SE 3XA3: Software Requirements Specification
OpenCameraRefined

Group #211, CAMERACREW
Faisal Jaffer, jaffem1
Dominik Buszowiecki, buszowid
Pedram Yazdinia, yazdinip
Zayed Sheet, sheetz

April 6, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
02/09/2020	1.0	Initial requirements specification
04/06/2020	2.0	Revision 1

1 Project Drivers

1.1 The Purpose of the Project

The purpose of OpenCameraRefined is to improve the accessibility of features in the Open-Camera application. By adding the ability to capture images hands-free, and the ability to have filters while maintaining the same clean UI, the team is focused to attract more users to the refined application.

1.2 The Stakeholders

1.2.1 The Client

- Dr. Asghar Bokhari
- TAs

1.2.2 The Customers

- In this project, the customers include the clients as well. This is because the project is not set to be publicly published in an environment such as Google Store. In that case, the target customers would include students, teenagers and young adults (ages 13 - 25) who have frequent use for their Camera applications. These people are the same demographic who tend to have higher use for extra functionalities such as filters and image detection.

1.2.3 Other Stakeholders

- Core developer team of OpenCameraRefined
- Other developers collaborating on the open source Open Camera application

1.3 Mandated Constraints

1.3.1 Solution Constraints

Description: The product shall be developed in Android.
Rationale: Since the existing implementation is in Android, the expansion of the system is also written in Android.
Fit Criterion: The product shall be tested using Junit and Android Studio.

Description: The product shall be compatible with Android 4.0 or later.
Rationale: The majority of Android Devices around the globe operate on V4.0 or later (API 24 or later).
Fit Criterion: The product simulation shall take place using Android Virtual Device running 4.0.

Description: The product requires a Camera enabled cell phone.
Rationale: As an Image Capture application, the application requires a Camera for the main input stream.
Fit Criterion: As part of the product testing, the application shall run one of one the Camera enabled devices from the developer team.

1.3.2 Implementation Environment of the Current System

The product in development will be installed on the target Android device through Android Studio setup process or if agreed by the developer team, it will be installed through Google Store.

1.3.3 Partner or Collaborative Applications

The product is mainly developed using Android Studio as it will also automatcially take care of the most of the configuration left. The Android API target is 24 or later. The main companion libraries and frameworks come from Google's Machine Learning services, namely TensorFlow and TensorFlow Lite. It is important to mention, that all these frameworks are automatically configured when the application is installed on any device.

1.3.4 Anticipated Workplace Environment

N/A

1.3.5 Schedule Constraints

The schedule constraint is given in the SFWRENG 3XA3 Course Outline, where a deadline is given for each of the design documents. The same schedule is applied to the Proof of Concept Demonstration, Pitch, Final Presentation and finally the source code.

1.3.6 Budget Constraints

N/A

1.4 Naming Conventions and Terminology

1. TFLite: TensorFlowLite is a machine learning framework used to train models to predict smiling faces.
2. Bounding Box: A square box displayed in the camera view around the faces detected
3. OpenCV: A framework that provides image manipulation functions.

1.5 Relevant Facts and Assumptions

With OpenCameraRefined the team was focused on brining a more accessible application to the users, as a result the assumptions made for the user characteristics were reduced as much as possible. A few assumptions exist nevertheless, namely user must be able to have the mental and physical capacity to do simple tasks such as Smiling, Clicking the touch Screen, and showing thumbs up. The final constraint would be the hardware constraint which is the requirement of a Camera enabled Android Device.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

As an Android Application, the architecture of the product is very strict and complex. Android system initiates its program within an Activity starting with a call on onCreate() callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity. This model is loosely based on MVC architecture with the controller being the Activity class, View being the resources and widgets and model being the entities or Classes with main Business Logic. This design creates a consistent and pleasant user experience. Regardless of whether the user comes back to the app several minutes after they've last closed it or several days later, they instantly see a user's information that the app persists locally.

