COMPUTER VISION BASED PRODUCT RECOMMENDER

B.Tech CSE - V Sem - Software Engineering
CS300

SOFTWARE IMPLEMENTATION REPORT

NAME	ROLL NO.	PHONE NO.	EMAIL
Rajath C Aralikatti	181CO241	7829158425	rajath.181co241@nitk.edu.in
Sangeeth S V	181CO246	7907932994	sangeeth.181co246@nitk.edu.in

Abstract

Most E-commerce search engines are still largely dependent on keyword matching and the user's knowledge base to find the product that fits most into the customer's requirement. This is inefficient in the sense that the description (keywords used) can vary a lot from the buyer's side to the seller's side. In this paper, we propose adding another layer to the search criteria. This smarter search engine would basically capture an image as the input and try to classify the image into a product description. This can also make searching for a product much faster and easier. Therefore, there is room for improvement in the search process and for making the customer experience smoother. The implementation proposed is a mobile application with a fast and interactive UI that classifies an image into its corresponding product class, using machine learning, which is then used as a search query in Amazon to list the products. To build such an application that has machine learning at its core we need to incorporate a development process/methodology that is able to have a focus on the aspects of ML domain that make it fundamentally different from other software application domains.

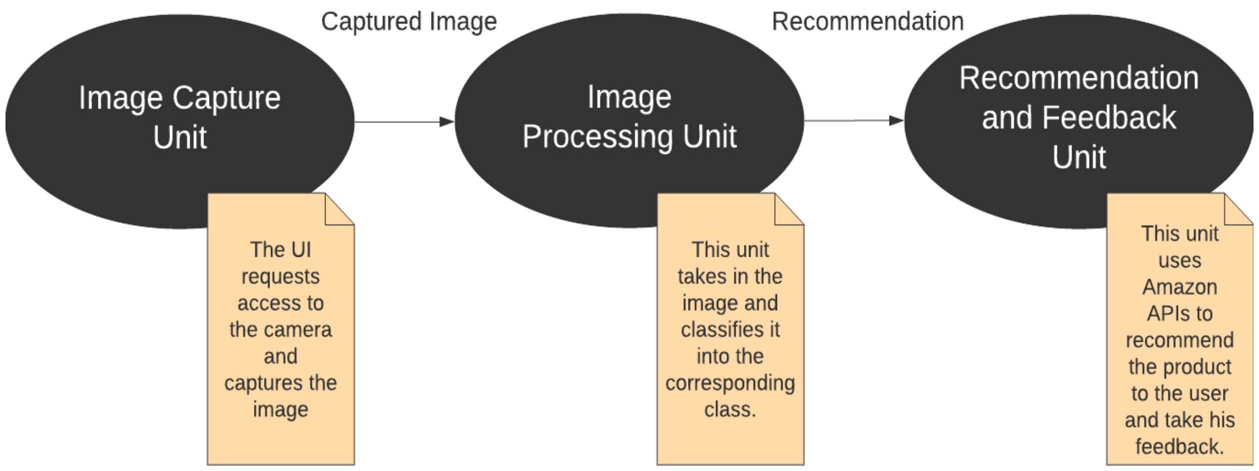
Introduction

The E-commerce system is fast-evolving and online shopping is rapidly becoming an unavoidable part of our daily lives. The latest challenge that online shoppers face is the sheer amount of digital information. This explosive growth creates an information overload, which in turn, inhibits timely access to items of interest on the Internet. Herein lies the need for a recommendation system. Almost every E-commerce website has its own implementation of a recommendation system based on available data such as recent searches, prior purchases, etc. The aim of such a system is to give the customer an efficient and more personalized experience. However, most recommendation systems are text-based and usually rely on keyword matching systems and the user's knowledge base. Moreover, the text-based description of a product can vary a lot from the buyer's side to the seller's side. With the rapid development in computer vision owing to the improvements made by neural networks these recent years, we can now make a shift from traditional word-based searching methods to searching methods involving visual similarity.

The chosen implementation for our product is a cross-platform mobile application with a fast and interactive UI that classifies an image into its corresponding product class, using machine learning based image processing techniques, which is then used as a search query in Amazon to list the products. We achieve this computer vision task of mapping the image of a product to its product class by using MobileNetV2 as the architecture for the machine learning classification model. We run the model on device and hence the computational resources of the device becomes an important factor to consider. The MobileNetV2 architecture is able to retain similar accuracies as larger models while minimizing the number of operations, making it the most appropriate choice.

Implementation Theory

The implementation chosen for our product is a mobile application for the very simple reason that it will be the most convenient for the user. The cross-platform mobile application will be able to classify and recommend products based on the image captured by the user. From the image data, the application determines what product is to be searched for. Upon successfully processing the image and producing the text-based classification of the image, the app is redirected to a list of recommended products on Amazon using the Google CustomSearch API. If the recommendation was wrong or only partially correct, then, a mechanism is available to collect the feedback which is then stored in firebase to refine the machine learning model at a later time.


Captured Image
Recommendation
Image Capture Unit
Image Processing Unit
Recommendation and Feedback Unit
The UI requests access to the camera and captures the image
This unit takes in the image and classifies it into the corresponding class.
This unit uses Amazon APIs to recommend the product to the user and take his feedback.

The image based product recommender does the following tasks,

- ❏ It uses the camera of the user's mobile device (after asking user's permission of course) in order to get the image needed.
- ❏ It then captures the image when the user taps on the capture button and the user is then redirected to a confirmation screen where the user can either continue with the currently captured image or go back to the camera to retake the image.
- ❏ Once the user is satisfied with the captured image, it is then passed as input to the image classifier.
- ❏ The image classifier makes use of the MobileNetV2 architecture trained over the ImageNet dataset as the ML model. With a softmax operation at the final output layer in the ML model we get a probability distribution for the product class over the 1000 supported classes of the ImageNet dataset.
- ❏ We then return the output classes with the highest probabilities as a list of possible product classes.
- ❏ The user can go through this list of classifications and click on the closest match. This triggers the opening of a webview within the application and redirects the user to the Amazon product list view.

- ❏ When the user returns to our application, an alert immediately pops up and asks the user if the experience was satisfactory and the recommendation was correct. If the user says that the recommendation is correct, then the app redirects to the home screen.
- ❏ If the user says that the recommendation was wrong, then he is directed to another screen with a text field which allows the user to enter a suggestion. This suggestion along with the image and the recommendation our app provided, is stored in a database maintained in firebase.
- ❏ Then, after the feedback is collected, the user is then redirected to the home screen.

Platforms and Other Requirements

The image based product recommender requires the following open-source/readily-available platforms/language support.

