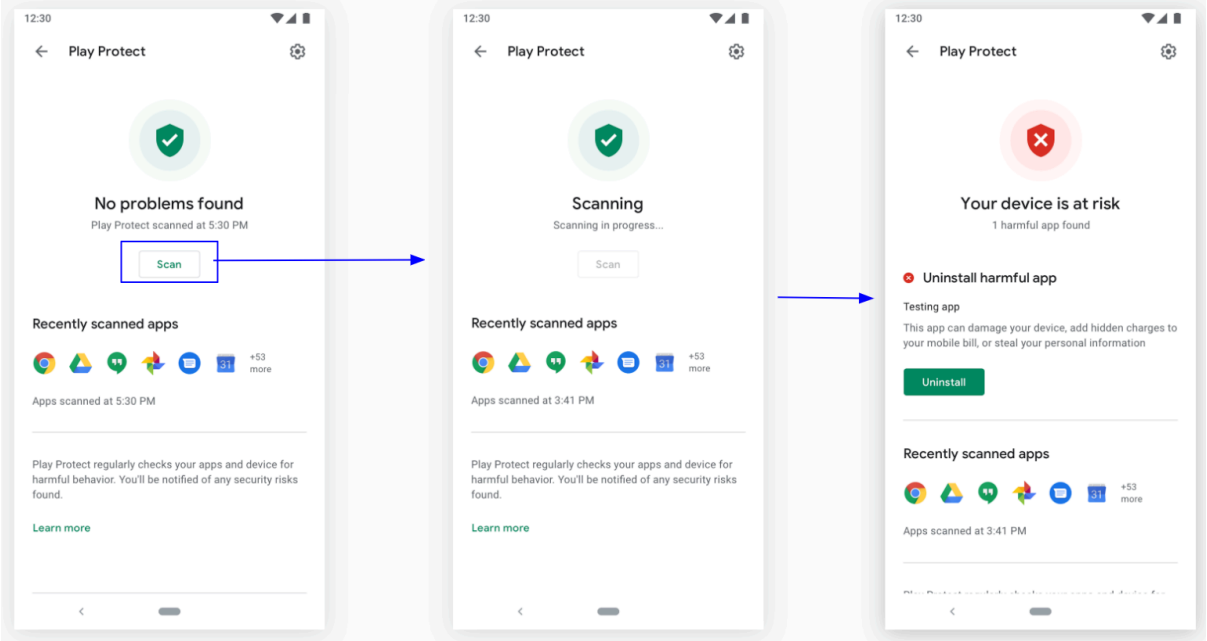


Card Interactions and Scanning

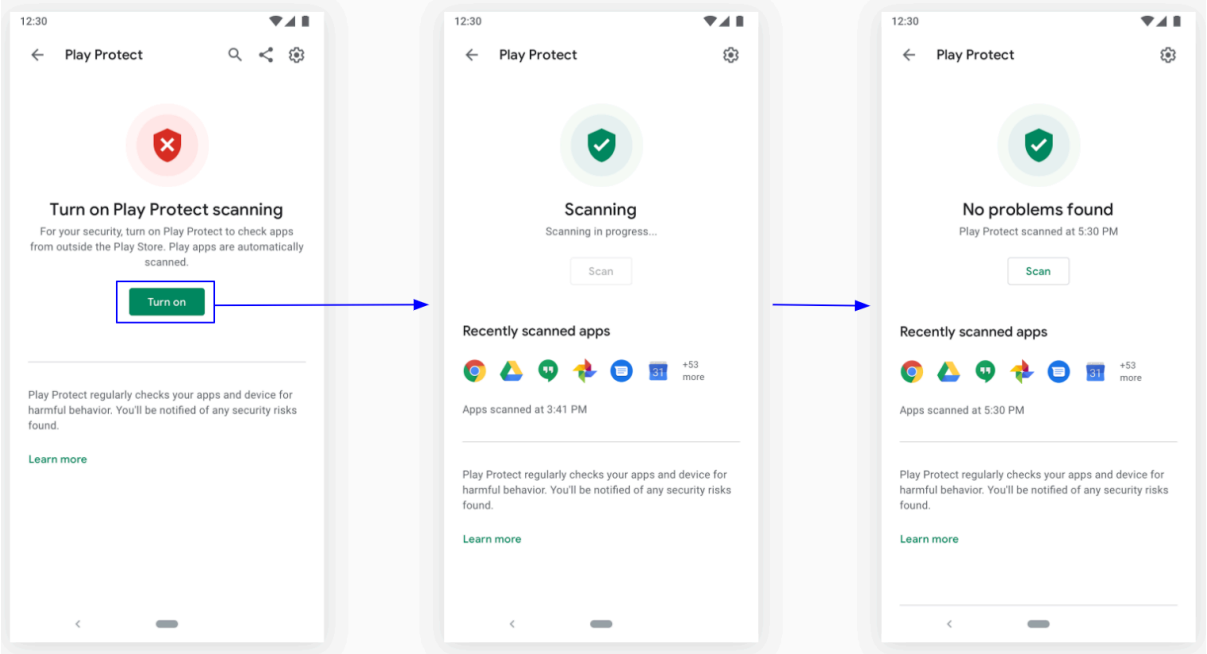
The Status card may additionally allow the user to perform an action through a call-to-action (CTA) button at the bottom of the card. The button will be present in the following cards:

- All status cards with a green illustration will have a “Scan” button. Clicking on the “Scan” button will trigger a Play Protect scan and display the Scanning-in-progress card.
- The state where Play Protect is off will have a “Turn on” button, clicking on which will turn on Play Protect app scanning, trigger a scan and display the

Scanning-in-progress card. When the scan is completed, the Status card will update itself to reflect the latest security state of the device.