2.1.2 Work Partitioning

Event Name	**Input and Output**	**Summary**
1. Image Capture through Smile	image preview (in), recognition model (in), captured image (out)	Uses the recognition object from gestureController to classify the preview
2. Filter Change through Thumbs Up	image preview (in), filter matrix (in), recognition model (in), modified preview (out)	Uses the recognition object to classify the thumbs up and the filter constants to apply the desired filter
3. Image Capture with Filter applied	modified preview (in), captured image (out)	Uses the modified preview to save the picture to keep the effects of the filter

Table 2: Work Partitioning

2.1.3 Individual Product Use Cases

UC1. The user captures an image by smiling at the camera.

UC2. The user changes the filter by clicking the filter button.

UC3. The user applies a filter by showing a thumbs up toward the camera.

UC4. The user changes the current filter using thumbs up detection.

UC5. The user captures an image with the filter applied.

UC6. The user zooms in and out onto an object.

UC7. The user captures an image using the main on-screen capture button.

UC8. The user views the saved pictures, regardless of the process of capture and the presence of filters.

2.2 Functional Requirements

VP1. ~~User~~

BE1.1 ~~The user wants to take a picture using a gesture.~~

i. ~~The system shall detect the target object.~~
ii. ~~The system shall "look" for the selected gesture.~~
iii. ~~The system shall save the taken picture.~~
iv. ~~The system shall allow the user to zoom in and out.~~
v. ~~The system shall allow the user to change the resolution.~~
vi. ~~The system shall allow the user to apply a filter in "real-time".~~

BE1.2 ~~The user wants to apply a "real time" filter.~~

i. ~~The system must display a menu of filters~~
ii. ~~The system must allow the user to select from the menu of filters.~~
iii. ~~The system must show a preview of a selected filter.~~
iv. ~~The system must allow the user to close the filters menu.~~
v. ~~They system must detect and place the filter in the correct context.~~

BE1.3 ~~The user wants to edit a saved picture.~~

i. ~~The system must allow the user to simply preview the original picture.~~
ii. ~~The system must allow the user to change the properties of the picture such as contrast.~~
iii. ~~The system shall create a copy of the original picture before each edition.~~
iv. ~~The system shall offer a menu of after effects that can be reversed.~~

REQ1. The system must allow the user to capture (save) a picture by Smiling at the Camera.

REQ2. The system must be able to detect the face of the user, and distinguish between Smiling and not-Smiling.

REQ3. They system must allow the user to view the saved picture.

REQ4. The system must allow the user to perform simple modifications on the saved picture.

REQ5. The system must be able to detect a thumbs up.

REQ6. The system must allow the user to apply or change the current filter through the filter button allocated in the UI.

REQ7. The system must allow the user to apply or change the current filter through thumbs up gestrue detection.

REQ8. If a filter is applied, the system must save the picture with the filter applied.

REQ9. The system must allow the user to zoom in or out with the filter applied.

REQ10. The system must allow the user to switch from the Rear Camera to the Front Camera with the same filter applied, if any.

REQ11. The system must allow the user to take a picture with the main capture button.

REQ12. If a filter applied, the system must allow the user to capture the filtered view with the smile detection.

REQ13. The system must use auto focus in any scenario where the user requests a picture capture regardless of the process or the context of the preview.

3 Non-functional Requirements

3.1 Look and Feel Requirements

3.1.1 Appearance Requirements

ARL1. The overall look and feel will remain unchanged from the original Open Camera application. The additional features shall seamlessly integrate with the existing interface in a way that majority of users dont notice any change.

- A simple questionnaire link can be given to users after a month of use where they can rate their satisfaction with the UI along a spectrum. 80% of the users need to rate the UI score higher than the average of the spectrum.

3.1.2 Style Requirments

N/A

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirments

EU1. The ease of use of the existing features will remain unchanged.

- Users can be asked during the first few times of new use with the application to rate the ease of use of the new features added.

3.2.2 Personalization and Internationalization Requirements

N/A

3.2.3 Learning Requirements

LR1. The user shall need no learning assistance in "Smile Capture" and Gesture based Filter change.

- A study of the user requests for additional assistance through help button included from the original project should show that 90 % of users do not request for further help once the new features are deployed.

3.2.4 Understandability and Politeness Requirements

N/A

3.2.5 Accessibility Requirements

N/A

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

SL1. ~~The performance of the existing features must not be affected by the additional features being added.~~ When filters are applied, the preview frame rate should not drop substantially and the preview should be still usable.