- ❏ Operating System: Android 8+ / IOS
- ❏ Frontend: Flutter/ Dart constitutes the codebase and SDK used to develop the application.
- ❏ Backend: Firebase for the efficient storage and retrieval of user feedback data.
- ❏ Image Processing Unit: Uses MobileNetV2 architecture for the ML model with python and tensorflow.

Why flutter?

Flutter is a free and open source mobile UI framework from Google that provides a fast and expressive way for developers to build native apps on both IOS and Android. It facilitates fast development, expressive and flexible UI and very high performance apps with minimal effort on the developer's part. Because it is platform independent, it can be used by both android and IOS developers with the same codebase and it automatically gets converted to the corresponding codebase: swift or objective C on iOS and kotlin or java on Android. This eases the effort for the developer during the initial development phase by requiring him to maintain only one codebase for both platforms. This also decreases the time and thus, the cost, of updates and maintenance of the app. Moreover, flutter is completely free and open source, which allows for the incorporation of many packages and features that the open source community has developed. The framework also gets regular updates from Google's flutter team making it one of the most popular frameworks for app development. For these reasons, flutter has been chosen as the implementation framework for the image based product recommender.

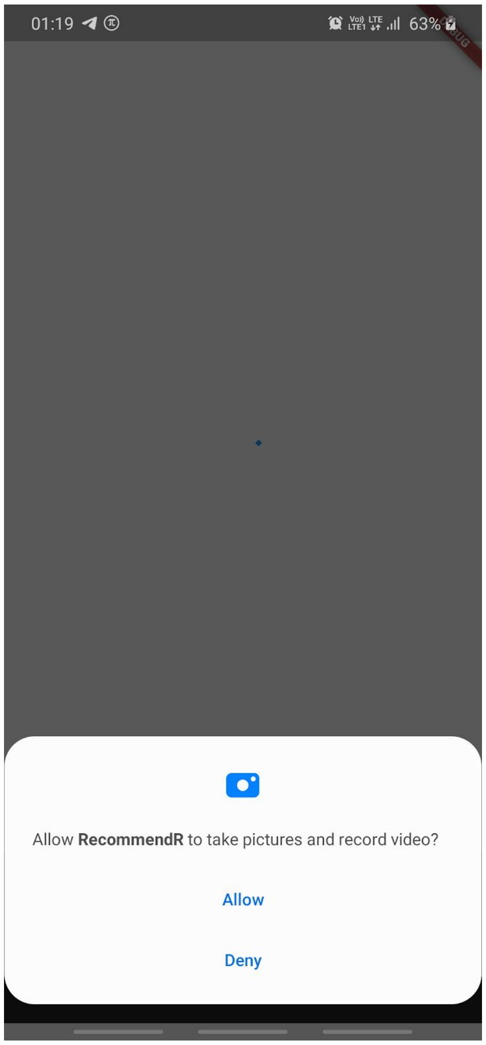
Why MobileNetV2?

In the implementation proposed, the image processing unit is run on the device and hence neural networks that require high computational resources cannot be used. MobileNetV2 pushes the state of the art techniques for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy, making it an appropriate choice for the machine learning architecture used.

Implementation Details

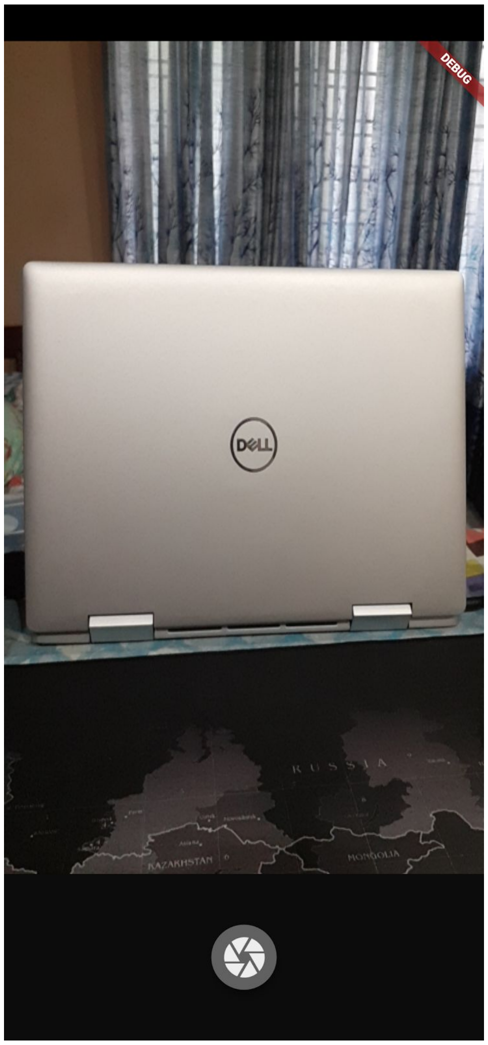
The cross-platform mobile application has been developed using flutter and dart with the tensorflow model added using the tflite package. The screenshots depicting the working of the application have been attached below.

1. <u>Camera Access:</u> The application requires access to the camera for the proper functioning. So, during installation, permission to use the camera is explicitly asked.

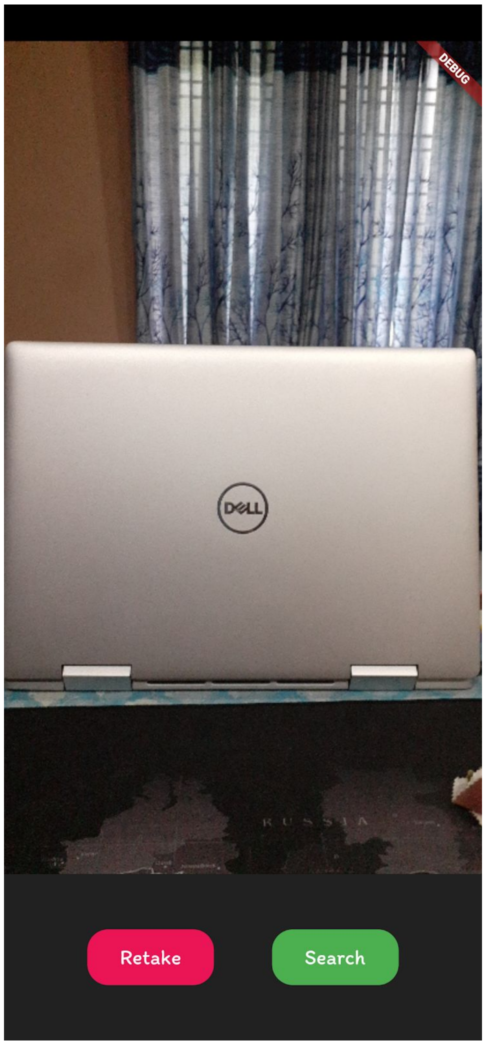


Allow **RecommendR** to take pictures and record video?

