2016

Help Me! A Consumer Product Assistance Application

Ryan Kingsley  
*Grand Valley State University*

Follow this and additional works at: [http://scholarworks.gvsu.edu/cistechlib](http://scholarworks.gvsu.edu/cistechlib)

**Recommended Citation**

[http://scholarworks.gvsu.edu/cistechlib/238](http://scholarworks.gvsu.edu/cistechlib/238)

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.
Help Me! A Consumer Product Assistance Application

Ryan Kingsley
4/25/2016
## Table of Contents

Abstract ........................................................................................................................................2
Introduction ..................................................................................................................................2
Program Requirements ..............................................................................................................3
Implementation ..........................................................................................................................4
Conclusion and Future Enhancements .......................................................................................12
Appendices ...............................................................................................................................13
Abstract

Technology is a key enabler for people to learn new things. Most people do not know how to find these resources using the technology that is available to them. Since the birth of the World Wide Web (WWW) over 25 years ago, Tim Berners-Lee’s goal was to provide a way to share information and ideas between scientists and universities across the world. Since that time the World Wide Web has exploded with data and people can easily share knowledge with each other. Today many colleges offer lectures, courses and whole degrees programs on the WWW. Today with all that information and advances in technology, many users still do not know what information is available and how to access it. Today people can search using Google or look for videos on YouTube but many users cannot effectively find help themselves.

For a company, price is not the reason for customer churn. Poor customer service is. What if customers could easily get help without having to talk to customer service? Help Me! Allows users to scan bar codes, take pictures or enter product names and find tutorial help videos to assist them to find these resources. Help Me! Is deployed on Heroku using Ruby on Rails for a back-end Rest API. Following the mobile first philosophy, apps have been created for the Android and iOS platforms providing users tutorials and help with their products.

Introduction

As a technical person over the years I have be inundated with questions from all kinds of family asking, how do I install my router? How do I use this feature on my phone? I know for me personally I am tired of all the questions. I do not have time to be the personal IT for everyone in my family. I found that the problem is that most people do not understand all the information that is on the Internet that could help them with their daily questions. For example, my wife loves to find home improvement projects for me
and my Father in law to do. While he has a lot of knowledge in the home improvement area he does not know everything or understand every product. Many times I have gone to YouTube to find installation videos. When we were installing my garage door opener the manufacturer had a step by step tutorial on how to install the garage door. This was much clearer than looking at the paper instructions. Had we not been able to find the video we might have returned the product because we did not know how to install the product. My Father in law never would have thought to look on YouTube for help.

All the apps that I was able to find in both the Android and iOS app stores do not focus on the general help or tutorial videos. Those apps are concentrated on a particular skill. For example, Learning to Program C, Weightlifting, or Makeup Tutorials. While these apps have their use but are not general enough to help users connect with their products.

The goal of this Help Me! is to help connect people with the help content to better use and install the products they own. To do this the user has to be able to easily enter the product they have and the type of query. They are able to find videos by first entering the query type next by entering the product name, scanning the barcode, or taking a picture of the product. Help Me! Will look up the product and find videos to play.

Program Requirements

This project is interesting because of all the courses that this project uses knowledge from. Because of this the features in any one area are not dug into as deep. The requirements are below.

- Users shall be able to use this product on iOS and the Android ecosystem.
- Users shall be able to find videos by scanning the barcode of a product.
- Users shall be able to find videos by taking a picture of the product.
- Users shall be able to find videos by entering the product name.
- Users shall be able to select the query type.
- Users shall be able to play the video from the application.
Users shall be able to see the description of the video before playing.

The backend API shall be deployed through a cloud service.

Implementation

Overall Design

In the design the Backend API handles all the requests from the mobile apps to any third party service. The reasoning for this design will be discussed in the Backend API section below. The Backend API abstracts away the third party services so in the future they could be replaced.
Backend API
Since both Android and iOS were implemented, I wanted to implement a backend rather than directly use the YouTube or Outpan barcode APIs because I wanted to abstract away what services that were being used for the future if it was determined to swap out the service providers for videos, barcode, and image information. Since, Facebook has announced they are shutting down Parse I was worried about being tied to a Backend as a Service provider that in the future the backend api would have to be completely rewrote.

Using Heroku is a simple cloud platform that provides the Infrastructure as a Service. This allows deployment of a Ruby on Rails server that is being used as backend api. At the moment this backend is not saving data in the database since Heroku limits the amount of data can be used for free. Deploying on Heroku is easy if you are using get as your source control. The following commands are all it takes once you have a Heroku account:
`heroku login`
`heroku create`
`git push heroku master`
Creating your database:
`heroku run rake db:reset db:create db:migrate`
While this is simplified because Heroku you have to use Postgres as your database and all the settings to deploy your Ruby on Rails app to production need to be added.
Heroku provides a nice dashboard that allows you to look at the logs of your application.