Perform an app scan from Status card

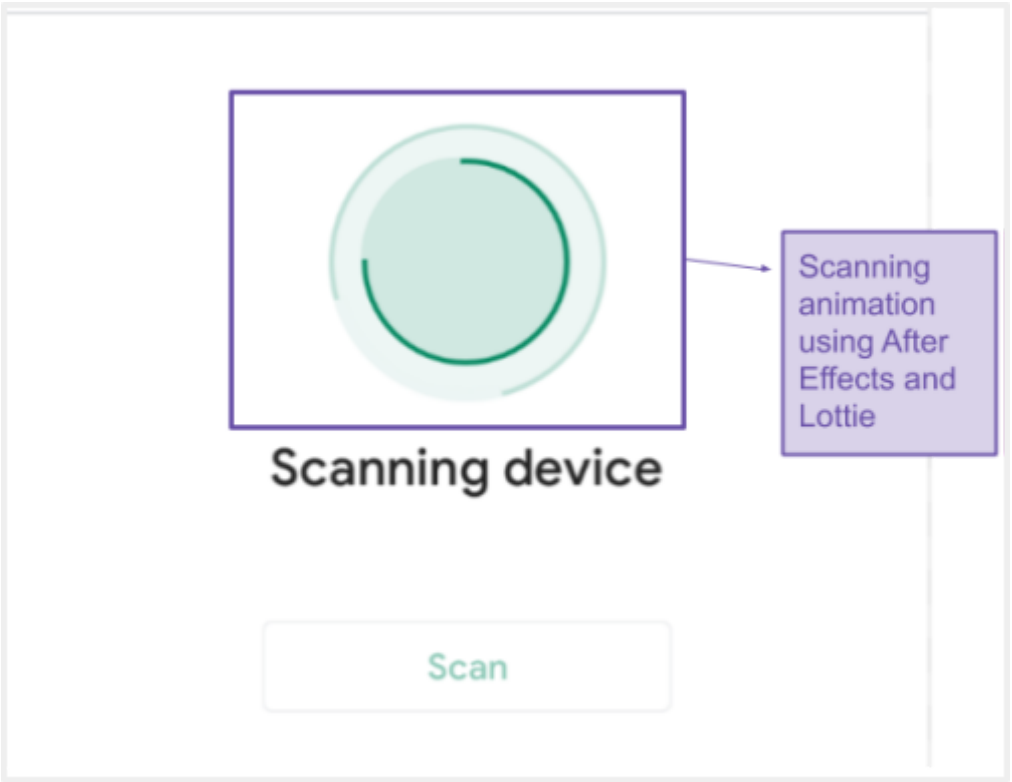


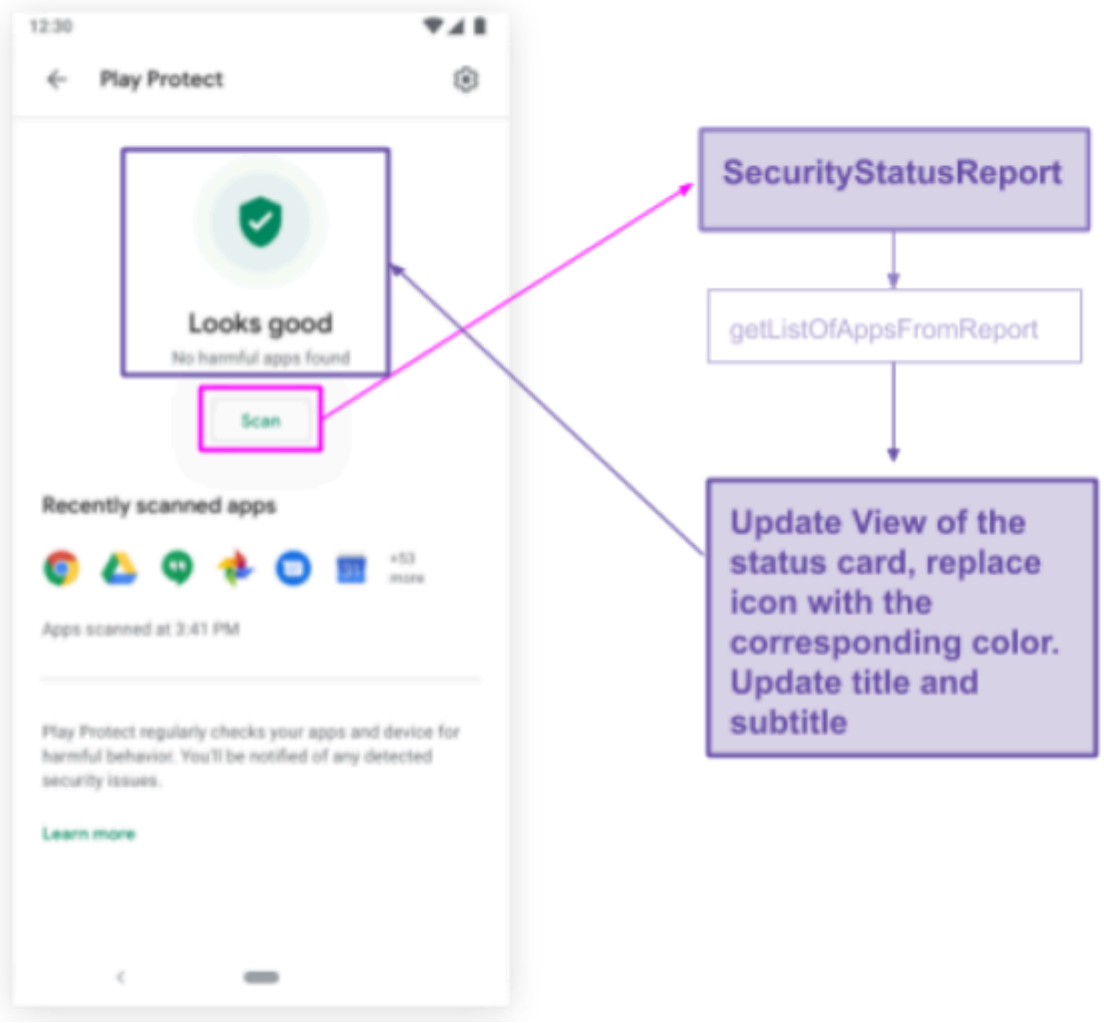
Turn on app scanning from the Status card

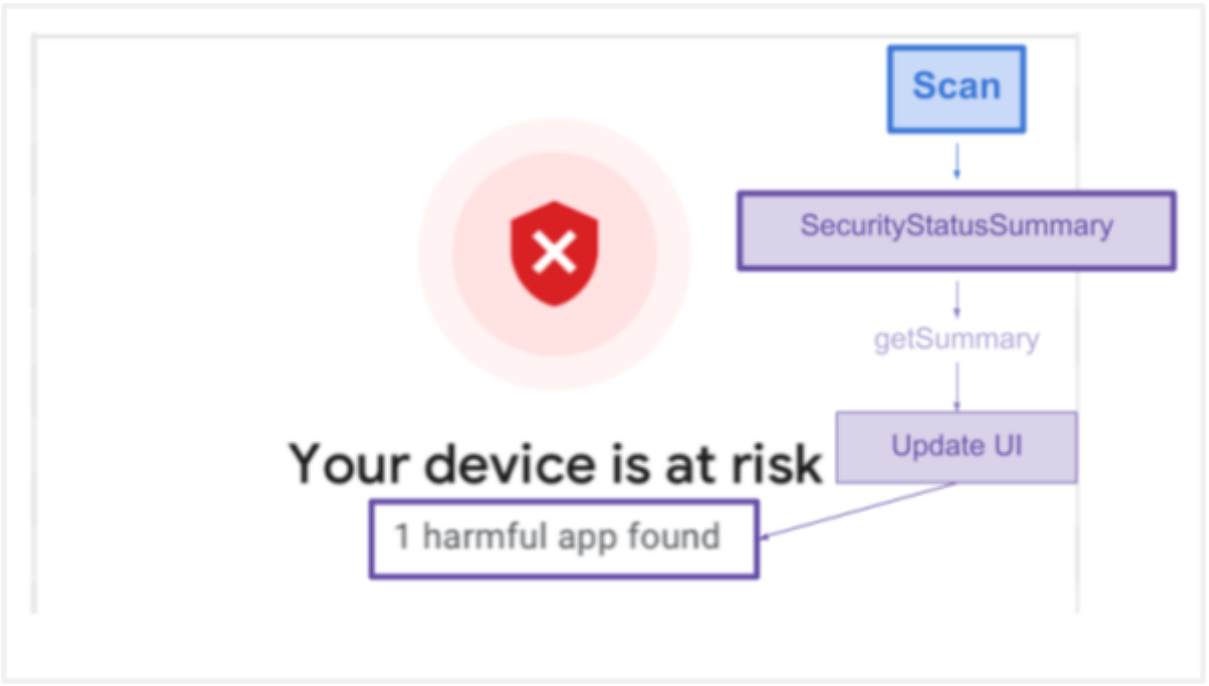
Shield color change

When a harmful app is installed, the overall state of Play Protect will be red. If the harmful app is uninstalled, then the state of Play Protect can continue to be red if there are still any harmful apps installed, or it can be either yellow -if any warnings left- or green if it looks good.
When an app is uninstalled, it needs to notify the view, so that the shield can be updated to a different color.

Apps scanning in Play Protect takes place in the background, so this is not visible for the user. For this reason there needs to be something that would indicate this is taking place, and not just a frozen screen. One way to show that something is happening in the background could be using a spinner.
With this project we wanted to take this one step forward and show an animation instead of a simple spinner. For this we used Lottie. Lottie is a mobile library for Android and iOS that parses Adobe After Effects animations, created using shape layers, exported as JSON data files with Bodymovin and renders them natively on mobile. It is a library that allows a motion designer to create smooth, scalable animations that retain consistency across multiple development environments without the need to spend countless hours/days/weeks endlessly tweaking animations with engineers.




12:30
Play Protect
Looks good
No harmful apps found
Scan
Recently scanned apps
Apps scanned at 3:41 PM
Play Protect regularly checks your apps and device for harmful behavior. You'll be notified of any detected security issues.
Learn more
SecurityStatusReport
getListOfAppsFromReport
Update View of the status card, replace icon with the corresponding color. Update title and subtitle


Scan
SecurityStatusSummary
getSummary
Update UI
Your device is at risk
1 harmful app found

Design Overview

Steps involved to build the Status Card:

- Add a new Controller, View, ViewBindable as well as xml layout for the Status Card.
- Populate Status card almost entirely from the “Security Status Summary”. Since the existing Summary does not cover all the desired Security States, a new method of Summary creation will be added, which will incorporate all new strings.
- Add functionality for Card interactions for “Scan” and “Turn on” as well as error-handling snack bars for when the intended actions fail.
- Add support for the Scanning-in-Progress card in the Status card Controller. We will use the Lottie animation library to display an animation when the scan is in progress, either by clicking on the “Scan” or “Turn on” buttons.

Chapter 5

Implementation & Testing

Play Store component implementation

A large number of tickets were created to address all changes in the UI. Each ticket was resolved on a separate change list that had to be submitted for pair review and approval. A total of 10 change lists were submitted for the implementation of all Play Store’s visual redesign.

NOTE: For confidentiality reasons code snippets ARE NOT Google’s real code. Code snippets had been edited to not reflect real implementation (real experiement’s name, class names, architecture, etc)

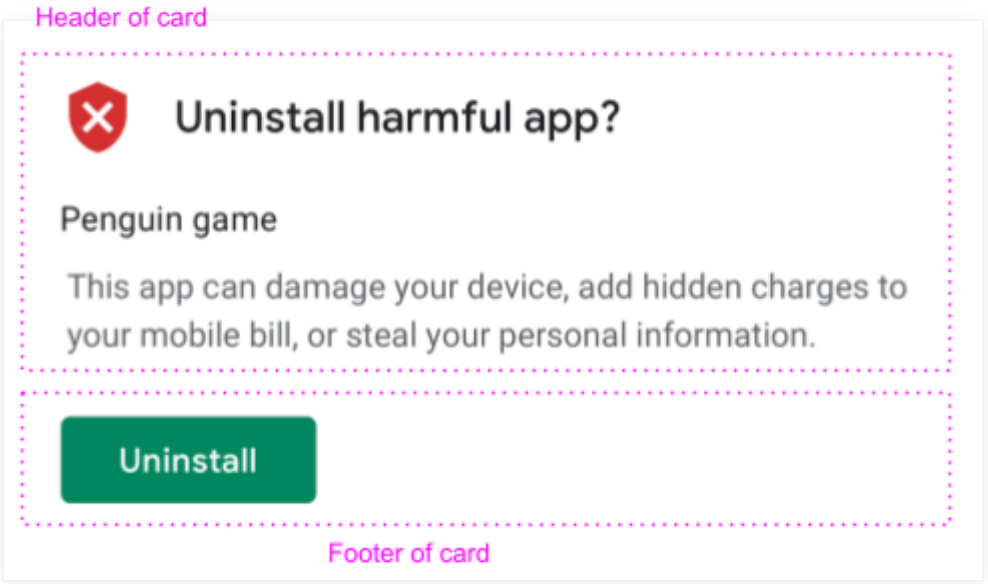
Changes to Play Protect Home Cards

The changes will be aesthetic only. We will

- Add new assets for the new styled icons in all Play Protect Home warning cards in
- Dynamically override icon/title padding and icon size to be smaller in the cards.