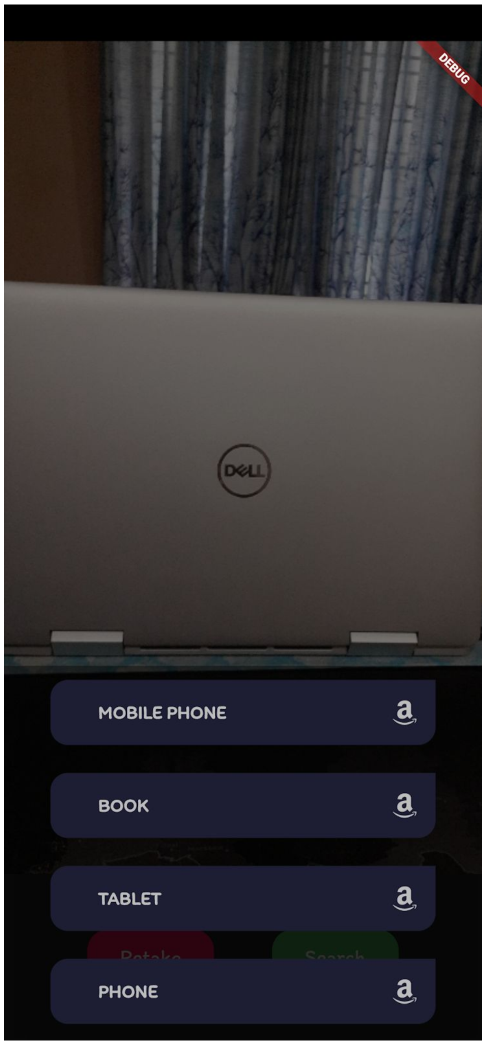
2. Image Capture Screen: This is the home screen where the app shows the camera view and provides a button, which when pressed, will capture the current camera view and create an image file and store it in a temporary location so that it can be used later whenever necessary.



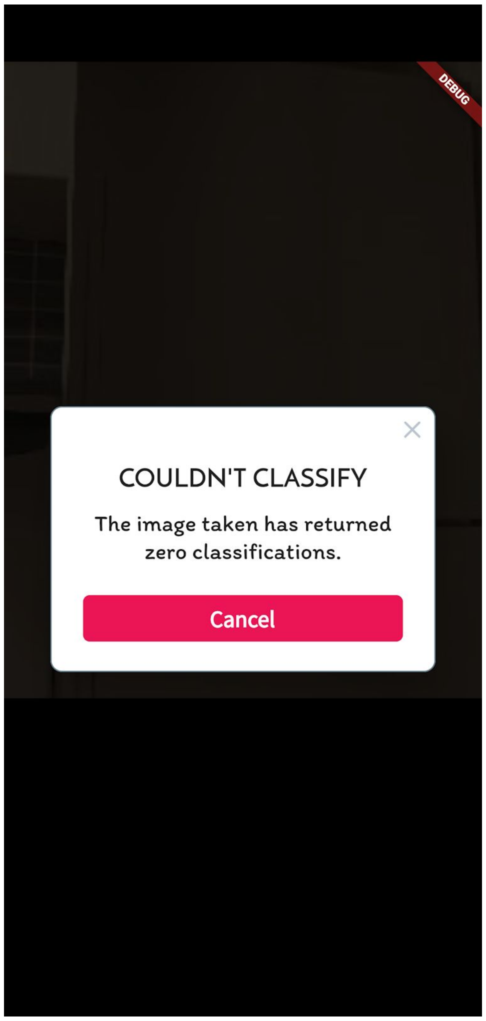
3. <u>Captured Image Preview:</u> This screen depicts the captured image and offers the user two options. If the image is not satisfactory or if it has not correctly captured the image, then an option is provided to *retake* the image and return to the previous image capture screen. Otherwise, the user can tap on the *search* button, and the app will redirect to the recommendations listing.



4. Generated Recommendations List: The recommendations generated from the captured image after it has been passed through the image processing unit (IPU) is shown in a list view for the user. The user can then select the classification that depicts the closest match. (In this example, we have selected '*tablet*' as the recommendation.)

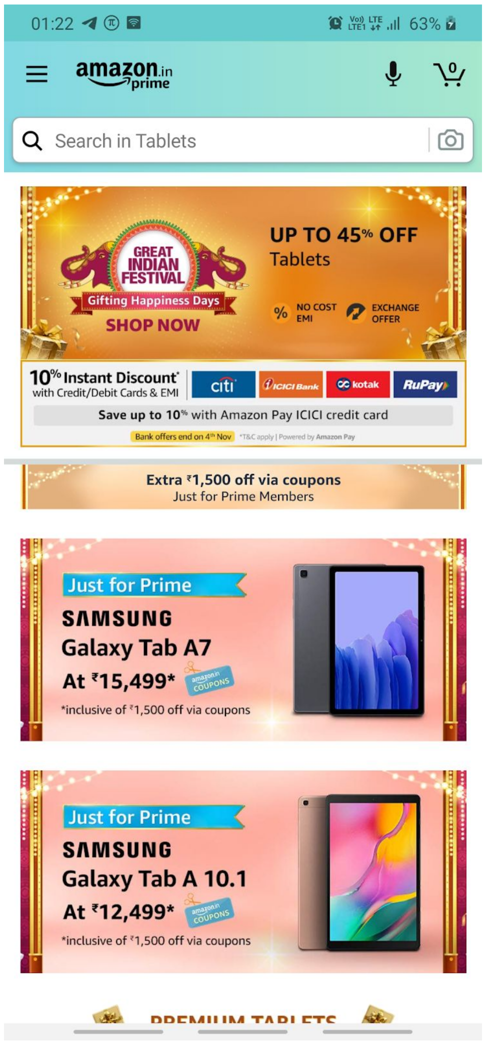


5. Unable to Classify: The implemented tensorflow image processing unit ensures that any classification that the model returns only those classifications that have a probability of at least 0.05. For this reason, there is a possibility that the model isn't able to identify any classification that has a sufficiently high probability. In this case, an alert is displayed that tells the user that the model wasn't able to classify. Upon pressing the Cancel button in the alert, the user is redirected to the home screen.