- Test cases processed at the end of development can constantly check the frame rate of preview and assert that at any moment, the frame rate does not drop below 20 frame rates per second.

SL2. ~~Saving a photo that is being edited shall take no more than 5 seconds.~~ Capturing and saving an image with real time filters applied shall take as minimal time as possible.

- There are multiple way to measure the time between image capture command and actual image capture. Using "TAGS" in Andoid Studio is one way where we measure the time down to millisecond. The time different should be less than 2500 millisecond.

SL3. Real time camera filters shall be activated immediately after the user selects one, either through buttons in UI or gesture detection.

- 20 Instances of filter application through filter button trigger or gesture detection trigger should result in 100 % of test cases taking less than 500 milliseconds to apply the filters.

3.3.2 Safety-Critical Requirements

N/A

3.3.3 Precision or Accuracy Requirements

PA1. If a photo is being taken using the gesture feature, the app shall detect the gesture in most of the scenarios.

- The application gesture detection can be tested in 40 different scenarios incuding different lightings, different faces and generally different environemnts. The detection should be successful in at least 85 % of the scenarios.

3.3.4 Reliability and Availability Requirements

RA1. The product shall be available to use for the customer at any given time for any extended eriod of time.

- A study of users usage after the first month can show the reports of incurred problems including any times where the application has crashed. The study should show that the 0 % of crash reports have taken place before the expected availability time of 5 hours.

3.3.5 Robustness Requirements

RR1. The application's false detection should remain as low as possible, in order to prevent false image captures and extra storage usage.

- 100 images of smiling and non-smiling faces should be represented to the camera. Eventhough the model might not be perfect, the non-smiling image detection should be less than 10 % of the presented volume.

3.3.6 Capacity Requirements

N/A

3.3.7 Scalabillity or Extensibillity Requirements

SE1. The project shall easily allow for addition of new trained models where it would allow the camera to detect new gestures other than thumbs up and smiling.

- 20 pre-trained models should simply be added to the gesture controller. The application should be asked to classify the items that the models were trained with, and resutls should show that in every case, the addition of new models work as expected based on how good the model is trained.

3.3.8 Longevity Requirements

N/A

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

EP1. The product shall be operable in any location provided that the camera is functional on the users device.

- The application must be able to function and installed on 95 % of Android devices that provide Camera functionalities with the minimum processing powers.

3.4.2 Requirements for Interacting with Adjacent Systems

N/A

3.4.3 Productization Requirements

PRE1. The product must be available to download on a public GitLab repository and installed thorugh Android Studio. Further developments could present the application in an APK form where users can easily deploy the new features.

- The git repositroy is public and includes instructions on how to clone and install the application through Android Studio.

3.4.4 Release Requirements

N/A

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

MR1. The hardware and software support for the application shall remain unchanged from the original application.

- Using Android Virtual Device, we can install the original applciation on 20 different devices. The new features should then be deployed and tested again for their functionalities. 100 % of devices should not have conflic with the new installations.

3.5.2 Supportabillity Requirements

N/A

3.5.3 Adaptability Requirements

ARM1. ~~Maintenance as a result of the additional features shall be kept minimal.~~ The application must be completely functional after each update associated with Android or any of the adjacent frameworks and dependencies.

- After two years of application release, the team should study the number of users that have stopped using the application after an Android update. The number of users should be less than 3 % of the entire users.

3.6 Security Requirements

3.6.1 Access Requirements

ARS1. ~~The application shall be available to download by the general public.~~ The product should not be dependent on any file or module that is recognized by Google Play Store or different Antiviruses as a malicious file.

- Upon each update, the developer team can scan the entire project for viruses or mishandled security protocols within the system. The scan result must yield in 100 % clean files.

3.6.2 Integrity Requirements

N/A

3.6.3 Privacy Requirements

PRS1. ~~The app shall not transmit any user data or user photos to a external source unless authorized.~~

N/A

3.6.4 Audit Requirements

N/A

3.6.5 Immunity Requirements

N/A

3.7 Cultural and Political Requirements

3.7.1 Cultural Requirements

CRC1. ~~This product shall not offend any religious or ethnic group.~~

N/A

3.7.2 Political Requirements

N/A

3.8 Legal Requirements

3.8.1 Compliance Requirements

CRL1. ~~The application shall conform to all applicable laws and regulations.~~

N/A

3.8.2 Standards Requirements

N/A

3.9 Health and Safety Requirements

HS1. ~~The gestures required to take a photo shall be subtle and not endanger those around the user.~~

N/A

4 Project Issues

4.1 Open Issues

Documentation

- The original project git lacks proper documentation.

