Nora

A global wishlist that finds the cheapest price of your product

Sujil Maharjan
Dr. Scott Frees
Ramapo College of New Jersey
March 2017

Abstract

In today's world, there are millions of products on the internet. Big online merchandises have created a new floor for marketing and exponentially increased the sales leading to lower prices of the products we see online. There are thousands of merchants selling the same product creating a sense of competition. This competition has led to multiple prices of the same product. This project solves the problem of finding the cheapest price of a particular product on the internet among various online retailers. Nora is a global wish list android application which does not bound our product preferences to a single domain like Amazon or Walmart. It is a wish list of all the products that you would like to buy saved directly from your phone's camera. It automates the process of searching the products from your wishlist through various sellers online and find the cheapest price from them. This method makes the process less hectic and financially beneficial to the users.

Introduction

In today's world, there are millions of products on the internet. It has made our lives easier as we can easily search for the products and get it delivered to our home. Big online merchandises have created a new floor for marketing and exponentially increased the sales leading to lower prices of the products we see online. Since it is a free and open market on the internet, there are thousands of merchants selling the same product creating a sense of competition. This competition has led to multiple prices of the same product. There are deals and offers by various people who sell a particular product in a cheaper price for growth and publicity. For consumers, this is a good news. There is also another problem before we try to buy the product: remembering the name of the product. For example, while going to work every day, we see various things on our way but don't quite remember the name of the products. Being able to know the product and the cheapest price of that product is a great deal.

However, it is incredibly hard to find the cheapest price of the product due to these varying prices in the cobweb of the internet. Amidst the ocean of deals for the same product, each deal varies. Going through those deals is not attainable by normal consumers. It is an automated process where you go through different deals by various merchants and compare prices each time. So, this project addresses this problem of finding the cheapest price among the renowned internet market like Amazon, Jet.com, Walmart, etc. This project is an approach to try to automate the search process of the products the consumer likes and find the optimal match in relation to the prices. This project also eradicates the need to know the name of the product by simply enabling the consumers to take the picture of the product and do the rest of the work for them.

Technology Stack

The technologies used for this application are:

Flask in Python

Python is an interpreted programming language that lets us build applications effectively and efficiently. I used Flask for building the entire back end of the application. Flask is a microframework that is used to build a web service using python. I used this framework because it is very effective in handling a medium scale application. Since the client of this project is only android phones, Flask serves as the best and unique framework.

Android Studio and Java

I used Android Studio to develop the android application for Nora. I built the entire frontend using Android Studio and Java programming language. The scope of the project is not limited to android phones but it was the most readily available device during the development process.

Android Volley

For the establishment of connection with the Flask server, I used android volley. Volley is a networking library developed by Google in order to make a smooth network connection without thwarting the UI experience of the user. This library enabled me to make RESTful calls to Nora Server.

MongoDB Database

MongoDB is a document-oriented database management system which follows the No-SQL principle. It uses JSON-like documents with schemas. I used this database management system because of the simplicity of the database for small scale storage. Even though the scalability of this application is huge, the schema for storing user information is simple and doesn't require a heavy connection between schemas.

Google Vision API

It is a service/library provided by Google that enables the developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API. I used this API in order to capture the content and categories of an image into simple text.

Amazon API

It is a service/library provided by Amazon for the developers to access the retail services provided by Amazon. It is one of the significant APIs from which I extracted the prices from various sellers in amazon for a particular product in order to get the user the best deal.

Imgur API

Imgur is an application where you can share images with each other. I used this platform to temporarily store and generate the URL for an image so that I could perform web scraping of an image. I used Imgur's API in order to generate the URL.

Structure of the application

The application is divided into 3 major parts: server, clients, and database.

Server

Nora uses Flask framework to build a robust RESTful web service as a backend of the application. The server handles the user requests for processing the images they like and user's account settings. It maintains the connection with the database in order to serve the user account calls. However, for the image processing, it does not correlate with the database at all.

Image processing serves as its own entity. In order to access the image processing functionality, the client should be a verified user and send an image in base64 format. Once the image is received, the image is passed through Imgur API in order to generate the URL of the image. The URL generated is then used for web scraping in www.google.com in order to find the best match

of the picture. Once we find out the best match, I have utilized Google Vision API in order to generate tags that define the rudimentary features of the product seen in the picture. The generated texts are then utilized in order to search for the product on Amazon and find the cheapest price among the retailers in Amazon.

Client

The current client is an android application. The application is made entirely from Android Studio which is a tool built by Google for android development. The android application provides the user with frontend features. The features that are currently included are:

a. Login/Sign up for an account
b. Take a photo
c. View all the pictures taken
d. Search a product through text directly

The front end is designed for simplicity. It is built in such a way that the user is only 3 clicks away from finding out the cheapest price.