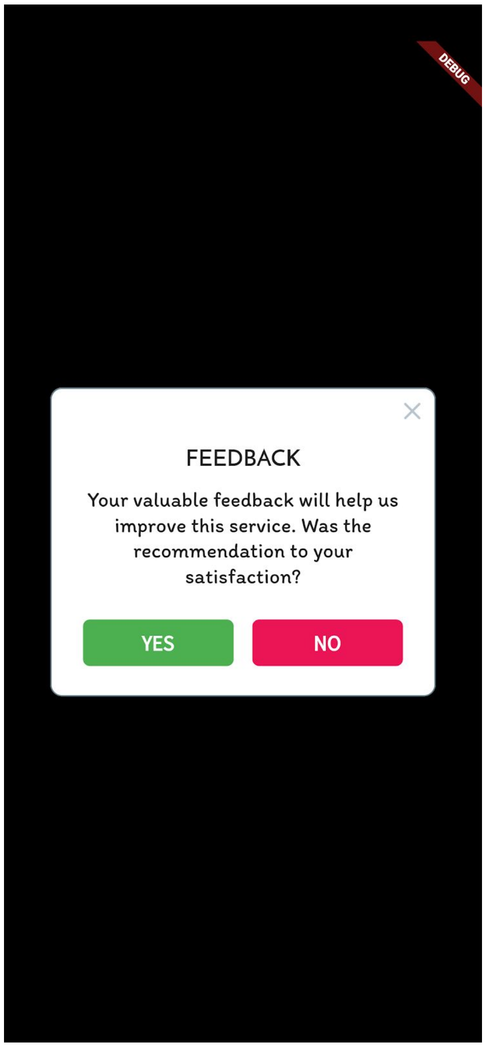

COULDN'T CLASSIFY

The image taken has returned zero classifications.

6. Recommendation Display: The text-based classification that the user selects is used to perform an API call to the google CustomSearch API with site-restrictions enabled so that only amazon.in is used for the search results. The URL thus obtained will then be used to open up a webview within the app itself. If for some reason, this is not possible, the URL is opened in the device's default browser.

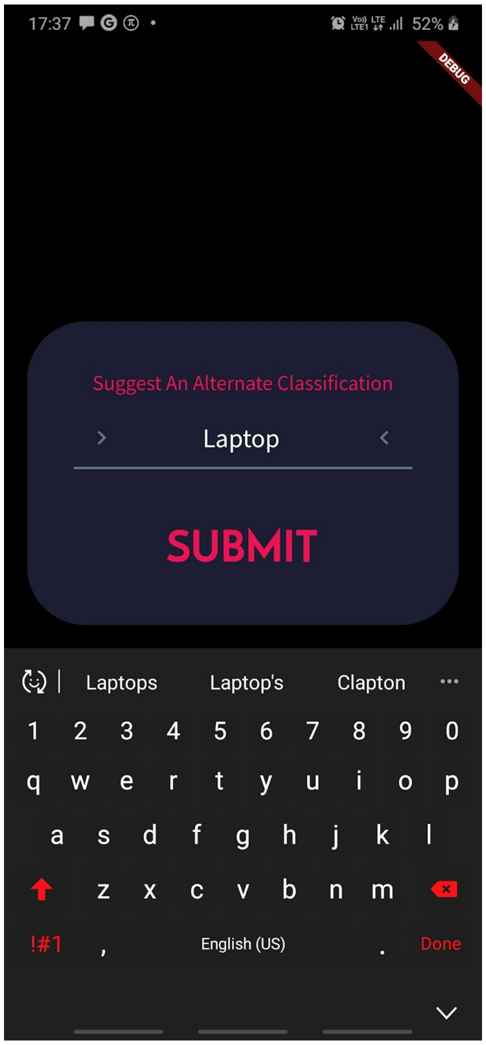


7. <u>Asking for User's Feedback:</u> When the user closes the webview and returns to the app, an alert immediately pops up asking the user for his feedback on whether the app has satisfied its purpose or if it can be improved. If the user is satisfied with the results, then he/she will be redirected to the home screen. If not, then the user is allowed to give his feedback.

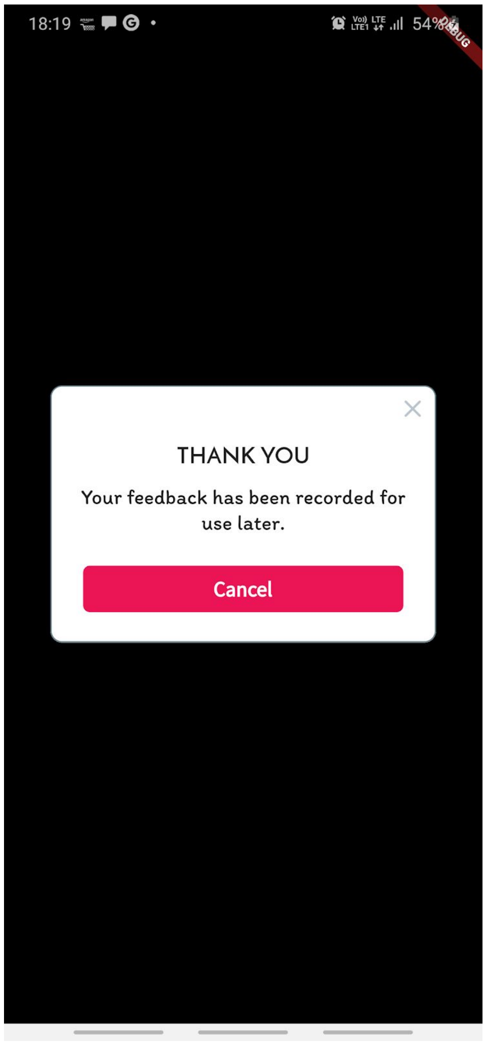

FEEDBACK

Your valuable feedback will help us improve this service. Was the recommendation to your satisfaction?

8. Feedback Screen: In this section of the app, the user is allowed to enter in a keyword / group of keywords that, along with the image and the app recommendation, gets stored in the app's firebase database, which will later on be used to modify / improve the user experience. The suggested recommendation string also goes through a cross checking mechanism to make sure that the suggested keyword is indeed valid.