Adapting layouts

- Header of card
- Footer of card
- “Learn more” card
- “Scan status” card



FOOTER LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.CardFooterView
    xmlns:android="http://schemas.android.com/android"
    android:id="@+id/card_footer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="@dimen/protect_card_default_margin"
    android:paddingStart="@dimen/protect_card_default_margin"
    android:paddingEnd="@dimen/protect_card_default_margin"
    android:orientation="horizontal">
  <!-- Clicking this button performs the action recommended by the card. -->

  <com.google.android.LinkButtonView
      android:id="@+id/card_footer_confirm_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="@dimen/card_button_margin_top_bottom"
      android:layout_marginBottom="@dimen/card_button_margin_top_bottom"
      android:layout_marginEnd="@dimen/footer_button_margin_end"/>

  <com.google.androi.LinkButtonView
      android:id="@+id/card_footer_dismiss_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="@dimen/card_button_margin_top_bottom"
      android:layout_marginBottom="@dimen/card_button_margin_top_bottom"/>
</com.google.android.CardFooterView>
```

UI components

Play Protect’s new functionalities

An even larger number of change lists were submitted for this part of the project.

When is the status of Play Protect updated?

- During daily scan
- During manual scan triggered by the user from Play Protect home
- When an app is uninstalled from Play Protect home
- Some other scenarios - for the full list, see the “Triggers for updating the data” section.

Status card’s Controllers

A new controller (**NewStatusController**) will have the responsibility of showing the Status card in Play Protect’s Home. The Status card should show the most recent Security State of the device, unless there is a scan ongoing in which case it should display the ‘scanning card’ which will contain the scanning animation.
The newly added **NewStatusController** will hold

1. securityStatus: the most recent security status summary
2. isScanning: whether a scan is ongoing in a temporary controller state.

Depending on this state, the controller will decide which card to show - the Security Status card or the Scanning card. The controller state will be updated when either a scan is triggered or finishes, or a new update is received.
The Status card will be populated from the **NewStatusController** which contains fields such as

- overallSecurityState()
- title()
- body()

The scanning card will be populated directly from the **NewStatusController**.

Controller’s details:

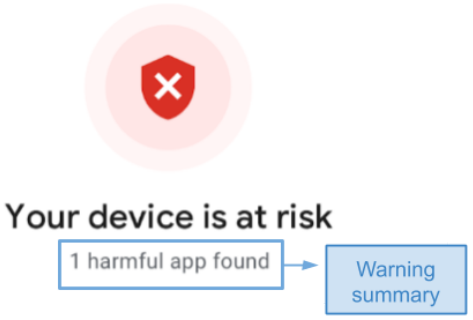
- Create a **new controller** for the Status card:
    - This controller will create all the different states of the Status card, getting the information from **SecurityStatus**.

- Make the subtitle display the number of warnings (summary of warnings) → This will come from **SecurityStatus.**
- Make the shield match the overall state of Play Protect’s security state (traffic light system) → This will come from **SecurityStatus**.
- Update controller:
    - **OldStatusController.java** is creating LGTM, Play Protect off and scanning cards at the moment. This controller needs to be unlinked from the old layout and add the new status layout instead, because these states will only be displayed in the new status card. These changes will all be carried out under an experiment flag, so the old design will still be supported when the flag is off.
- **Play Protect is Off:**
    - Previously, for changing statuses in the card, there was a **WarningLevel** that would get assigned for each app, and based on it the card would draw the right UI. For the new status card this is not going to be the case anymore. We are going to be using the **OverallStatus** from **SecurityStatus.java.**
    - **Why SecurityStatus? :** It’s a summary from the report that provides all we need to populate the status card’s UI: title, body, icon type, overall state of the device. This prevents us from having to re-do all the logic to display the right UI.

- **Re- use security status DANGER or create new status Play Protect OFF discussion:** We could potentially use the DANGER status to display the right UI when Play Protect is off, since the device is supposed to be in danger. What could be tricky about this is that even though both status (harmful app picked by Play Protect or Play Protect being turned off) are **RED**, they each have different UI (icon, title, body, button). The **SecurityStatusFactory.java** provides a **verifierDisabledSummary()** that provides the right data for the off status. The only change needed would be to update the icons that it shows under the experiment flag. Another option would be to create a new **Play Protect_OFF** status, with the UI components customised for this case. **Creating a new Play Protect_OFF status would be a better choice in this case.**
- **ProtectHomeClusterModule:** This is a cluster that shows transient cards, which affect the SecurityStatus. In here there are provided all the controllers that show cards.

We would need to replace the **provideStatusController()** which at the moment provides the old controller (which displays LGTM, scanning and Play Protect off cards in the old status card) and replace it, under the experiment flag, with the new **StatusController.**

Summary of warnings in the subtitle



The **SecurityStatus.java** provides the functionality needed to display:

- Title
- Body of the card
- Overall state
- Icon type
- Navigation target

Based on this, the summary of warning will be provided by SecurityStatus. **SecurityStatus.getSummary()** retrieves the most recent SecurityStatusSummary. The Strings available will be updated.

Replacing the Legacy Status Controller

This controller will replace the existing **OldStatusController** which shows the current “Looks good”, “Scanning in Progress” and “Play Protect scanning off” cards.

Error handling

When the actions performed by the “Scan” or “Turn on” methods fail, we trigger error snackbars. If a Play Protect scan fails, we update the controller’s State to indicate that the scan has finished and update the card to show the most recent summary stored in the controller State as a fallback. If turning on app scanning fails, we keep showing the Play Protect off Status card.

Improvements from Legacy Status Card

- Making the Controller and View aware of when a scan is ongoing means the scanning card as well as animation will be retained correctly on View re-creation (such as orientation change)
- Storing the most recent Status means that if a scan fails, we still have the most recent status to render in the Status card
- Ensuring the Verifier sends a security status update when Play Protect is turned on, so if a scan fails after turning on, the controller will receive the latest security state to replace the Play Protect “off” state.
- Allowing the controller to depend on security status updates (as opposed to manually triggering an update) to update the Status card when a scan finishes, so we never show stale data in case the update is received after a scan finishes. However, if an update is received after the scan starts, but before it finishes, we *do* trigger an update of the Status card so as not to be stuck forever on the scanning card.
- Introducing a security state for when no scan data is available.

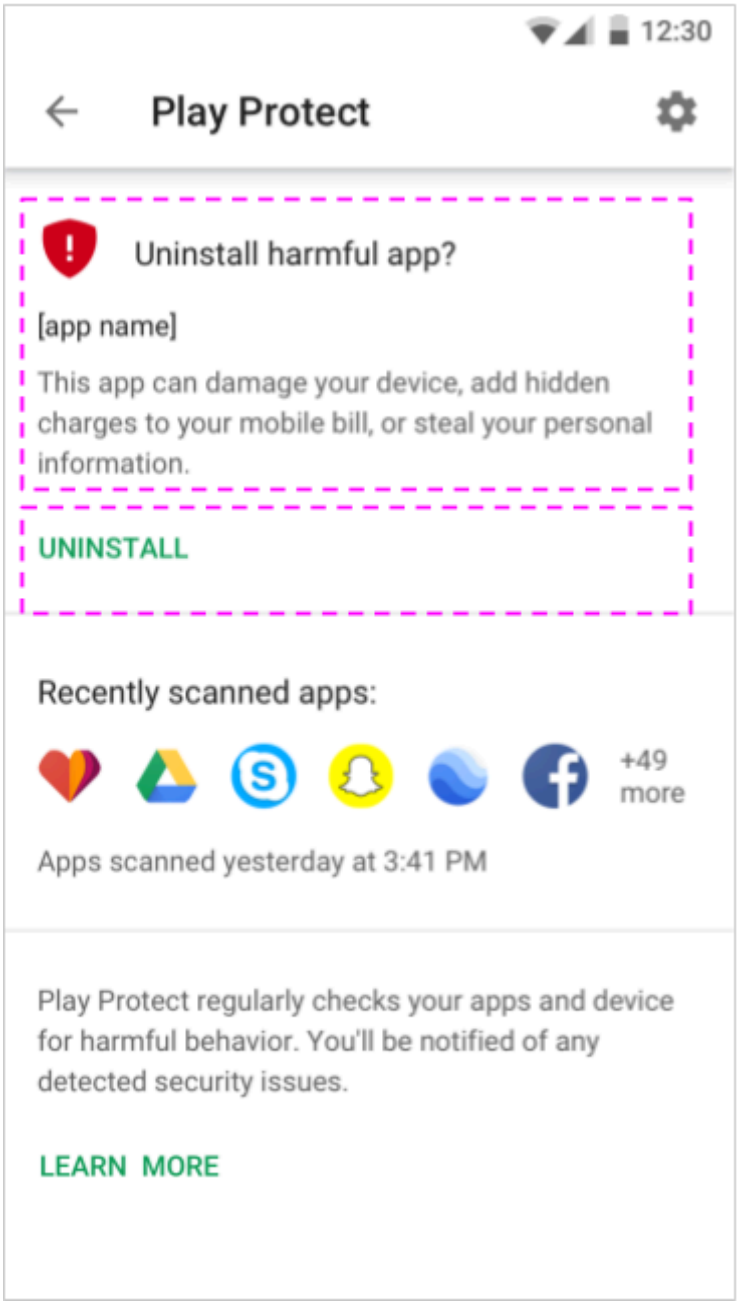
View

- Create a new view file for the new layout: **CardStatusView.java**
- Create new **CardViewBindable.java**
- Get the layout to always display on top of the screen.

Why do we need a new layout and not reuse the old one?

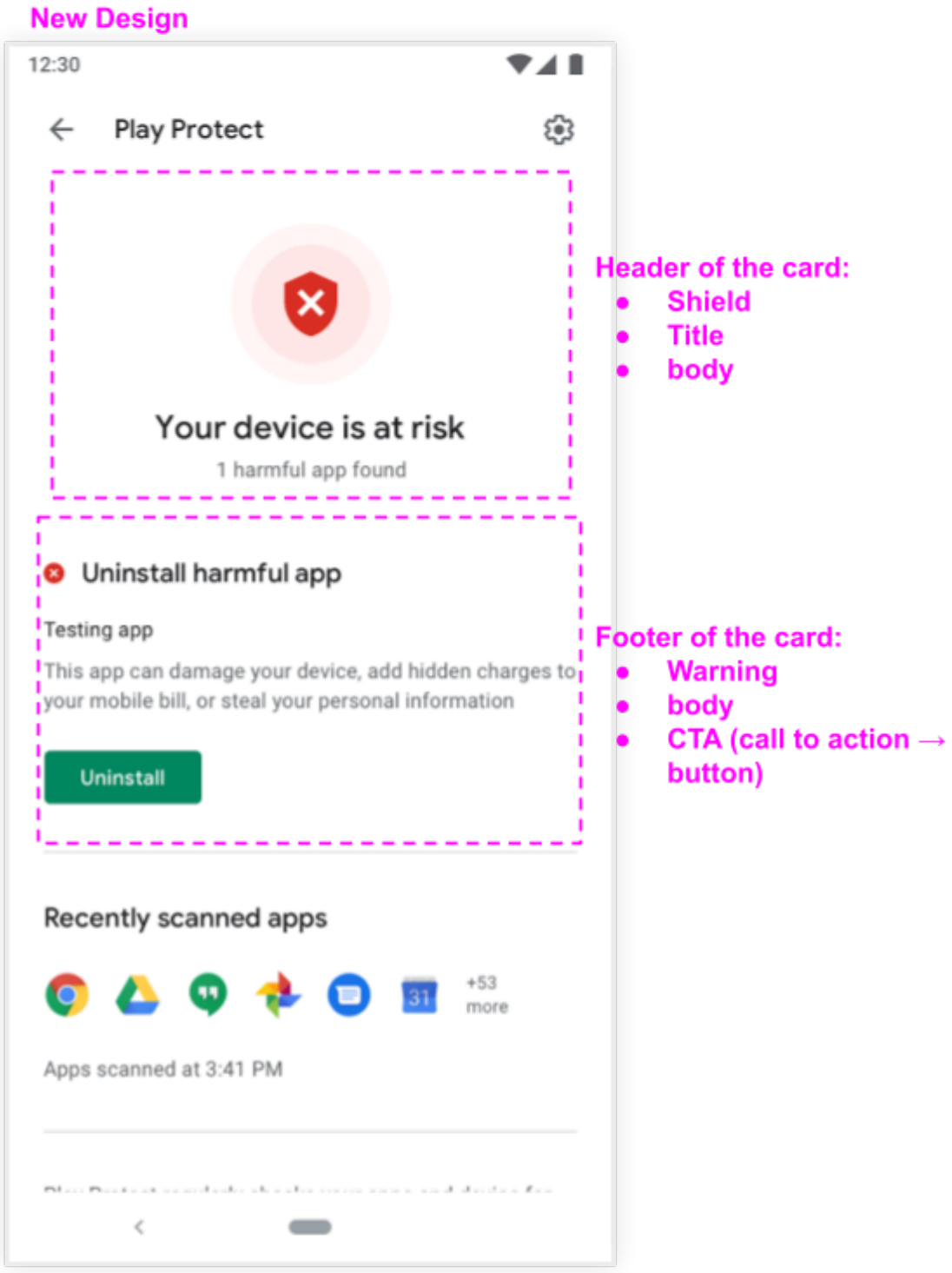
The new layout will be for the top part of Play Protect: shield, title, subtitle and button.
It’s better to split this from the **card_header** layout because we still need to maintain the old design for the “warning” part of the card.

Old design



Header of card:
- **Warning**
- **body**

Footer of card:
- **CTA (call to action → Button)**



In these images it can be noted that the old warning card can now be used as the new Status card Footer.

Creation of new strings for the new Status card

New strings had to be created to populate the new Security status card. These strings are created in a way that can later be translated to any language.