I used SharedPreferences in order to keep the session of the user login info so that the user does not have to log in every single time s/he opens the application. Once the application is open, one can take pictures. The pictures are saved both locally and in the server. It is stored on the server in order to retrieve the images in case the user deletes the application from the phone. Moreover, the photos can be accessible from different devices once multiple clients are made on the web, iphones, windows phones, etc.

The user can also search for a product by directly typing it in the search box. Then, the search terms use part of the image processing functionality to find the cheapest price of the product mentioned.

Database

MongoDB handled the data storage for the user. There are two in Mongo:

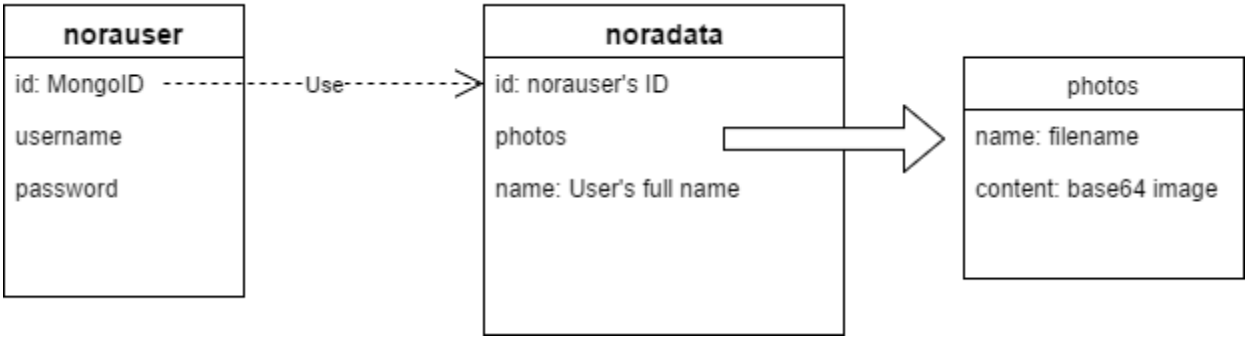
Nora User (norauser)

It stores the username, password and mongo id for the user. The mongo id is then used to refer to the user.

Nora Data (noradata)

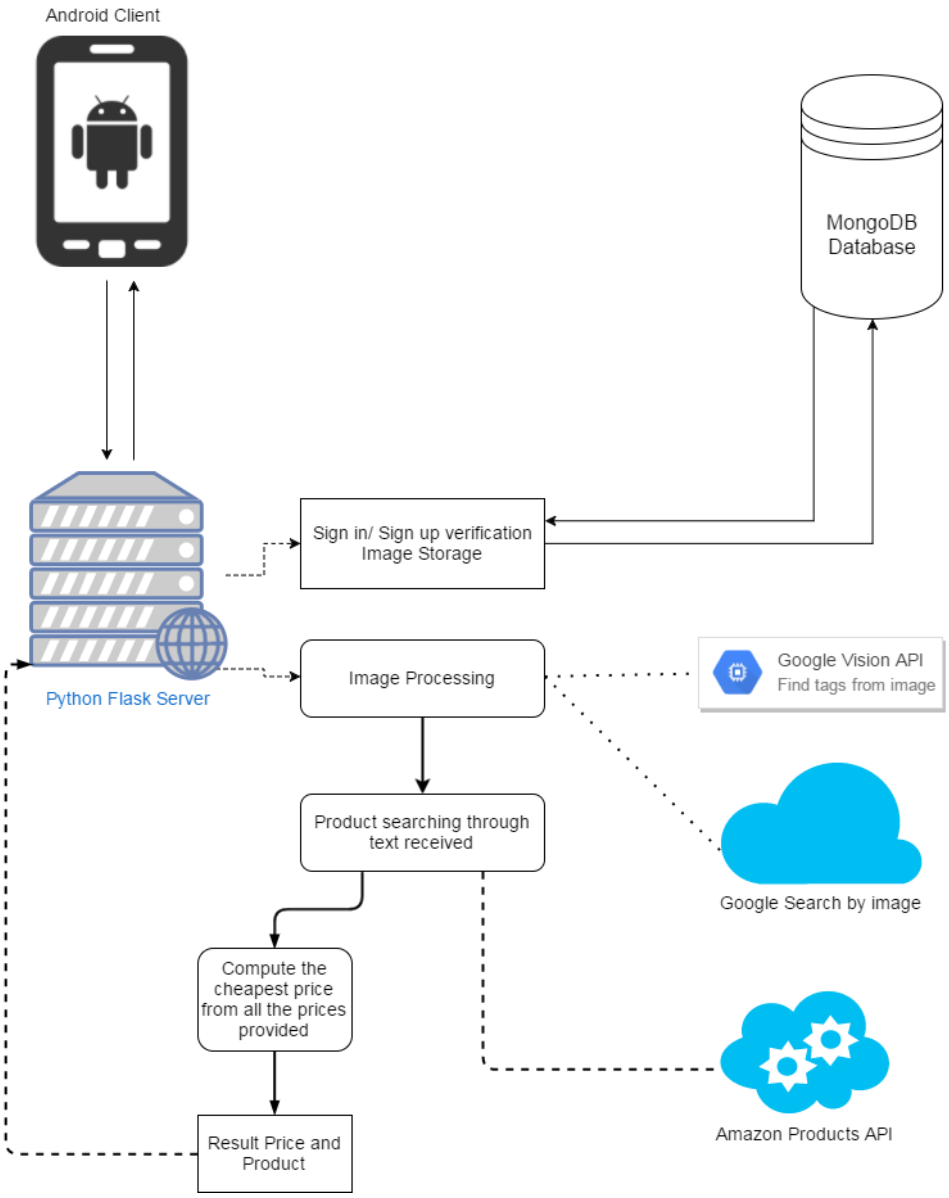
It stores the details of the user and the photos that the user have taken in base64 format. Each user has an ID which is the same as MongoID of that user in norauser collection.