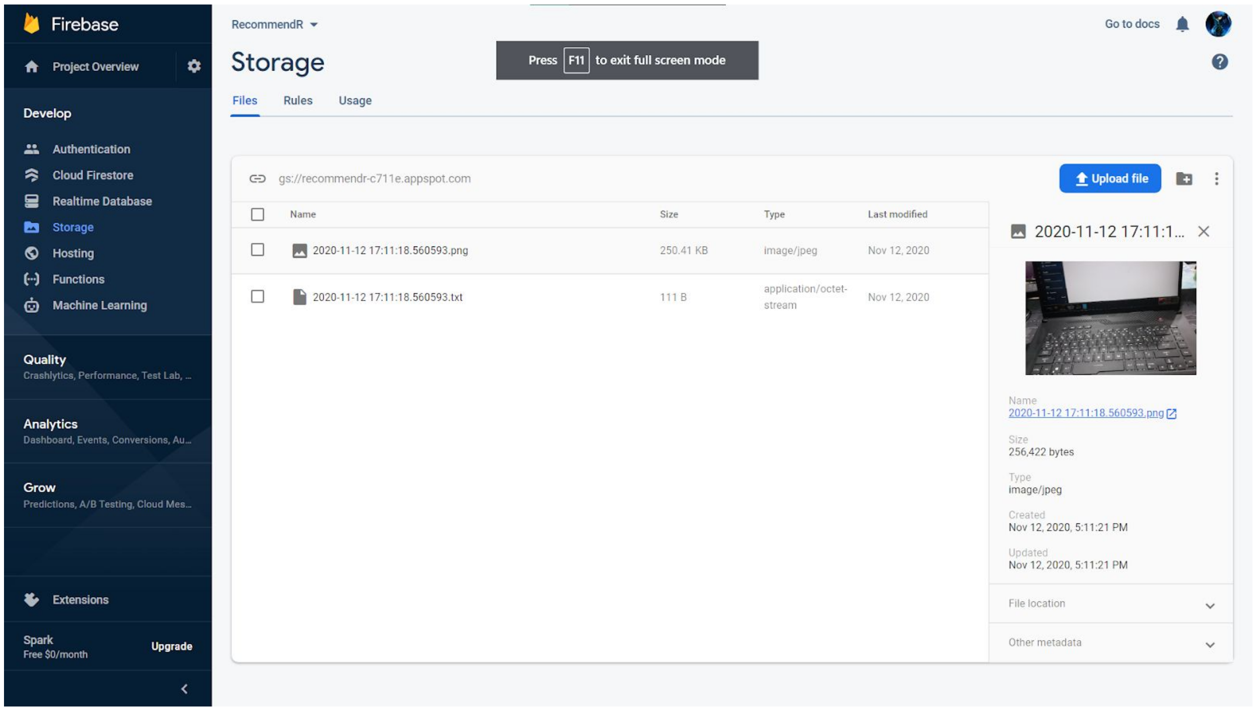


9. Thank You for the Feedback: Once the user submits the recommendation, the app redirects to a screen where the user is thanked for taking the time to provide the feedback necessary to maintain and improve our app and its user experience. The alert that pops up is automatically dismissed after a short duration, or the user can dismiss it simply by tapping on the *Cancel* button. Dismissing this screen redirects the user to the home screen.


THANK YOU

Your feedback has been recorded for use later.

10. User-Feedback gets stored in Firebase Storage: Once the user provides feedback, the image as well as a text file (named using the current time) containing the app recommendation and user-suggested recommendation is uploaded to the Firebase Storage instance. This happens in the background and so, there is no delay in the working of the app.



gs://recommendr-c711e.appspot.com

Name	Size	Type	Last modified
2020-11-12 17:11:18.560593.png	250.41 KB	image/jpeg	Nov 12, 2020
2020-11-12 17:11:18.560593.txt	111 B	application/octet-stream	Nov 12, 2020

2020-11-12 17:11:1...