```
<!-- The security summary description to show on the persistent status card in
Play Protect for security notification warnings: harmful app removed or disabled,
mix green warnings -->
  <plurals name="security_summary_notification_body">
    <item quantity="one">1 security notification
<xliff:g id="string">%2$s</xliff:g></item>
<item quantity="other"><xliff:g id="number">%1$d</xliff:g> security notifications <xliff:g id="string">%2$s</xliff:g></item>
  </plurals>

  <!-- The security summary description to show on the persistent status card in Play Protect -->
  <string name="security_summary_verifier_body">Play apps are automatically scanned, but you need to turn on Play
    Protect app scanning to check apps from outside the Play Store</string>

  <!-- The security summary description to show on the persistent status card in Play Protect when Play Protect is scanning -->
  <string name="security_summary_scanning_body">Apps scanning in progress</string>
```

New colours for icon shields

I had to add new colours required for the icon assets as in an xml for resources

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
  <color name="shield_amber">@color/protect_amber_alert</color>
  <color name="secondary_text_color">#FF616161</color>
  <color name="red_warning">#d32f2f</color>
</resources>
```

Creation of shield icons

An experiment flag was created to guard all the changes behind it, to make sure new changes didn’t affect the preexisting code. The flag could be turned on and off, showing the new changes or not.

Verify if experiment is on or off

```
isExperimentAvailable =
        experimentFlag.getBool(
            ExperimentsFeatures.ENABLE_VISUAL_REFRESH);

    // Create exclamation shield for redesign. This icon will only be tinted with yellow.
    final exclamationShield =
        VectorDrawable.create(
            getContext().getResources(), Drawable.shield_exclamation_24dp, null);
    shieldExclamationDrawable =

    DrawableCompat.setTint(
        shieldExclamationDrawable, getResources().getColor(R.color.amber_alert));

    // Create exclamation shield, do not tint as various shield colors are used.
    final VectorDrawableCompat shieldAlertDrawCompat =
        VectorDrawableCompat.create(
            getContext().getResources(),
            isExperimentEnabled
                ? Drawable.shield_cross_24dp
                : Drawable.alert_black_24dp,
            null);
    shieldExclamationMutableDrawable = DrawableCompat.wrap(shieldAlertDrawCompat).mutate();

    // Create a refresh icon, and a checkmark shield icon, both pre-tinted the standard green.
    final VectorDrawableCompat shieldCheckMarkDrawCompat =
        VectorDrawableCompat.create(
            getContext().getResources(),
            isExperimentEnabled
                ? Drawable.ic_gpp_shield_tick_24dp
                : Drawable.ic_play_protect_check_black_24dp,
            null);
```

Implementing Lottie library for “scanning” animation

```
private LottieAnimationView scanAnimation;
 static final int SCAN_ANIMATION_LIGHT_THEME_JSON =
     com.android.scanning_light_theme;
 static final int SCAN_ANIMATION_DARK_THEME_JSON =
     com.android.scanning_light_theme;
@Override
 protected void onFinishInflate() {
   super.onFinishInflate();
   shieldIcon = findViewById(R.id.shield_icon);
   innerCircleIcon = findViewById(R.id.shield_inner_circle);
   outerCircleIcon = findViewById(R.id.shield_outer_circle);
   title = findViewById(R.id.status_title);
   subtitle = findViewById(R.id.status_subtitle);
   button = findViewById(R.id.status_button);
   scanAnimation = findViewById(R.id.lottie_scanning_animation_view);
   darkModeOn = DayNightThemeUtils.isDarkThemeApplied(getContext());
   // Add side padding decoration
   ContainerPaddingDecoration.addPadding(this);

   // Set scan animation for dark and light theme.
   scanAnimation.setAnimation(
       darkModeOn ? SCAN_ANIMATION_DARK_THEME_JSON : SCAN_ANIMATION_LIGHT_THEME_JSON);
```