4.2 Off-the-Shelf Solutions

- Smile Capture

  An Android application that only lets you capture an image when a smiling face is in view. This application does not have any UI buttons but only a camera view.

- Native Android Camera Application

  This is the native camera application that has a hidden feature in the settings. It lets the user smile to capture a picture.

4.3 New Problems

~~A new problem might arise when a user wants to capture an image without smiling. This might create accessibility issues where the user would have to turn off the feature.~~ One of the consequences of the new features is the added processing power it takes to transform each picture through the desired filter. In addition, since optimizing these new features is out of the current scope of work, the frame rate of preview with the filters added will be reduced. This extra processing also means extra storage and addition of power usage in the phone. These are polishing problems that if the development on the application continues outside of the course plan, they can be properly optimized and fixed.

4.4 Tasks

Task	Assignee	Timeline
Import TF + OpenCV and refactor	Software Engineers	Feb 5th
Model Training	Software Engineers	Feb 8th
UI Face Bounding Box Implementation	Software Engineers	Feb 10th
PoC	Software Engineers	Feb 11th
PoC Review	Client	Feb 11th
Live Filter Implementation	Software Engineers	Feb 16th
Unit Testing	Software Engineers	Feb 21st

Table 3: Sub-tasks

Further information and details can be found within the Gantt Chart.

4.5 Migration to the New Product

None. The model trained to detect different gestures and faces will have a general understanding and will be exported in terms of weights in the APK deployed to each device. Therefore no special migration needed to export to different devices.

4.6 Risks

The only risks involved with the project are risks that come with the usage of Google's services. Most of these services are very fluid and they are constantly evolving. In many cases, the free service might turn into a paid membership or they might revamp the structure of the project forcing the developer team to change the way the services are being used as well. Some smaller risks that have to be considered are complexity of the project, difficulty of testing and finally bad programming practices.

4.7 Costs

There are no costs associated with developing the application, as it is a course project. The system will be developed with 0 $ spending and the only possible spending would be used on

Advanced Machine Learning services that we require in order to train the detector. Other than that, the only cost considered is the time and effort that has to be put on this which the team has came up with ways to minimize the difference between contribution so that everyone contributes the same amount more or less.

4.8 User Documentation and Training

None. The new features added will have intuitive user flow and easy to understand icons to help user navigate the application.

4.9 Waiting Room

One of the main features that was part of the original development plan of the team, was addition of animal filters to the face detection. Basically, that meant that very similar to popular applications such as Instagram and Snapchat, the user would be able to animate its appearance using a specific filter in real time. Although, early developments included the feature, since the model was not perfectly trained it was decided to hold off on the feature. Such a feature requires a very sophisticated the model and it was in our own interest to focus on the primary functions first and then decide to develop this feature further.

4.10 Ideas for Solutions

1. TFLite + OpenCV

    - Tensorflow lite is a framework used to make ML predictions on device. It trains the model on a computer with a dedicated GPU and then saves that model into weights and an inference that can be recompiled on the mobile device to make predictions. In our solution this method can be trained on a wide variety of images with smiling faces and various of gestures to make a highly accurate prediction. This model can be imported into the OpenCamera application where the camera view will be fed in as the input image. The output from the model will assign a label with an accuracy percentage. We will accept any prediction that is made over 85%. OpenCV can be used to add live filters into the app. This framework has built in functions that convert a BGR image into gray-scale and allow other image manipulations.

2. Face Detector API

    - FaceDetector API is built into Android 6.0 and later to help find faces in images. Since it is built into the OS, it is highly optimized and is pre-trained on Googles servers for high accuracy. Although it is highly effeicnt at finding faces, it offers APIs that can distinguish between a smiling and a normal face. Also it lacks support for detecting other gestures like a 'thumbs-up'.