The structure of the database is as follows:



Workflow

The client starts the workflow of the application. The user can easily install Nora in their android application by installing the .apk file on their phone. Once the application is installed, the user is directed to a login page. The user can either sign-up or log in from this activity.



If the user logs in,

A call is made to the server to verify if the username and password match the record in the database. The call is made by using Android Volley which takes in the Uri '/login' and user's username and password as a body. Then, the server will hash the password using bcrypt and then check it against the norauser collection in the database for matching entry. If the entry matches, the server returns a success code with the user id to the android client. This user id will be saved in UserPreferences of android. UserPreferences conserves the information of an application while

restarting the application. This information is used to keep the android user logged in and the user id is sent to the server in the header in every server call for verification purposes.

If the user signs up,

A call is made to the server with the username, password and full name of the user. Then, the server stores the user in mongo database.

When the user captures an image,

The android client uses its camera controller to take the picture. Once the picture is taken, the picture is internally stored in the android application as a JPEG file. Android provides an internal storage system where files can be stored such as only the application can access those files. Once the image is internally saved, a POST call is made to the server with the image in base64 format. On the other side, the server receives the base64 image and stores it directly to the database referring the user by their id provided from the request header. Then, it returns a success status to the android client.

When the user views the wish list,

The images stored in the android device are displays in a list view. These images are clickable which will lead to processing the images and getting the information about the cheapest price.

When the user clicks on images in the wish list,

A call is made to the server with image content as a base64 format string. Once the server receives the image, the server uses Imgur API in order to temporarily generate URL for the image. Currently, the image is temporarily stored publicly, however, it can be stored privately too. Since it is temporary, I have stored it publicly. The temporary URL generated is very important for web scraping step. There are two major components that extract the information from the user: Web Scraping and Google Vision API. Since there is no search API provided by Google in order to search by image, I am searching it using the temporary URL generated. From the search results received, I am extracting the text that relates to the product. Secondly, I am also passing the image through Google Vision API which will effectively return the tags related to the image. The tags returned defines the characteristics of the product in the image in a surficial level. Google Vision uses various machine learning algorithms in order to find out the tags of the image.

The generated tags from Google Vision and text from Google search are then sent to Amazon Products API in order to find the products. Amazon API provides the information of the products found for the text sent. Amazon also provides the list of retail sellers other than Amazon and the prices they charge for the same product. The algorithm goes through all these prices and tries to find the lowest prices from the seller. There are conditions when Amazon provides very low prices that do not define the product. The algorithm also finds these out and gives the best deal.

The lowest prices generated is then returned to the android client along with the link to a description of the product on Amazon and link to the seller who sells it for the cheapest price.

When the user searches by text,

A call is made to the server that directly puts the search text through the algorithm that searches Amazon for the product. It uses the same algorithm used while processing the image (5). Instead

of processing the image, it directly goes to Amazon and searches the image for the given search terms. The result is returned in the same fashion.

Conclusion

This project is a new way to find the cheapest price of the product we like in an efficient way. It reduces the time spent for finding the cheaper prices online by automating the process using various available web services like Amazon and Google. While the scope is currently limited to Amazon's services, it can be expanded to various other online retail market to find better prices and quality of the product.