Curious Reader Container App Specification

Curious Learning

Specification for
a distribution mechanism that allows a user to engage with multiple web-based learning apps via smartphone and tablet devices

Prepared by: Ben Burrage <bburrage@curiouslearning.org>

DOCUMENT REVISION HISTORY *assign date stamp/sign off to accepted versions*		
Date	**Version No**	**Author**
November 8th, 2022	v.1.0	bburrage@curiouslearning.org

Introduction

The following document will outline the concept of a Curious Reader “container” app (aka: an “app of apps”). This product is being developed to solve multiple issues related to the Curious Learning use of the Google Play Store as the sole means of distributing content. Those issues are:

1. The Play Store experience inherently creates a siloed experience for the user– native apps cannot connect to other apps and users must be re-acquired driving up the total cost of literacy.
2. Early literacy learning is a journey– providing a semi-structured environment to add new content as it becomes needed for users to explore, engage, learn, practice, and move on is paramount to pedagogy.
3. Having 50+ localized versions of each app in the Play Store is a very painful technical endeavor– especially when requiring mandatory app updates!

From ~January - August 2022, we created a web version of the Feed The Monster app in React and H5P– proving that the bulk of the gameplay could be handled as a web app instead of a Unity-based app.

In September 2022, we pivoted to a Typescript and HTML version that afforded better animation and which we wrapped as a progressive web app (PWA) that would allow you to download the game and run it offline.

We ran multiple advertising tests on Facebook promoting this new PWA version of Feed the Monster and found that users were averse to being forced to install the web app (~1% conversion) but not averse to hopping in to play the web-based app (~10% conversion) if given the chance. This was still slightly less than the ~13% conversion that would install the Unity-based version from the Play Store.

However, without installing the app on a phone or device, the assumption is that engagement would drop significantly.