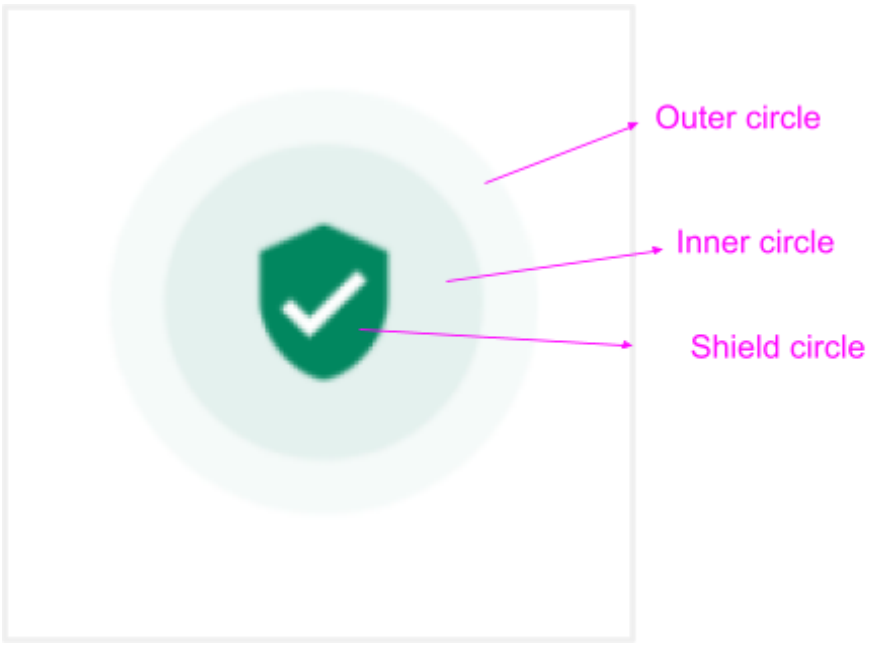
Resources for the animation

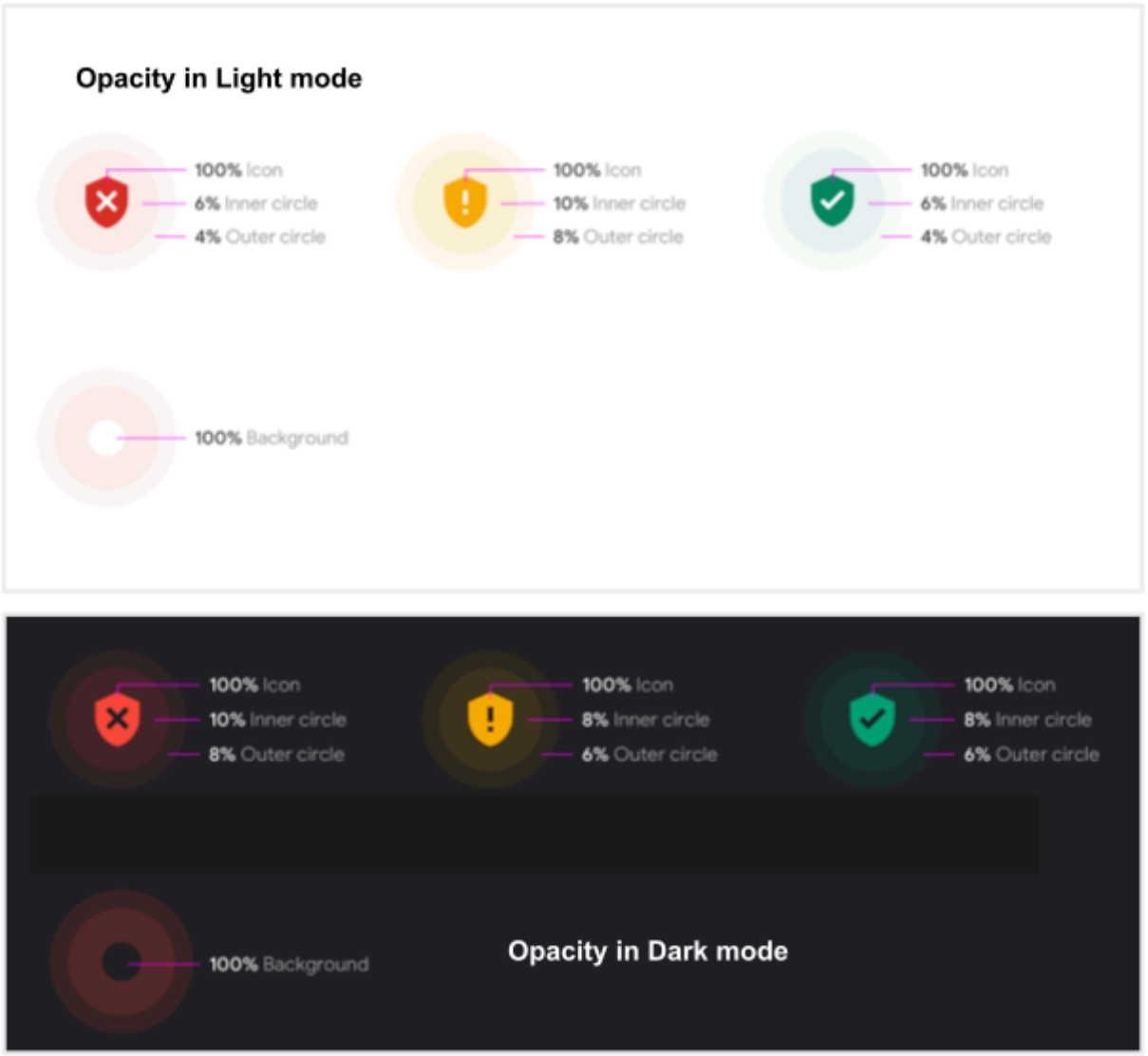
Configure icon in Status card

Shield illustration

The shield illustration will be composed of two concentric rings and a shield.

- Outer circle
- Inner circle
- Shield





UX and Scanning Animation

New classes StatusCardView and StatusCardViewBindable were added, and new layout files status_card.xml. Reusing existing classes and layouts for these was useless because the new designs and UX were completely different.

Scanning animation

The Lottie Library was used to implement the scanning animation. Lottie is a mobile library for Android and iOS that parses Adobe After Effects animations, created using shape layers, exported as JSON data files with BodyMovin and renders them natively on mobile. It allows a motion designer to create smooth, scalable animations that retain consistency across

multiple development environments. Animation values are stored in a JSON file, so the file is small. Lottie is also used by other teams in Google.

The animation was provided in the form of a JSON file from the UX team. It was configured in StatusCardView and triggered to start when the scanning card is bound. The animation was set to invisible for the summary cards, where the illustration was shown instead in the same position. The animation was set to loop infinitely until hidden in favour of a different Status card.

With Lottie, animations can be accurately and quickly converted into code. Animation values are stored in a JSON file.

Pros

1. The animation is stored as a JSon file, so, small file size
2. Free
3. Works better for mobile platforms than it does for web platforms. Initially created to work with iOS but then expanded to Android.
4. Already know this technology, I’ve worked with it previously.
5. Other teams also use Lottie for animations.

Cons

1. Needs AfterEffects animation

The animation was implemented on the view of the persistent status card, since there was no logic attached to this. It consisted mostly of turning views visible and invisible while the animation was playing.

An example:
When animation is playing the animation view will be visible, while the shield icon and the surrounding circles will be set to be invisible. Once the animation is over, the animation’s view is set to be gone, and the views of the icon are set to be visible.

The “scanning icon’s assets discussion”

Options to implement the new icon:

- **Single asset icon** (Initial recommendation from UX)

  We could have each icon composed of a single asset. With this approach we would end up needing 8 different assets:

  1. Red state - red cross icon light mode
  2. Yellow state - yellow exclamation icon light mode
  3. Green state - green check mark icon light mode
  4. Red off state - red exclamation icon light mode
  5. Red state - red cross icon dark mode
  6. Yellow state - yellow exclamation icon dark mode
  7. Green state - green check mark icon dark mode
  8. Red off state - red exclamation icon dark mode

  This would solve the problem of the different opacities, but would leave us with a lot of new assets. This is not ideal.

  This would make it easier for the creation of the Layout, since we would only need to add an **ImageView** to our xml file.

- **Multiple asset icon** (Recommended approach by me)

This would be composed of 3 different assets that we can combine to form all icons. With this approach we would end up needing 2 different assets:

1. Inner circle
2. Outer circle

We would continue to use the old shield icons for the center of the new icon. The circle icons would be imported in a neutral color that we can then change by code when each icon is created in the View. By having different assets, we can alter the opacity of each piece separately, as well as the size or color.

Using this approach would prevent us from having so many assets.

In order to stack the images to form the final icon, we could use a **RelativeLayout**. Using this layout would make it very straightforward to align all images together. We could have a separate xml file just for the icon, that would contain the RelativeLayout, and we can **include** this xml into the xml with the rest of the card UI. With this we will avoid nesting layouts, which is recommended to avoid by **Android.**

Pros and Cons of having the icon composed of different images.

Pros

- It might be easier for the animation part of the project if the icon is already split into different parts that can be individually animated.
- pacity can be changed depending on the shield (this is needed for light mode / dark mode with yellow shield)
- By having different assets, we would only need to import 2 new assets for the circles of the icon (since the shields are the same as the old ones), and in comparison to having a single asset per icon, this would be 2 new assets vs 8 new assets.

Cons

- Would need to have 2 separate layouts to make it work and to avoid having nested layouts.

We finally came to the conclusion, as a team, that the icon should be composed of multiple separated svg files.

Solution using multiple assets to generate multilayers icon:

- Created another layout just for the TextViews. Added the textViews within a RelativeLayout: this will allow me to stack one image on top of the other.
- Within the status card layout, this shield_icon layout is added as an <include> as recommended by Android. This avoids the need to nest layouts.

Changes to the layout included:

- Create a new layout of the status card: **card_status.xml and status_shield.xml:** This will be split into 2 layout files to separate the shield layout from the rest of the card. This way there’s no need to nest views.
  - Shield image on top of the card
  - Center the title and subtitle on the card, under the image
  - Add a button for “Scan”
  - Add this to ProtectSingleViewCard.java
- Update car_header.xml
  - Make shield icon smaller
  - Update paddings on card
  - Remove refresh icon
  - Remove the LGTM from this card’s controller.

Icon configuration

```
/**
  * Configures the status icon on Persistent status card. This icon is formed of 3
  * independent images: central shield, inner circle and outer circle.
  * This will configure the icon as a whole by setting the corresponding
  * Drawable, color and opacity to each asset.
  */
 private void iconConfig(
     Drawable shield,
     Drawable innerCircle,
     Drawable outerCircle,
     int color,
     int innerOpacity,
     int outerOpacity) {
   shieldIcon.setImageDrawable(shield);
   innerCircleIcon.setImageDrawable(innerCircle);
   outerCircleIcon.setImageDrawable(outerCircle);
   setColor(shield, color);
   setColor(innerCircle, color);
   setColor(outerCircle, color);
   iconInnerCircleDrawable.setAlpha(innerOpacity);
   iconOuterCircleDrawable.setAlpha(outerOpacity);
 }
```

Tablet

Since the button positioning is different in the tablet spec, a different xml layout will be used for it.

Divider under Play Protect off card

On the Status card for the “Play Protect off” state, a bottom divider will be shown to visually separate the content below the Status card. Generally, we use the DividerDecoration component to add bottom dividers, however in this case, since we only want to show the divider *sometimes,* the component doesn’t work well as it shows/hides the divider with a lag. Instead I inserted a View that looks like a divider into the Status card layout, and show/hide it depending on the content of the Status card.

Testing

Although testing is discussed last in the chapter, it was not the last thing done in the project. In fact, testing was done together with every change done to the code.

