# Health Canada COVID App for Vaccines and Treatments

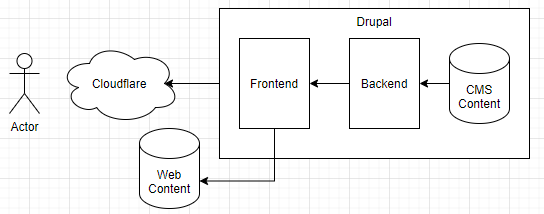
## Overview

The Health Canada COVID Response App for Vaccines and Treatments will be mobile, available for download from app stores, with links back to additional information and documents on the main Health Canada site; and provide up-to-date information about vaccines and treatments. The long view is that this could become a starting point for the *Consumer Health Information App*.

* + Consumer-focused – emphasis here on easy to use, not intended for clinical users.
  + Accessibility is high value – WCAG, French/English language support is mandatory, etc.
  + API improvements have high value – CVT is an example client consumer of the API.

## Current State

The current Health Canada COVID Vaccine and Treatment site looks like this:

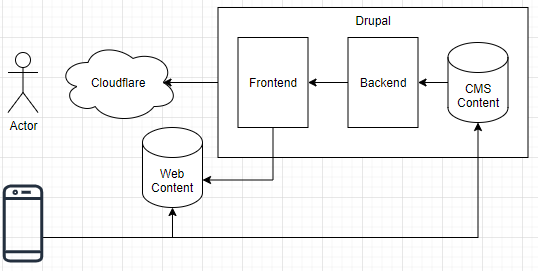


<https://covid-vaccine.canada.ca>[/](https://covid-vaccine.canada.ca/)

## Future State Technical Approach

### Develop new Mobile App with Existing APIs

* + Using a framework like React Native – generate Android and iOS install packages.
  + Prioritize changing the existing APIs over developing separate new APIs.
  + Prioritize a multiplatform framework over parallel development for different channels.
  + This allows for some substantial improvements in User Experience.
  + **Disposition:** this is our preferred approach.



**Technology options considered included:**

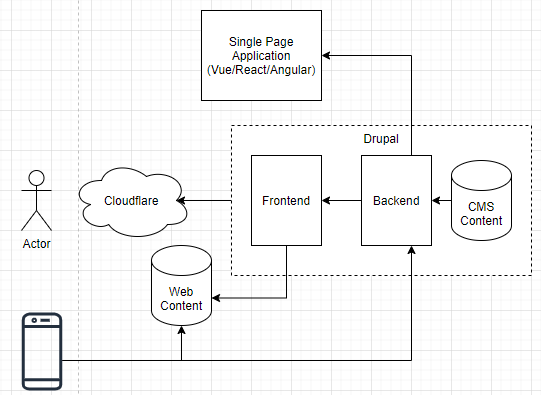
* + Enhance existing portal (covid-vaccine.canada.ca) with better responsive design.
  + *Develop a net new hybrid/native mobile app with minimal changes to the existing APIs.*
  + Convert the existing Drupal portal into a headless CMS to support net new mobile app.

### Enhanced Responsive Design

* + Using a Responsive Web Design library like Bootstrap will accommodate a collapsed view and menus on a small screen like a cell phone or tablet but would not accommodate offline use or discovery through a marketplace like the iOS or Android.
  + The existing Drupal portal does have some ability to collapse gracefully but does leave room for improvement.
  + This allows for some improvements in User Experience.
  + **Disposition:** inadequate – doesn’t address discovery through marketplace, etc.

### Headless Drupal

* + Builds on previous option, requiring additional work.
  + Starting point for further front end development, which would provide improved User Experience beyond the existing Drupal portal.
  + **Disposition:** future direction, room to grow, doesn’t fit immediate timeline.



### Recommendation

Use React Native Expo to build a mobile app that runs on iOS and Android, leveraging a variety of approaches for adaptation (direct access to API, parsing API information, screen-scraping HTML from the Drupal-based COVID Portal, potentially server-side JavaScript could be added as well using a contained Node service or something along those lines.

## Design Considerations

### Components

By design, React Native is component-based, with a recommended approach and guiding principle of “use the best component for what you want to do”, rather than a more prescriptive approach like Ionic. Realistically, using a single set of UI components simplifies application of themes and leads to a common, consistent look and feel. The intent here is to create a consumer mobile app with GC branding.

### Lessons Learned

* We have been using a cross-platform React Native toolchain called ***Expo***, which allows us to deliver development work products that run on the Expo Go App, rather than natively on the phone. Before we deliver, we will need to also generate release artifacts for iOS and Android.
* Using a ***single component library*** is valuable –we are settling on React Native Paper, although we have used React Native Elements. React Native Base is also a viable option.
* We started out using cheerio.js, a ***screen-scraping library***, to compensate for early lack of access to the COVID Portal API. This may (have to) be a viable alternative to getting all data in the API.
* We have been following a guiding principle of only bookmarking product information which is provided for us in the API, but we are learning that because of the way Consumer Information is currently available, perhaps this should also be bookmarked for offline use.
* Ideally, any adaptation that we perform within the App, using cheerio or plain JavaScript could reside in a Node/JavaScript/TypeScript/Rhino ***adaptation service***. A common pattern used in enterprise mobile development leverages server-side JSON adaptation.
* ***React Redux*** is a best of breed library for API/State management, but there are alternatives that may be easier to work with, like React Query.
* ***Webviews*** were a useful stopgap, but are just too hard to work with, and we have started using react-native-render-html instead, which is a huge improvement.
* ***Accordions*** are a really good way to include a LOT of information in an easy to navigate form, allowing us to develop a single landing page for each Product, with accessible links out to the mobile browser for each of the different related Product Resources.
* ***JavaScript vs. Typescript*** and other appropriate standards. (FHIR?)