Name
2020-11-12 17:11:18.560593.png

Size
256,422 bytes

Type
image/jpeg

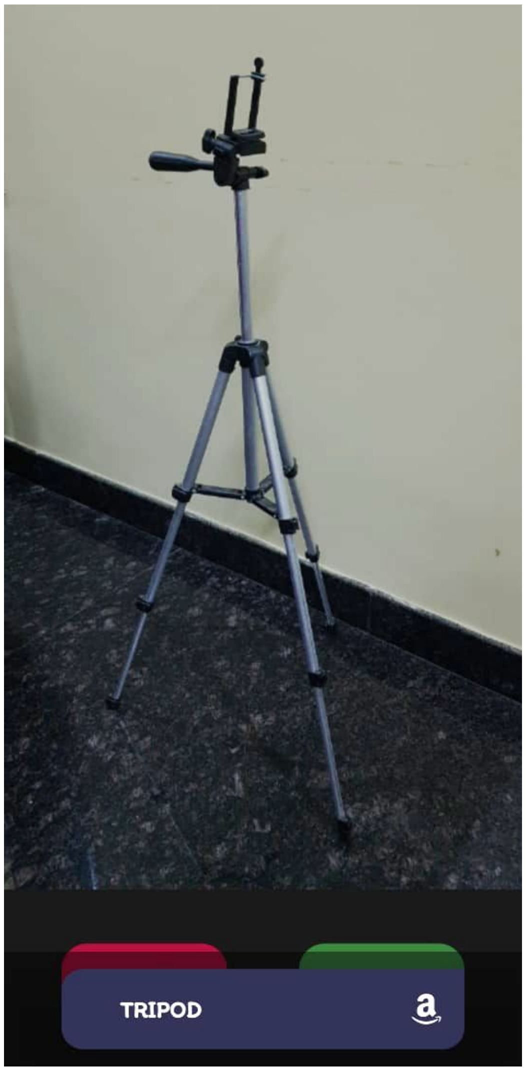
Created
Nov 12, 2020, 5:11:21 PM

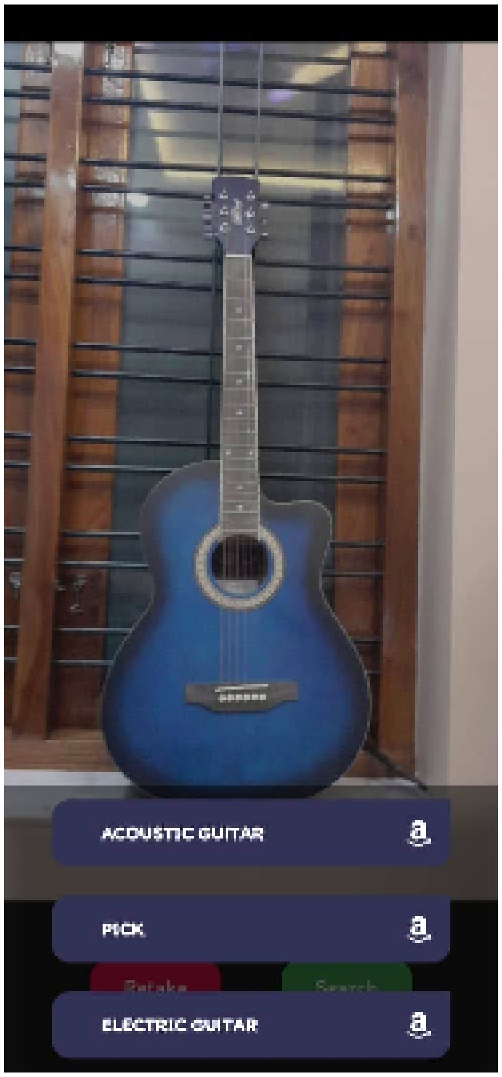
Updated
Nov 12, 2020, 5:11:21 PM

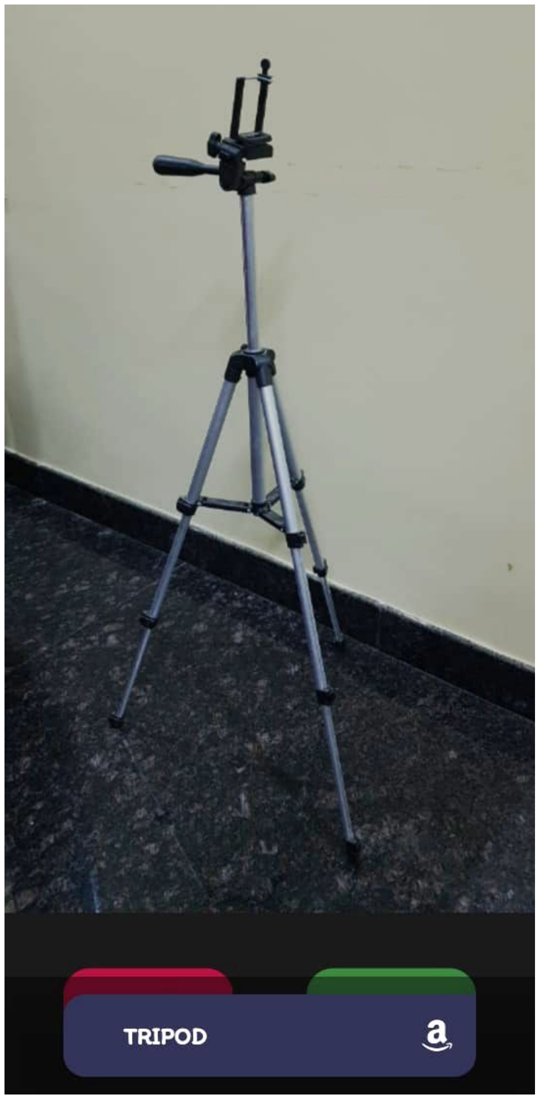
Software Testing

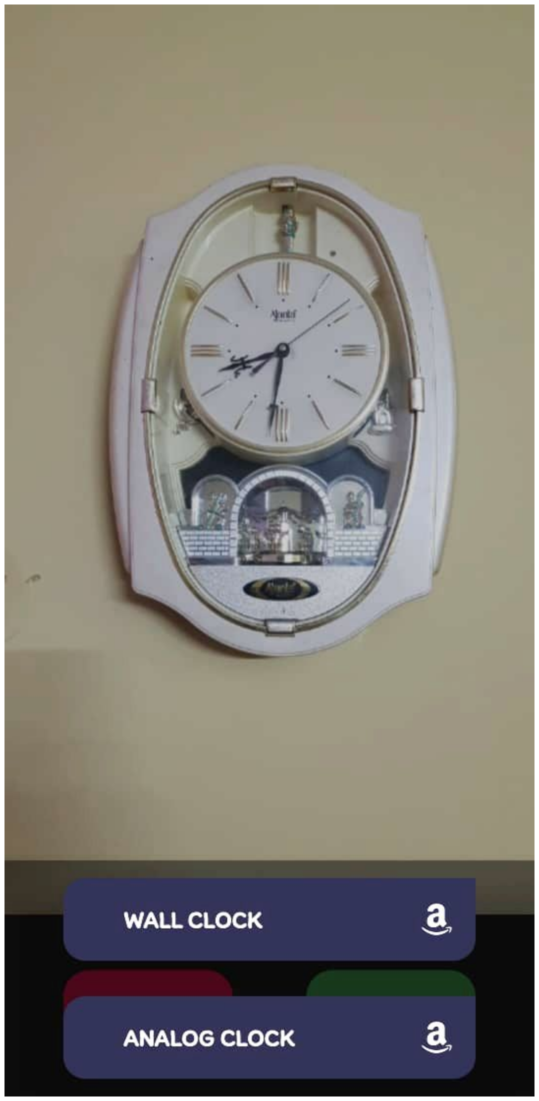
In order to perform testing of the mobile application, we first decided to run it on various different devices. It has no problems while being run on Android 10 devices but we have been unable to check on actual IOS devices although it runs smoothly on the IOS emulator.

Shown below are some of the cases where the object has been classified correctly by the application.