Type of testing done in the project:

- Unit tests

- Integration tests
- UI tests
- Standard accessibility testing for Google standards

Unit Tests

Unit tests are a must on every new feature to verify it’s working as expected and not breaking the already existing functionality. High percentage of test coverage is a requirement for launching any feature.

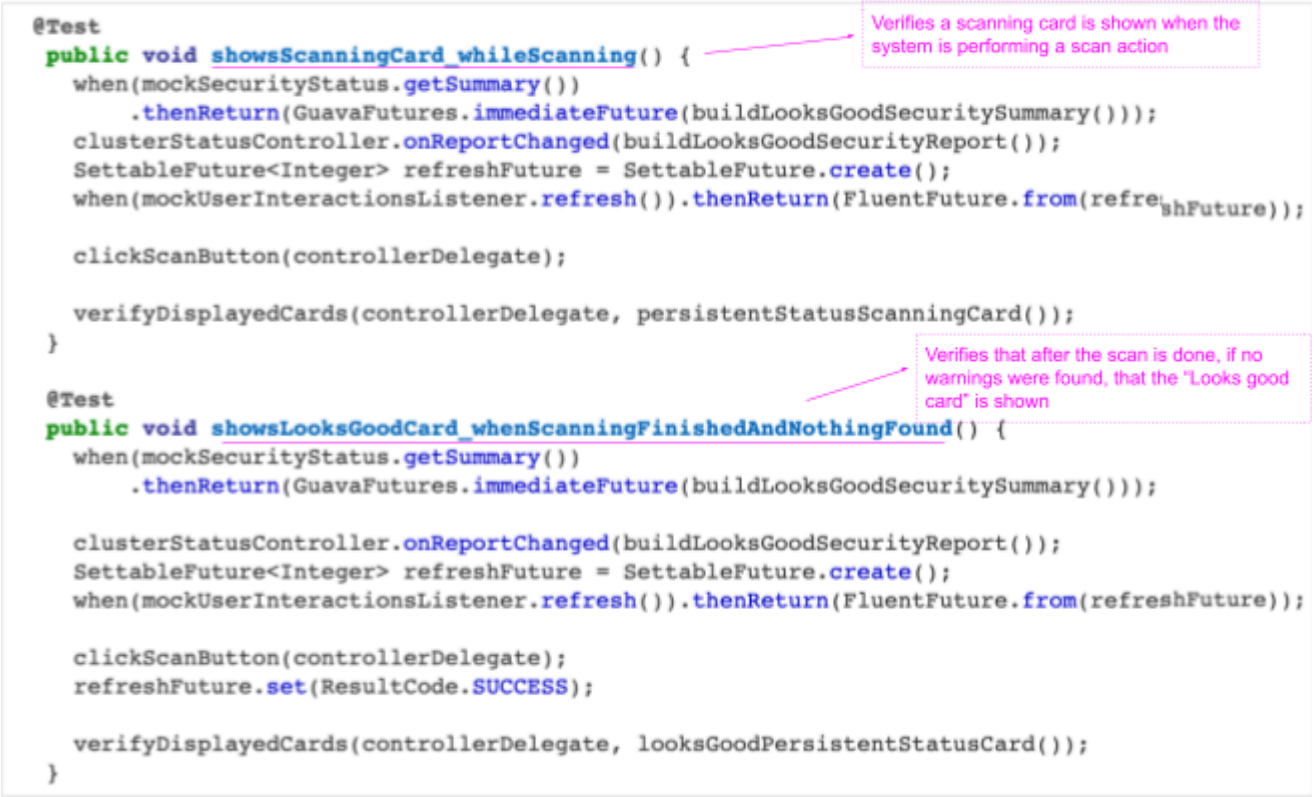
```
@Test
  public void setsBodyContentDescription_whenSpecified() {
    String expectedBodyContentDescription = "Test body content description";
    SecurityStatus securityStatus =
        SecurityStatus.builder()
            .setTitle(TITLE)
            .setBody(BODY)
            .setBodyWithStringLink(BODY_WITH_LINK)
            .setBodyContentDescription(expectedBodyContentDescription)
            .setOverallState(OVERALL_STATE)
            .setIconType(ICON_TYPE)
            .setNavigationTarget(NAVIGATION_TARGET)
            .build();

    assertThat(securityStatus.bodyContentDescription().toString())
        .isEqualTo(expectedBodyContentDescription);
  }
```

Functional tests

These are end to end tests to ensure the functionality of the app is as expected. Software modules are tested individually or in partial groups, dependencies in the system are mocked out. These tests are run automatically by Google Integration Testing Suit.

FUNCTIONAL TESTS EXAMPLE




```
@Test
public void showsScanningCard_whileScanning() {
  when(mockSecurityStatus.getSummary())
      .thenReturn(GuavaFutures.immediateFuture(buildLooksGoodSecuritySummary()));
  clusterStatusController.onReportChanged(buildLooksGoodSecurityReport());
  SettableFuture<Integer> refreshFuture = SettableFuture.create();
  when(mockUserInteractionsListener.refresh()).thenReturn(FluentFuture.from(refreshFuture));

  clickScanButton(controllerDelegate);

  verifyDisplayedCards(controllerDelegate, persistentStatusScanningCard());
}

@Test
public void showsLooksGoodCard_whenScanningFinishedAndNothingFound() {
  when(mockSecurityStatus.getSummary())
      .thenReturn(GuavaFutures.immediateFuture(buildLooksGoodSecuritySummary()));

  clusterStatusController.onReportChanged(buildLooksGoodSecurityReport());
  SettableFuture<Integer> refreshFuture = SettableFuture.create();
  when(mockUserInteractionsListener.refresh()).thenReturn(FluentFuture.from(refreshFuture));

  clickScanButton(controllerDelegate);
  refreshFuture.set(ResultCode.SUCCESS);

  verifyDisplayedCards(controllerDelegate, looksGoodPersistentStatusCard());
}
```

QA testing

This part of the test plan was carried out by a specialist QA on our team. After each session of QA, the tester would either approve the changes done or raise bugs in the system, attached to the specific problem found.

Chapter 6

Results & Evaluation

The work was completed within the expected timeframe, which was a great accomplishment for such a big project.
The Play Store new design was successfully incorporated into the Play Protect app, and from now on any UI change to Play Protect will be made using Play components. This will make any UI change easier and more straightforward.
The redesign was a multi team effort:

- UX teams from Play Store and Play Protect worked closely together for a long time until the new Play Protect designs were ready, and approved by Play Store.
- A lot of “alignment” meetings with the Play Store engineering team were required to understand which component would replace which piece of UI. This involved me, as Play Protect engineer leading this project, to get together (virtually, since the Play Store team is in the USA) with Play Store engineers leading the redesign work.
- Back and forward between engineering (myself) and the UX Play Protect team to communicate the requirements and specifications.

During the course of this phase of the project many design “bugs” were identified and addressed. It was not ideal to see the bugs on this stage, but at least we got them early enough so we could still fix them before sending them to production. Sometimes, even though design specialists create mocks of the designs and prototypes, until you see the changes on a real device, made with real code, it’s not obvious to see what’s not looking right. Such bugs were:

- Add extra small padding on top and bottom of the card dividers to give them more space (they were looking too tight)
- Change padding on bottom of the “Looks good” card
- Change “scanning” text to be a subtitle rather than a title
- Replace large padding with extra large on bottom of the cards

Once all Play Store components were included on our app, and we were all happy with this, we called this stage of the project closed and started with the second half of the project: The Play Protect redesign.

The organized and staggered implementation of the requirements made it easier to be organized in such a big job. It was easy to get overwhelmed with so many things to work on, so splitting the work into smaller, more manageable tasks, worked perfectly. My approach was to “grab” a requirement, for example: create the Status card. I would then split this task of creating the card into UI changes and functionality changes. This helped me focus on one thing at the time.

The work for the new “icon shield” on the Status card was a lot more complex than I expected. There were a lot of different ideas on how I should approach the development of this task. There were also discussions about the colours chosen for the shields, and we had to try on several options until we finally chose one. It would have been ideal to have this

decided before starting the implementation, but nothing is perfect, and we had to write and delete code a few times until we liked what we saw.

After all the cards and functionalities associated with them were completed, I started with the “scan animation” task. This was a very fun part of the project, as it involved something that was never done before on Play Protect. I was the first engineer to introduce the Lottie library to the team and use it on our codebase. The animation was a complete success!
I had to get together with the motion designers who created the animation, and together with the UX lead we picked the best animation from many options available. It was up to me to decide how long the animation loop would last.

An agile approach was followed, where we would have daily standups with engineering and UX together, and biweekly sprint meetings (also engineer & UX). This was a great approach to keep us all on the loop.

All features and UI changes were successfully implemented and the project was launched to the public on a 1% gradual rollout.
We first enabled the experiment for 1% of users, monitored the metrics to verify they were not negative, and then finally rolled out to 100%. This means that the new Play Protect UI changes were released for 1% of all Play Protect users in the world at first(sounds like a small number, but it’s actually released for hundreds of thousands of people). As very few bugs came back after the release, we rolled out to 100% of the users.

It’s very likely, given all the user research previously done for this project, that users will find all the new changes very useful and would be happy about this.

We based the success metrics of this feature on the success metrics of:

1. Impressions
2. Clicks on the “Scan” and “Turn on” buttons

For confidentiality reasons the actual metrics results can’t be shared. But I can confidently say that the whole project was a success. Google play Protect’s new redesign can be now enjoyed by millions of people all around the world.

Future work on more design updates are being considered for the future of Play Protect.

Chapter 7

Conclusions

This project was huge. It took me a long time to complete, but always within the expected timeframe. I always had the support of the whole team, and this was very reassuring.