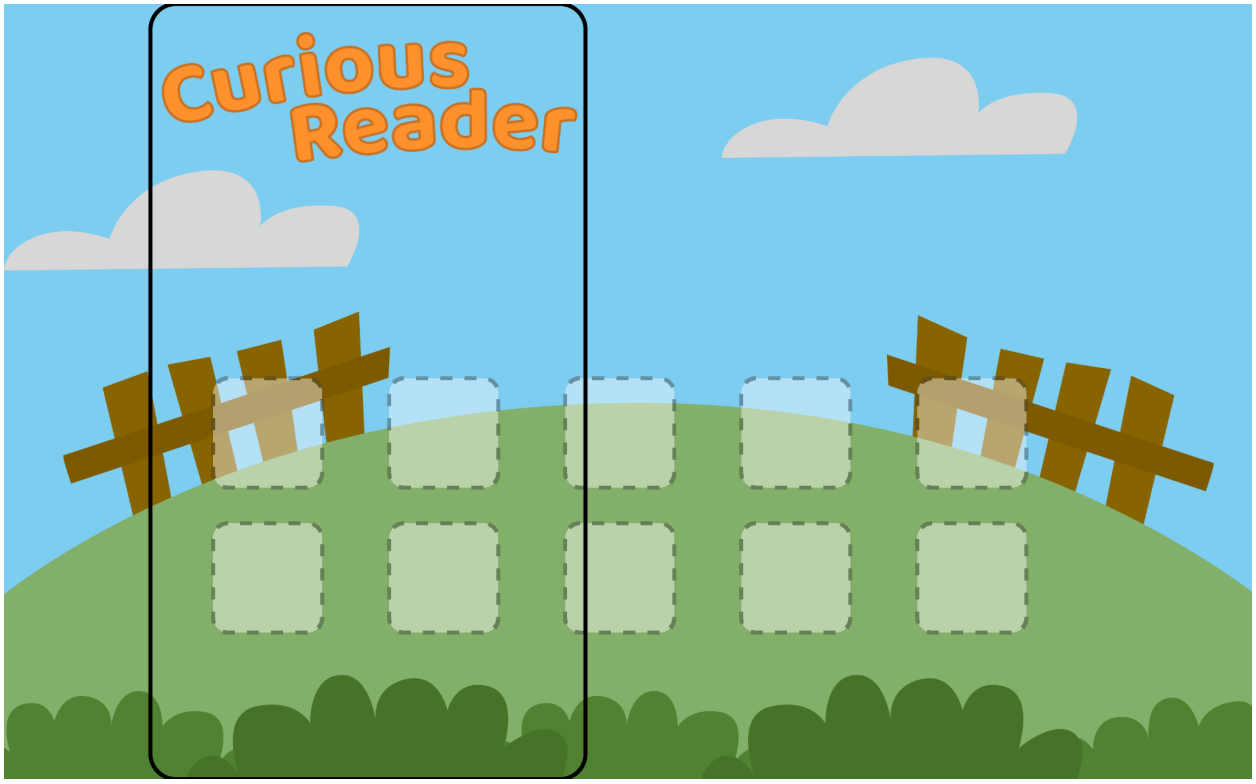
Overview

The Curious Reader container app is an attempt at solving both the Play Store issues described above and the extended engagement issue with web apps we saw as a result of the PWA installation (or lack thereof!) campaigns.

The Curious Reader container app is an Android app that contains:

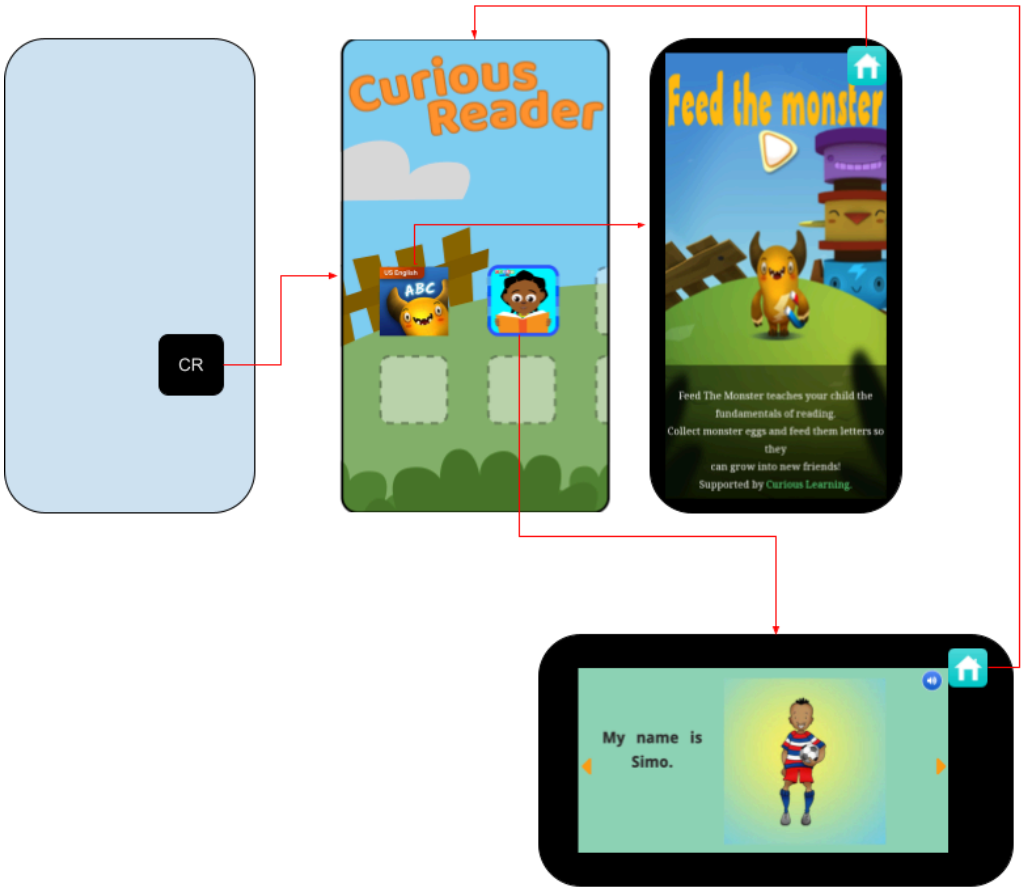
1. a simple webview for which to view and consume web-based content,
2. a service worker to store files of web apps for offline use in the Curious Reader container app,
3. an analytics package (Firebase) to collect and report data,
4. a data storage mechanism that may be used to store information pertaining to meta-data about the user's experience– such as what apps to display, when to display new apps, what language they are learning in, etc. etc.