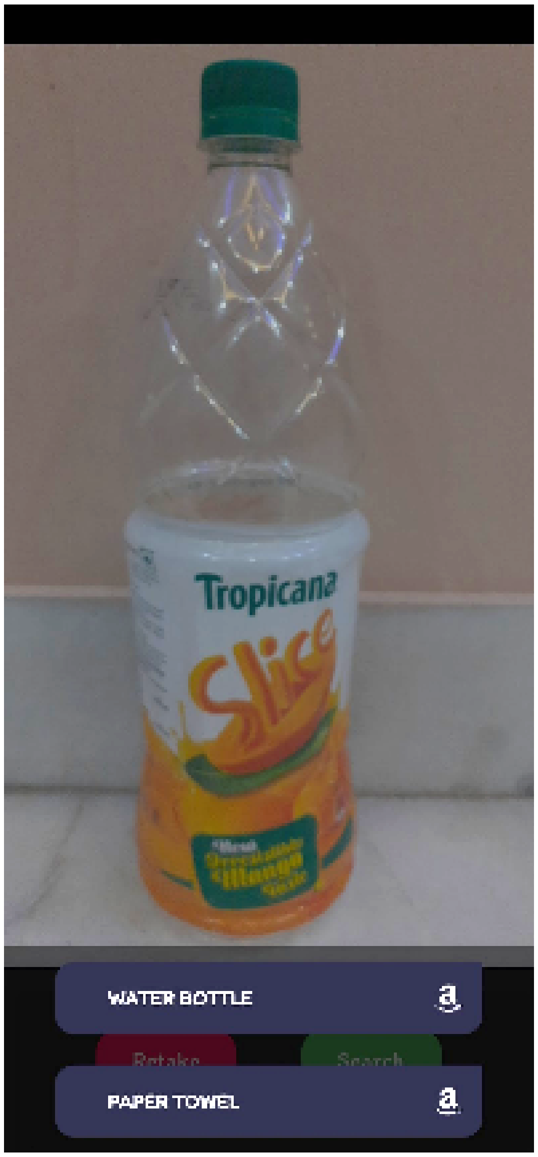


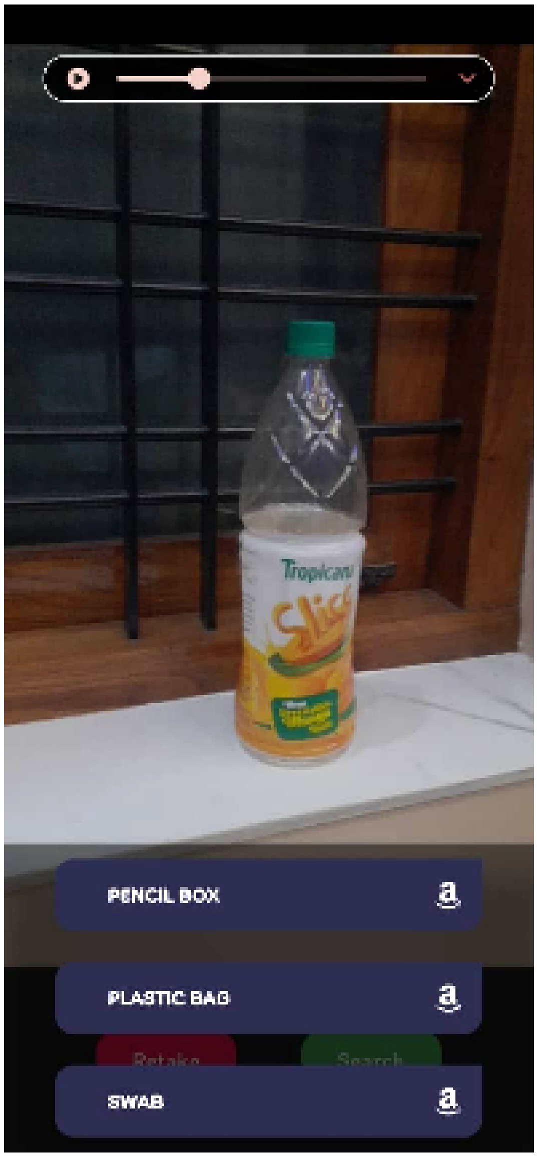



TRIPOD


WALL CLOCK
ANALOG CLOCK

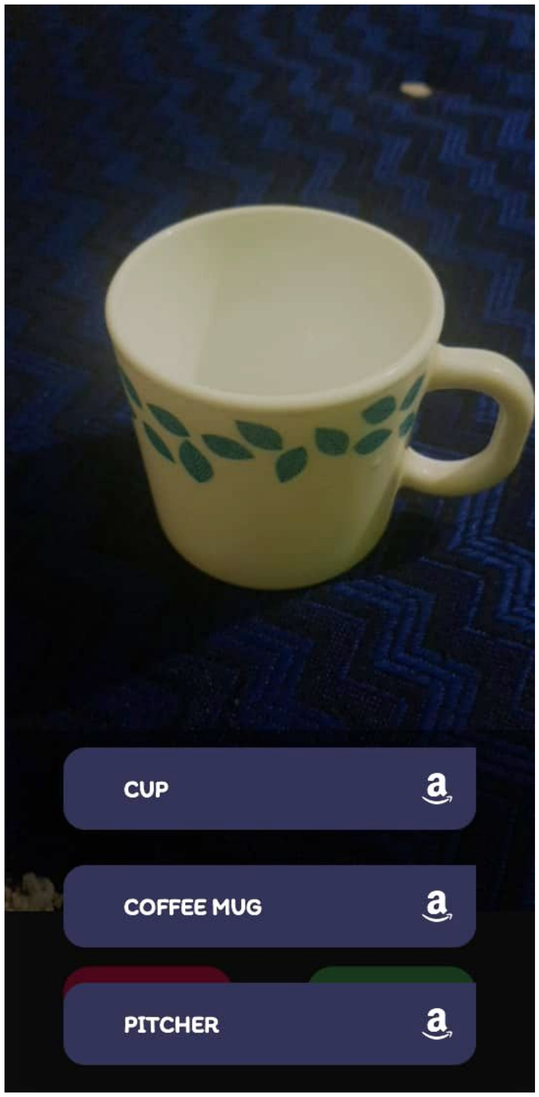
While testing the app using a wide variety of objects, we could see that as long as the object in question is placed under a plain background, the classifications that were produced were more accurate. On the other hand, if the object has a very cluttered background, the accuracy of the application declines.