I want to summarize in a very simple way what this project included:

- Implementing the reusable UI components (that not only reduce the amount of redundant work, but also increase consistency and velocity of building said UI) created by Play Store.
- Adopting the new design language from the Play Store, and adapting all our app UI to match with it.
- Implementing the “dark mode” feature on Play Protect, also using the design specifications from the Play Store.
- Creating a new and better user experience by adding a persistent Status card that showed the security state of the device.
- Redesigning all the cards on the app according to our new style.
- Adding a scanning animation to the Status card

All of the project’s objectives were successfully achieved within the expected timeframe, as previously mentioned.
In the few years I’ve been working at Google as an apprentice, this was the second end to end project I ever worked on. And certainly this was the biggest one. This put some pressure on me, since I was under the risk of feeling deeply overwhelmed. Up to this point, one of my biggest challenges had always been about splitting the work into smaller, manageable, pieces. I always required help from more senior staff to do this properly. But for this project I was challenged by my manager to do this by myself. Since the beginning of my time on this team my manager always required that I do this “splitting into smaller tasks” exercise, no matter the size of the task given. This was very valuable for me and when the time came to work on this big project, I already had the tools in me to try and break it up into smaller problems. Structuring the project the way I did was a good way of doing it, and gave me the organization I needed to carry out all the changes in the requirements without getting lost in the development. It was also a multi team effort that involved UX and engineering from Play Store, and UX and engineering from Play Protect.

There were some setbacks during the project, like for example UI requirements changing after development had already started, or having to make decisions about how to implement certain things that I hadn’t considered before starting. These were all included in the project’s time estimation (we always have to allow time for setbacks), and also were a great growth opportunity for me.

A recommendation I have for this project, though, is that there were design specs and mocks, but there weren’t any prototypes where you could actually see the interaction and flow of the new cards. Perhaps it would have been a good idea to have these available for the project, and we would have spotted some design problems earlier on.

I must emphasise that all this work was an incredible growth opportunity for me. Not only to grow my technical skills, but my personal ones as well. From the technical aspect of the project I had the opportunity to work a lot making use of different design patterns, understanding how to create a cleaner architecture, and improving my testing skills (since there were so many tests I had to write!). I had the opportunity to tackle very different problems and fix them on my own. I had to do so much research about different approaches, different ways of doing things. I had to go back and see how other previous projects were done, or even how other teams had previously implemented things I wanted to implement. From a more “people skills” point of view I was taught a lot of very valuable lessons like working with different teams that are in different parts of the world, and improving my communication skills to transmit to others all my thoughts about the work we were doing in an effective way. So there was definitely a lot of learning involved.
This was a very valuable experience for me as an apprentice and I feel I’ve become a better developer.

Apprenticeship Standards

1: Create effective and secure software solutions using contemporary software development languages to deliver the full range of functional and non-functional requirements using relevant development methodologies.

This can be demonstrated throughout my project. I used the latest approaches and techniques on every step of the way, following Google standards.

Karishma V - Peer
Veronica worked on the Redesign Project (part of a redesign of the Play Protect Home page), for which an animation had to be implemented. Before this project, animations were not used in our team’s product. Veronica used the Lottie Library to introduce this animation and implemented it independently and well.

2: Undertake analysis and design to create artefacts, such as use cases to produce robust software designs.

This was done during the analysis stage where we did the feasibility study and broke down the project into consistent pieces and defined all use cases in the form of users stories.

Simon W - Manager
Veronica produced one large design in this period, for the UX redesign of Google Play Protect home. Her design was detailed, and her focus on the UI design aspects was thorough.

Niko B - peer
Verónica demonstrated their ability to analyse and design in their design doc for the Google Play Protect (a.k.a GPP) Home Status Card feature. Her design doc is fairly detailed as to how resources and animations would be managed/implemented, with a thought out rationale as to why those design decisions were made.

Some relevant feedback regarding that design doc is that it lacked the same level of detail for the actual logic changes, involving the new controller and modifications to use the SecurityStatusSummary in said controller.

Raissa DG - Host
Veronica has written a couple of documents outlining her thought process (e.g. her mini-design for the re-enable flow for a product use case, design doc for Redesign UI). She also noted down her ideas in her snippets.

3: Produce high quality code with sound syntax in at least one language following best practices and standards.

The project was done using Java coding language. My code was constantly being reviewed by my peers, as part of Google’s reviewing process. Also my code had to be up to the standards of the tooling and automation process that doesn’t allow a developer to add code to the codebase that hasn’t been tested, and will alert you if the code breaks any other tests. The system also measures performance, and runs the code in different devices.

Niko B - peer / host
Verónica mostly worked on Java, so my feedback here will be focused on Java.

As described below, her smaller and more focused changes mostly adhered to our internal Java style guide and best practices. I think this is evidenced fairly well in unit tests: changes are focused, new tests are added when necessary, and they are clean and are adequately named and structured.

Code quality does take a hit once changes get larger and less focused. This can be evidenced by how much feedback these changes receive during code review as opposed to smaller ones. To be clear, more senior engineers will also struggle to maintain and submit larger changes, but they also tend to avoid authoring large changes altogether, opting instead to break the problem down into smaller chunks, and submitting multiple bite-sized changes rather than a single XL one.

Karishma V - peer
Veronica has mostly worked in Java (Android development) and through experience and code reviews, has learnt best practices and standards

Simon W - Manager
Veronica has made good progress in this area, if you compare the quality of code she was producing when working on the `DateFormatter` project with her more recent work on UX redesign you can see a clear improvement in the number of revisions her CLs took, and her ability to split her work into clear units.

Her Java has likewise improved, adhering more closely to general best practice and the specifics of Google’s style guide. There are still aspects of the Java language that she struggles with, including generics and futures. The main area of improvement for Veronica here would be producing code where there are no pre-existing examples to base the code on.

4: Perform code reviews, debugging and refactoring to improve code quality and efficiency.

Code HAS to be reviewed by other engineers, and the CI system runs all tests before submitting the code.
Peer feedback always encourages you as a developer to refactor your code to have the best version of it possible. Google coding and review standards are very high.

Simon W - Manager
In some ways a lot of the work required for Veronica’s project can be considered to involve refactoring of existing flows. For example she needed to update the way controllers receive model updates to correctly pass the needed data through to her new view.

Veronica performed a few code reviews, but I wouldn’t be expecting her to be performing a great number of code reviews at the moment.

Niko B - peer / host
Verónica did refactoring work as preparation for her work on the GPP Home Status Card. Specifically, (1) she reworked our FakeStreamSectionControllerDelegate to use more generic ViewBindables as representations for UI elements, and (2) she changed how we deliver security status updates to our controllers by also providing a summary.

Both pieces of work were not strictly necessary for her project, but were necessary to avoid

incurring in technical debt.

As to code reviews, Verónica did not perform enough for me to be able to provide meaningful feedback. As to debugging, Verónica did so well and when necessary, so I don’t have much feedback to provide there either.

5: Test code to ensure that the functional and non-functional requirements have been met.

Simon W - Manager
Veronica created unit and functional tests for her projects. Her tests were normally clearly written, following closely the conventions of the existing tests, and were usually sufficient comprehensive at first try to cover the change

Karishma V - peer
Veronica has developed skills on unit testing through her work on the Redesign project.

Niko B - peer / host
Verónica added unit and functional (when needed, e.g for Detox) tests to ensure her changes were correct and to avoid regressions. Her tests were often clear, short and precise, and served fairly well as additional documentation of the system’s behaviour.

Verónica’s unit tests usually followed best practices such as:

- Only one behaviour tested per unit test
- Include behaviour, condition, and expectation to the test name
- Keep tests short and focused
- Visually split the test into arrange/act/assert sections.

Raissa DG - host
Veronica consistently wrote tests to accompany her code changes, and I believe she put in a good effort in meeting this practice in our team.

6: Deliver software solutions using industry standard build processes, and tools for configuration management, version control and software build, release and deployment into enterprise environments.

Simon W - Manager
Veronica used the same version control and build tools as the rest of our team, and was confident in using both our internal tooling (citc), tooling that is available externally (bazel) and tooling that would be considered industry standard (mercurial, Android build tools).

Niko B - peer / host
Verónica mostly did not interact with build or deployment systems directly. She did however work with Phenotype/Mendel as a flag configuration system in order to enable/disable features she was working on server-side.

As for version control, Verónica used CitC directly. I think not using a version control system (on top of CitC) that would support chaining CLs (e.g. git5 or fig) may have hindered her development speed. The reason I state this is that she often found herself with large CLs which are typically a lot harder to maintain and submit, whereas the same code in smaller chunks would have been easier to get through.

Technical Knowledge

Knows and understands:

7: How to operate at all stages of the software development lifecycle.

Raissa DG - host
Veronica wrote a design doc for her plan to implement Play Protect redesign UI, and the design was reviewed by the team. She implemented all development, however she did not manage to see it through code completion before going on Maternity leave.

Niko B - peer / host
For the GPP Home Status Card feature Verónica produced a design doc focused on how to add animations and tweak the UI to adapt to the UX requirements. She also prototyped the feature, and managed to implement and test most of it.

Karishma V - peer
Veronica has had experience working on multiple stages of the software development cycle, including

1. Product development phase: For the Redesign project, Veronica wrote the Play Protect Passport (a document summarising the project scope and requirements).
2. Design phase: For the Oleg project, Veronica came up with and wrote a design document.
3. Development phase: Veronica has worked on development in multiple projects including the Redesign project, Detox project and Visual Refresh project.
4. Bug-fixing: For the Visual Refresh project, which involved refactoring our team’s code to align with a Play store-wide UX redesign, Veronica contributed to many UX bug-fixes for this project, and worked towards an external team’s deadline.