### Planned Architecture Spikes

In general, each of our Architecture Spikes has been time-boxed at a week – what can we learn and deliver as a proof of concept within a single sprint. This has been effective so far.

1. *Automate generation of development/PR work products that can be shared on Expo Go.*
2. *Use of Redux to manage information between the API and internal State Store.*
3. *Update all labels to use French and English internationalization.*
4. *Use of Redux to manage information between the API and AsyncStorage for Bookmarks.*
5. *Minimum Push Notification within the CVT App. Send a simple notification to yourself.*
6. *Move Consumer Information from Product Resource to Product Detail.*
7. *Use of Redux to manage information between cheerio and AsyncStorage for Bookmarks.*
8. *Better branding on Splash and Home Screen*
9. **Generate APK release artifact for Android.**
10. **Generate IPA release artifact for iOS.**
11. **Connect to Public API with light authentication.**
12. **Minimum Push Notification Service using Product Identifiers. Probably Node-based.**
13. **Apply universal themes to key components.**
14. **Automate process of creating release artifacts and submitting them to the marketplaces.**
15. **Use of TypeScript?**

### Anticipated Risks

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | Description | Impact | Mitigation |
| **1** | Go To Market may require jumping through some unanticipated hoops, like device testing | M | Architecture Spike |
| **2** | Loading Consumer Information from HTML not API | M | Add to API |
| **3** | Stability of API may be problematic | M | Develop middleware |
| **4** | Loading Regulatory Announcements from HTML not API | M | Add to API |
| **5** | Hard to tell Resource Types apart | L | Add to API |
| **6** | Notifications may prove challenging | M | Architecture Spike |
| **7** | API is not REST-based | L |  |
| **8** | API contains XHTML slices instead of data | M | Use of cheerio and rnrh |
| **9** | Dates in API need to be formatted in JavaScript, and use ISO rather than the current DD/MM/YYYY format | L |  |
| **10** | Use of Portal Product Details – currently, we only have a link for Consumer Information, which we can use – sort of – to determine the main Product landing page | M | Requires API change |

## Maturity Model and Major Milestones

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Source Control** | **Deployment Pipeline** | **Scope Management** | **API Support** | **Push Notifications** |
| **4: HPCI** |  |  |  | API Improvements |  |
| **3: GTM** | Public Client Repo | Automated marketplace release artifacts submitted |  | Public API | Notification Service |
| **2: MVP** | Private Client Repo | Manually release as IPA/APK for User Testing | GitLab Kanban | Public API | Push Product Notifications |
| **1: MXP** | Private non-Client Repo | GitHub Actions: Pull Request updates Expo Go | Excel spreadsheet delivered to GCCollab | Multiple Snapshot Staging extract | Push Simple in-app Notification |
| **0: New** | Local build | Local development builds |  | Single Snapshot | Visual Cues |

### 1: MXP – minimum requirements for a React Native App developed for Executive approval

Use Expo to get a quick start, including SDK for things like Camera, BarCodeScanner, AsyncStorage, Calendar, Facebook, MailComposer, Analytics. It may not be perfect, but it will get us going quickly.  
Also, select a component library

1. Multiple pages with navigation main page with vaccine info, secondary page with links out
2. Health Canada and/or Aurora Design Guide branded
3. Screens for Products, Bookmarks, Product Details (Consumer Information & Product Resources)
4. Real-time information access (based on a hosted snapshot of JSON data)

### 2: MVP – minimum requirements for a React Native App developed for Publication

1. We have built screens for Products, Bookmarks, Product Details (with embedded Consumer Information and Product Resources), and we have outstanding work for working with:
   1. Regulatory Announcements (requires API support)
   2. Recalls and Safety Alerts (requires API support)
   3. We could either link out to the PDF Product Monograph or just link back to the Product landing page in the COVID Portal.
2. Switch to Production API
3. What’s New, Help, Privacy Statement, About the App, Provide Feedback
4. Push Notifications for New Products, Recalls & Safety Alerts, Product Content changes
5. EAS Process, or some other way to resolve our App into an APK/IPA
   1. Test APK generation
   2. Test IPA generation
   3. Test Eject to pure RN app – we may choose to defer this anyway.

### 3: GTM – this represents going to market

1. Deliver to Marketplace and Lock-down API somehow.

### 4: HPCI – Additional Features (Out-of-scope for CVT):

1. Barcode scanning
2. REST-based API
3. Actual Search, rather than Filtering

Downsides to Expo: 25MB+ application due to one size fits all, can’t add native modules, Expo Eject  
<https://hackernoon.com/how-expo-is-fooling-everyone-mc1r34i6>