Design Mockups



A horizontal and vertical rendering of the Curious Reader container app.

User Flow



1. User launches the Curious Reader container app from their Android device.
    a. User can swipe left or right to see a carousel of icons.
2. User selects a web app to play.
3. User can return to the main screen at any point to choose another web app to play.

Over time, icons for new apps will appear. For version 1, this will be when the main server is updated with a new app icon leading to a new piece of PWA content. Think of this like how a web browser like Google Chrome would merely display a webpage when you input a URL.

Additional Needs

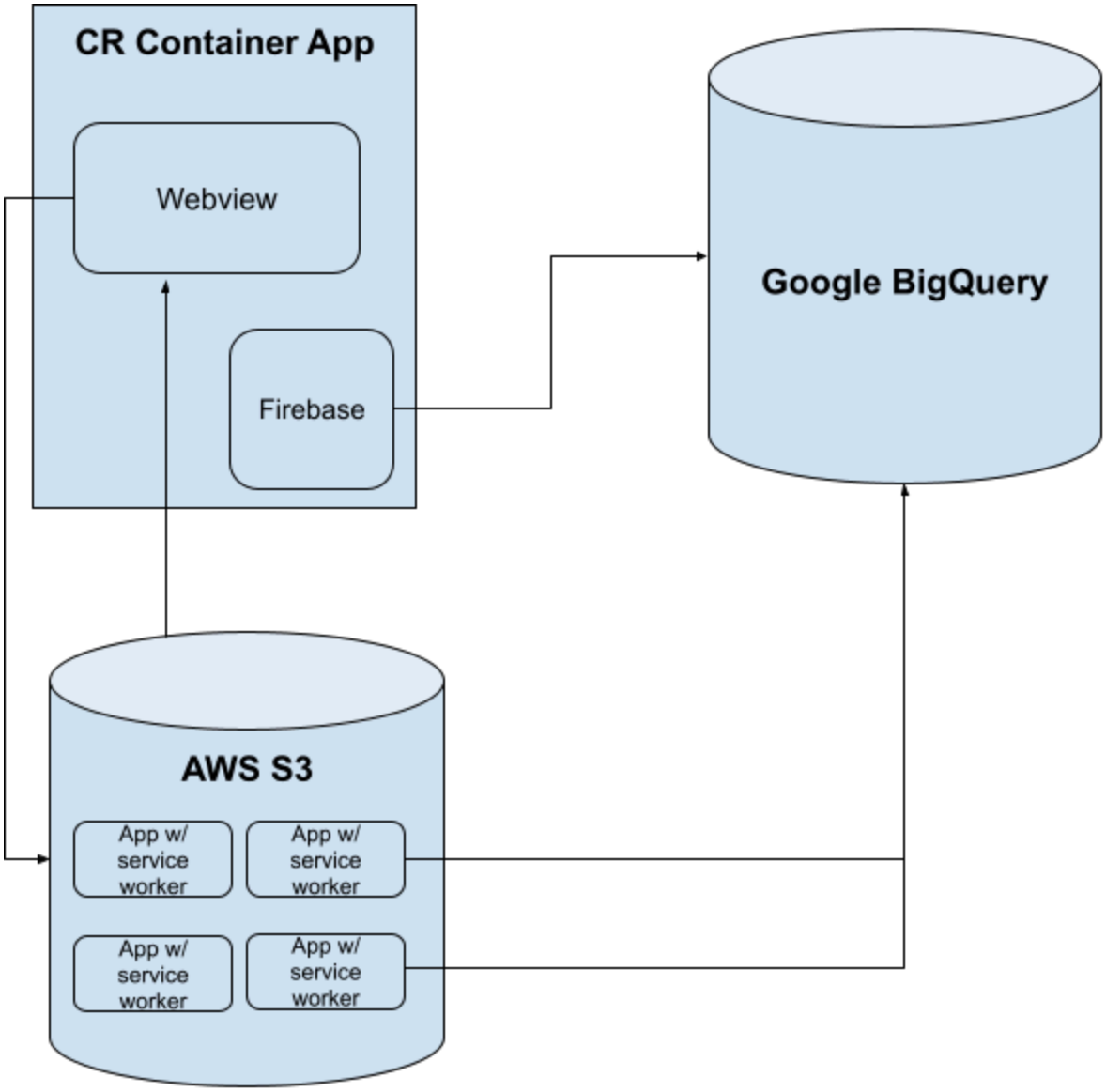
The Curious Reader container app should be able to work offline with locally stored PWA content, once that content has been downloaded initially. The Curious Reader container app however will require internet access in order to:

- Update the UI of the Curious Reader content app and what content is available to be transferred via the main hosting server.
- Receive aforementioned new PWA content from the server if it is available.
- Update PWA content under two scenarios:
    - If an update to an existing PWA has been pushed to the server.
    - If the service worker for an existing PWA loses files necessary to run the PWA and must reach out to the server to re-obtain them.

if the Curious Reader container app needs to connect to the internet to obtain files necessary to run new or existing PWA content but is not connected to the internet, the Curious Reader container app should be able to inform the user that they will need to connect to the internet. The user should ALWAYS be able to enter the Curious Reader container app, be able to try to open a piece of PWA, and be informed by the app if it needs to connect to the internet in order to run.

The Curious Reader container app should also be able to have basic analytics tracking capabilities. This includes, in a first version, being able to track basic usage statistics like how long an average user spends playing PWAs within the Curious Reader container app. Later versions and iterations of data collection can become more granular, like being able to track *which* PWAs are being played and for how long.

Data Flow



Open Questions & Future Considerations

V0

- Would Google take an app like this down where they don't have as strict control and review over the content as it is dynamic? Are there additional considerations we need to take into account publishing an app like this through the Play Store?
- Is the Curious Reader container app actually a PWA itself? Or an Android-based app with a webview?
- What happens if the service worker loses a file necessary to play a PWA? Are we able to know what is happening and redirect that information to a user say if they are not connected to the internet?

- How will new content be displayed in the UI when it is available? Does it just show up in the Curious Reader container app UI? Are there markings to designate that it is new?
- What happens if a web app is updated and requires the service worker to pull down new files?
- What happens when the container app has Firebase and the web apps themselves have Firebase?
- Do we need to change the privacy policy on curiouslearning.org?

V1 & beyond

- Add a second PWA in the same language to the Curious Reader Container app (a Curious Reader book)

Language selection and appropriate content delivery:

- Can we pass a language UTM parameter into the Curious Reader container app that auto-selects a language?
- How will we allow someone to change their language if they downloaded the app without a passed language UTM parameter? Would we allow someone to switch languages? How would that experience work?

Personalization:

- Where does the logic live for when new apps should be displayed? In the app? On a server?
- From “Feature expansion”, is it possible to create bi-directional communication between the Curious Reader container app and the PWA? Or is only one-way communication possible? Or is communication between the two pieces not possible at all?

Feature expansion:

- Can new content within a single app be delivered in “batches” or “lessons” such that they are not having to download a lot of unnecessary content data?
- How will we notify offline users when new content is available?
- Can we recreate the “profile” concept from H5P FTM in the Curious Reader container app that contains meta-data about their experience? (e.g. what profile did they select, what apps do they have available, what milestones have they completed in each app, etc. etc.)
- How would selecting a profile at the Container level affect play of the PWA app? Would they need to communicate with one another? How would we store progress?
- Is it possible for a web app contained within the Curious Reader container app to bubble up information that can be stored? E.g. This user has reached level 10 on Feed the Monster– therefore we want to show a new Curious Reader book the next time the Curious Reader app is launched.
- What types of data do we want the Curious Reader container app to collect longer term?
- Is there a limit to how many files or size of the cache the service worker has access to?