Simon W - Manager
Unfortunately Veronica did not get the chance to work on the last stage of a software development process, release the new feature and analysing its success due to taking early maternity leave. Hopefully she will get a chance to work on this in the remaining time she has working with us. She worked on all other parts of the software development lifecycle, from producing an initial design through to delivering working code with tests.

8: How teams work effectively to develop software solutions embracing agile and other development approaches.

Simon W - Manager
Veronica worked within our fortnightly agile process. She was a friendly and collaborative team member. She would reach out for help when she needed it, but I needed to encourage her (and still do) at this stage in her career to reach out more when stuck for assistance from her colleagues. There is always a balance when developing new skills in trying to solve problems independently and with help, and at the moment I think Veronica would benefit from more regularly asking for help than she currently does.

Karishma V - peer
We have an agile development approach on our team and follow 2-week sprints where we commit to a certain number of tasks every sprint. Veronica participated in the team sprints, organising and planning her tasks in advance, and also adjusting course mid-sprint if necessary.

Niko B - peer / host
Process-wise, Verónica would participate in the team’s fortnightly scrum-like iteration plannings, and some other times opt to individually plan her work for the next couple of

weeks. She also participated in our daily standup meetings.

As to interacting with other members of the team:

- She worked closely with UX designers while designing and prototyping the GPP Home Status Card feature
- She would reach out to more senior engineers in the team to ask questions, or get their feedback. As mentioned throughout this doc, this occasionally happened a bit too late, where Verónica would be stuck for quite some time before reaching out.

Raissa DG - host
Veronica demonstrated this well when we worked together as a team to fix bugs for an upcoming deadline. She contributed 9 fixes from a bug list of about 28 issues (calculated from number of CLs from main contributors).

9: How to apply software analysis and design approaches.

Niko B - peer / host
Verónica demonstrated their ability to analyse and design in their design doc for the GPP Home Status Card feature. Her design doc is detailed as to how resources and animations would be managed/implemented, with a thought out rationale as to why those design decisions were made.

Some relevant feedback regarding that design doc is that it lacked the same level of detail for the actual logic changes, involving the new controller and modifications to use the SecurityStatusSummary in said controller.

Karishma V - peer
The Visual Refresh project involved refactoring our team’s code to align with a Play store-wide UX redesign. Veronica worked on multiple UX and layout change tasks for this. This project also required adapting to new UX standards and components. Veronica worked with engineers outside our team to understand the new processes better and apply them in our team’s work.

Simon W - Manager
Veronica’s design approach was thorough in a number of ways, and definitely this helped her progress in the project. However I would suggest that she focus more in the design on

the technical difficulties she is likely to encounter - currently she is more likely to either use an existing code example, or an extensive period of “prototyping”, neither of which are going to help her grow her skills to reason about software up-front before writing it.

10: How to interpret and implement a design, compliant with functional, non-functional and security requirements.

Simon W - Manager
Veronica implements her design for UX redesign, and also several parts of the Visual redesign doc from others.

Karishma V - peer
Veronica worked on the design of the Redesign project which is part of a redesign of the Play Protect Home page. This involved gathering requirements from Product and UX team members, designing the feature, writing a design document, having the team review the design and addressing any questions and concerns about design.

Niko B - peer/ host
Verónica demonstrated this skill by implementing her own design for the GPP Home Status Card feature. Though she did not finish the implementation, she did get to a working implementation covering most user flows.

Her work with implementing VisDRe (a wider Play Store UX update) and Dark Mode compliance in GPP Home also demonstrate her ability to implement others’ design specifications.

Raissa DG - host
Veronica has shown skill in translating mocks into code, however she sometimes struggles in creating robust solutions.

11: How to perform functional and unit testing.

Raissa DG - host
Veronica’s code changes, where possible, are consistently accompanied by tests.

Niko B - peer / host
Verónica added unit and functional (when needed, e.g for Detox) tests to ensure her

changes were correct and to avoid regressions. Her tests were often clear, short and precise, and served fairly well as additional documentation of the system’s behaviour.

Verónica’s unit tests usually followed the Java style guide’s advice closely:

- Only one behaviour tested per unit test
- Include behaviour, condition, and expectation to the test name
- Keep tests short and focused
- Visually split the test into arrange/act/assert sections

Simon W - Manager

Veronica performed unit testing and functional testing on her code changes to good quality.

12: How to use and apply the range of software tools used in Software engineering.

Simon W - Manager

Veronica used a range of software tools for engineering, including

- internal and external build tools
- version control tools
- IDEs (Android Studio)
- emulators
- debug logging and stack trace analysis
- layout inspector tools for Android.

Niko B - peer / host

Verónica useda wide variety of tools:

- Robolectric as a testing framework for Android
- ADB for interacting with Android devices
- Layout viewers (such as HSV)
- Android Studio
- Android developer options
- Version control
- Phenotype for guarding features behind flags

Raissa DG - host

Veronica has met this expectation very well, given the range of tools we have to use at Google on a daily basis!

E.g. Veronica set up her ASWB for code changes and compiling from google3/, uses Critique for code reviews, etc.

References

Internal References

All information was taken from Google’s internal documentation. For this reason I can’t make direct reference to links or titles.

- New Play Store design documentation
- Play Store UI components
- Play Store code samples
- New Play Protect design documentation
- Play Protect code samples
- UX research documentation
- UX writing documentation

External References

Animations

Native animations on Android

Android Developers (2018). *“Introduction to animations”*. [Online]. **Google**. (Last updated April 10th, 2019). https://developer.android.com/training/animation/overview

Android Developers (2018). *“Animate layout changes using a transition”*. [Online]. January 15th, 2018. **Google**. (Last updated April 10th, 2019). https://developer.android.com/training/transitions

Android Developers (2018). *“Create a custom transition animation”*. [Online].January 15th, 2018. **Google**. (Last updated April 11th, 2019). https://developer.android.com/training/transitions

Android Developers (2017). *“Animate movement using spring physics”*. [Online]. March 27th, 2017. **Google**. (Last updated April 20th, 2019). https://developer.android.com/training/transitions

Lottie animations

LottieFiles (2018). *“Join the revolution in motion”*. [Online]. September 2018. **Airbnb**. (Last updated April 25th). https://lottiefiles.com/

LottieFiles (2018). “What is Lottie: the official guide to Lottie”. [Online]. September 2018. **Airbnb**. (Last updated April 25th). https://lottiefiles.com/what-is-lottie

LottieFiles (2018). *“Learn from lottie experts”*. [Online]. January 2019. **Airbnb**. (Last updated April 28th). https://lottiefiles.com/course

LottieFiles (2018). *“Saptarshi’s guide to Lottie and Lottie files”*. [Online]. January 2019. **Airbnb**. (Last updated April 28th). https://lottiefiles.com/course/saptarshi-s-guide-to-lottie-and-lottiefiles/?utm_source=twitter&utm_medium=social&utm_campaign=saptarshi_course

LottieFiles (2018). *“Icon animation for UX / UI & product Designers”*. [Online]. March 2019. **Airbnb**. (Last updated April 30th). https://lottiefiles.com/course/icon-animation-for-ux-ui-and-product-designers

Haiku animations

Haiku (2018). *“Create engaging animations for any app or website”*. [Online]. December 2018. **Haiku**. (Last updated April 15th). https://www.haikuanimator.com/

Haiku (2018). *“Blog”*. [Online]. January 2019. **Haiku**. (Last updated April 17th). https://www.haikuanimator.com/blog/

Technical

Componentization

Bedell, Crystal (2019). *“componentization (component-based development)”.* [Online]. **TechTarget.** Published *November 2018.* https://searchapparchitecture.techtarget.com/definition/componentization-component-based-development

Kyung Su Hwang, Jian Feng Cui, Heung Seok Chae (2009). *“An Automated Approach to Componentization of Java Source Code”.* [Online]. **IEEE.** Published October 2019. https://ieeexplore.ieee.org/document/5329105

L. Tahvildari, Shimin Li (2006). “*JComp: A Reuse-Driven Componentization Framework for Java Applications”.* [Online]. **IEEE.** Published 2006. https://www.researchgate.net/publication/4242325_JComp_A_Reuse-Driven_Componentization_Framework_for_Java_Applications

Stackoverflow

Stackoverflow (2017). *“Lottie animation not working”*. [Online]. August 27th, 2017. **Stackoverflow**. October 2nd, 2019. https://stackoverflow.com/questions/45924996/lottie-animation-not-working

Stackoverflow (2017). *“Using lottie Android with vector image or svg image?”*. [Online]. July 5th, 2017. **Stackoverflow**. October 14nd, 2019. https://stackoverflow.com/questions/45924996/lottie-animation-not-working

UI

Google Developers (2019). *“Google Material”.* [Online]. September 2019. https://material.io/design

Google Developers (2019). *“Improving layouts. Reusing layouts”.* [Online]. September 2019. https://developer.android.com/training/improving-layouts/reusing-layouts

Other

Medium (2017). *“Difference between MVC and MVVM”*. [Online]. July 28th, 2017. **Mobidroid**. July 17, 2019. https://medium.com/mobidroid/difference-between-mvc-and-mvvm-456ec67181f6

Android Developers (2019). *“Layouts”*. [Online]. April 2019. **Google.** September 12th, 2019. https://developer.android.com/guide/topics/ui/declaring-layout.

Rastogi, Udit (2019). “replace Conditional Statements (IF/ELSE or SWITCH) with Polymorphism”. [Online]. **Codementor community**. Published February 07, 2019. https://www.codementor.io/@uditrastogi/replace-conditional-statements-if-else-or-switch-with-polymorphism-ryl8mx4ns

Michael C. Feathers (Robert C. Martin Series). (2005). *“Working effectively with legacy code”*. New Jersey: Prentice Hall.

Robert C. Martin (Robert C. Martin Series). (2008). *“Clean code”*. New Jersey: Prentice Hall.