**Figure 2. Heroku Logs**

**Figure 3. Heroku Database**

While Heroku does not scale as well as other cloud providers using a dns provider to point to Heroku would allow the backend to move IAAS service providers and be transparent to the mobile applications. Eventually moving this backend to Amazon AWS or the Google Cloud platform will scale this application for larger workloads. With apps that are not currently deployed to either Android Market or the Apple AppStore paying the larger scale does not make economic sense.
The Ruby on Rails backend api has limited amount of storage with the deployment to Heroku. At the moment the queries and images are not being stored and only has a limited Rest Api available that does not comply with all the CRUD operations at the moment. Below are the limited calls available.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>GET</th>
<th>POST</th>
<th>PUT</th>
<th>Destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>searches</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>upc_lookups</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>images</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 4. REST API Endpoints

Images endpoint does not currently save the data since this would quickly go over the limited allowed by Heroku. Future deployments would stored the data to allow this application to be smarter adding machine learning elements.

Mobile Applications

Search Screen
Uses just a basic search bar and some predefined queries that a user can pick from that are applied to all the different search input modes. Once the user enters the product name they can select the query. From this screen does a GET request which the back-end service does a GET request to the YouTube v3 API. The results populate the list of videos that feed into the List View. Figure 5 through Figure 9 show the screens walking through a text search.
Figure 5. Manual Text Entry

Figure 6. Waiting for Request

Figure 7.
Figure 8.

Figure 9.
Figure 10 shows the basic calls that are made through the back-end that complete a search.

Barcode Scanner

The Barcode tab allows the user to scan the barcode of the product. This feature uses the AVFoundation that is built into iOS to scan the barcodes and the Android GMS.Vision library to do the same in Android. The green line appears to show the user that a barcode has been found. Both applications use the endpoint upc_searches request to the backend Rails server. The Rails server is using Outpan API to find the product name from the EAN-13. Figure 11 shows the calls that are done to complete a UPC lookup.
Capture Image
This feature provides the most challenging implementation of this project. A large amount of time was spent trying to implement storage of images to the Backend API. Instead of spending more time implementing this feature from scratch I used the paperclip gem that handled the issue of storing the image and provided the URL of this image. Having the URL is important because that is the easiest way to use the Google Reverse Image Search. Unfortunately, Google no longer provides an API for this feature. That has long since been deprecated. I was attempting to scrape the page for the product information. Without providing a User-Agent in the HTTP GET request header Google sends a redirect loop. After hours of being stuck in the redirect loop I was finally able to figure out the User-Agent issue that was a quick fix. Implementing the Multipart HTTP POST was also difficult until I found AlamoFire cocoa pod that simplified this implementation. On the Android side Volley to complete the HTTP calls. Volley is an Android Open Source Project development that includes cache management as one of its many features. One of its main use cases is for populating the User Interface asynchronously.

The user selects a query type on the search page and next selects the camera tab. Next, the user takes a picture. The app uses a HTTP Multipart POST to the Backend API. The back-end sends the image URL to the Google Reverse Image Search API and
scrapes the data from the HTML response to find the image. Next, the app takes the product name and does a search to receive the video content.

**Play Videos**

Each application is able to play the videos. On the Android side the app uses the YouTube Player intent to play the video using the native YouTube app. While the iOS side uses the a cocoa pod call YouTubePlayer to play the video.

**Conclusion and Future Enhancements**

Implementing the front-end and back-end of this system was a daunting task. I ran into many issues including but not limited to learning to deploy to Heroku, using Google Reverse Image Search in a non-conventional way, learning Android Development and implementation of a multipart HTTP POST request. All of these individual issues are not a great deal of work but having such large scope was a challenge.

Creating the back-end and multiple mobile apps left a lot of future features that need to be implemented. The list of features are below:

**Back-end**

- Full REST API for the components - This allows searches to save results and videos that are useful for a product. Allow UPC to save barcodes and if not found allow the user to input the information.
- Storage - Store results in back-end database.
- Analytics of what videos are useful for a type of query.
- Move to Amazon AWS.

**Mobile**

- Deploy to App Store
- Show user product choices if matches do not show up
- User History - Allow user to save and search past history.
- Save useful videos - Allows user to save a list of videos that they found useful.
- Social Media integration - Allow users to share content directly from app.
- Add a recommendation system for useful videos - Add a social aspect of recommending videos for a product.
- Enter barcode information if missing.
Appendices

Open Source Libraries

**iOS**
- YouTubePlayer [https://github.com/gilesvangruisen/Swift-YouTube-Player]
- Alamofire [https://github.com/Alamofire/Alamofire]
- SwiftSpinner [https://github.com/icanzilb/SwiftSpinner]

**Gems**
- Paperclip [https://github.com/thoughtbot/paperclip]

**Android**
- Volley [http://developer.android.com/training/volley/index.html]

**Icons**
- The Noun Project [https://thenounproject.com/]