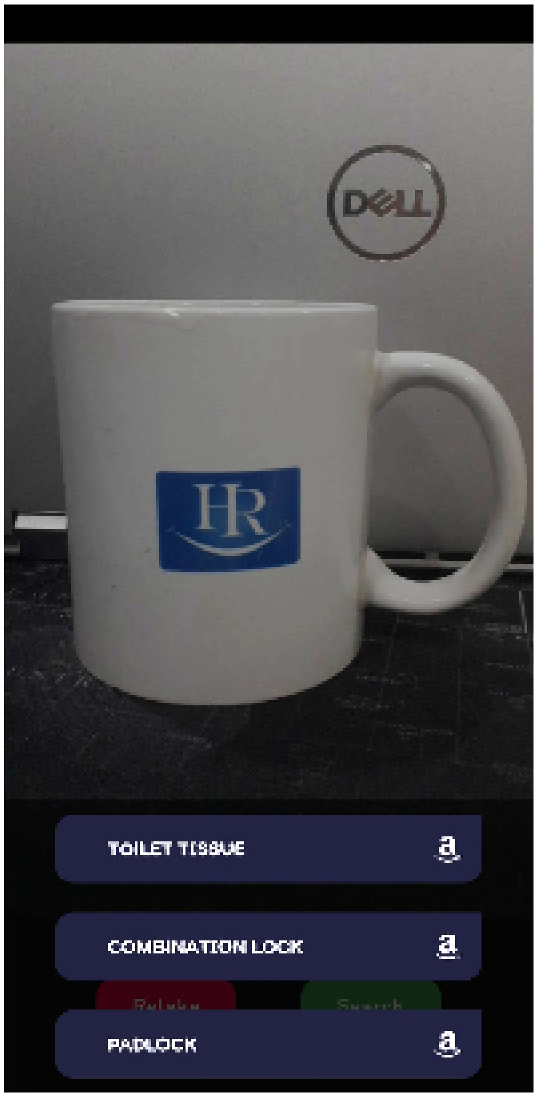




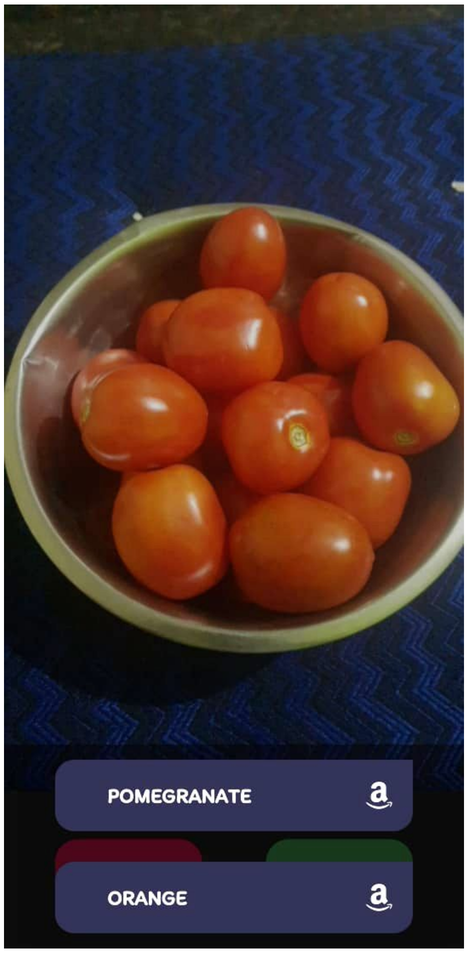
As you can see above, two images of the same object, one in a plain background and one in a slightly cluttered background have returned different classifications.

Two different images of coffee mugs both produced different recommendations. While one produced the correct recommendations, the other was wrong and confusing the object for a padlock, combination lock and toilet tissue. This is again probably due to the reason mentioned above as we see that the mug with the clear background is classified correctly while the one with the cluttered background isn't.



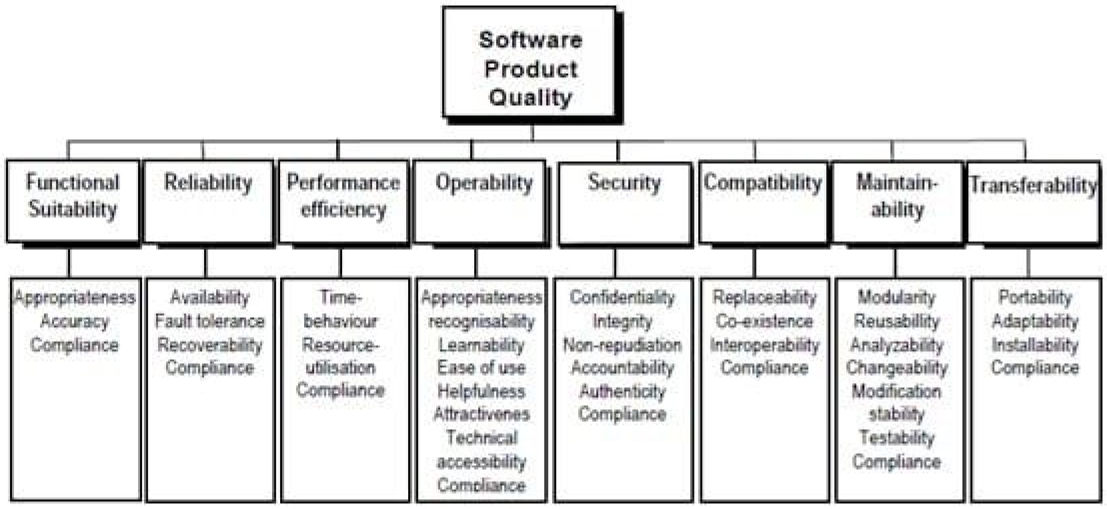


We also took an image of tomatoes which is not a supported class to examine what classification is made. The recommendations made were for pomegranates, oranges which are supported classes. These objects are similar to tomatoes and the classification that is made is reasonable.



Software Quality

In a broad sense, the term software quality refers to the degree to which the software requirements are satisfied, ie, whether the software satisfies the basic requirements put forth by the client after implementation. However, there is a difference between just achieving the requirements and efficiently doing so. To provide a standard view on Software Quality, the International Organization for Standardization (ISO) developed ISO 25010 as a model for specifying non-functional requirements. The following points describe the quality of the computer vision based product recommender in accordance with the ISO 25010 model.


Software Product Quality
Functional Suitability
Reliability
Performance efficiency
Operability
Security
Compatibility
Maintain-ability
Transferability
Appropriateness Accuracy Compliance
Availability Fault tolerance Recoverability Compliance
Time-behaviour Resource-utilisation Compliance
Appropriateness recognisability Learnability Ease of use Helpfulness Attractivenes Technical accessibility Compliance
Confidentiality Integrity Non-repudiation Accountability Authenticity Compliance
Replaceability Co-existence Interoperability Compliance
Modularity Reusability Analyzability Changeability Modification stability Testability Compliance
Portability Adaptability Installability Compliance

❑ Functional Suitability: The mobile application implemented completely satisfies the requirements specified in the requirements analysis phase. It efficiently performs all its functions as well as implements a feedback mechanism to update and maintain the software going forwards.

❑ Reliability: The tensorflow model working behind the app pushes the state of the art techniques for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy. It boasts a Top-1 accuracy of 71.8% and a Top-5 accuracy of 90.6% rating in classifying the images.

❑ Performance Efficiency: The app is fast and does not produce any noticeable delays while working. The only place where a lag is apparent is during the Application Programming Interface (API) call and this has little to do with the app and more to do with the internet bandwidth and device processing power. There are no wasted resources in the app because any such resource is immediately deallocated once its purpose has been fulfilled.

❑ Operability: The app has a very interactive user interface (UI) and consists only of taking a picture and tapping on the classification required. There is no learning curve in using this app. The other requirements state that the operating system of the device should be

Android 8+ / IOS. This is not a very restrictive constraint as most devices currently in use are up to date and satisfies the platform requirements.

- ❏ Security: The application is an open-source venture and the code is freely available on github and it does not need any authentication steps currently. If the app is scaled, suitable steps should be taken to authenticate the users.

- ❏ Compatibility: The implemented application is compatible with any device provided the platform requirements are satisfied, ie, it is not specific to any one device model.

- ❏ Maintainability: All major components of the application have been written in a well-documented, highly modular way. The codebase is written in such a way that there is very little interdependence between the different modules. Therefore, updating any single component does not pose any risk to the working of the application.

- ❏ Transferability: Flutter allows the developer to make cross-platform apps from the same codebase. It automatically builds the IOS app using swift or objective C and Android apps using kotlin or Java as specified. The apk file is just over 40MB despite the large ML model it includes and is thus very lightweight and transferable.

Conclusion

The cross-platform mobile application for the computer vision based product recommender has successfully been implemented using Flutter/Dart for the user interface and MobileNetV2 architecture for the image classifier model. The code and all related assets and dependencies can be found at: “*https://github.com/sangzzz/Image-based-Product-Recommender*”.