

Santa Clara University Scholar Commons

Computer Engineering Senior Theses

Engineering Senior Theses

6-12-2018

Perfect Snap

Manoj Adhikari

Santa Clara University, madhikari@scu.edu

Colby Harper

Santa Clara University, charper@scu.edu

Sean Karstein

Santa Clara University, skarstein@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior

 Part of the [Computer Engineering Commons](#)

Recommended Citation

Adhikari, Manoj; Harper, Colby; and Karstein, Sean, "Perfect Snap" (2018). *Computer Engineering Senior Theses*. 114.
https://scholarcommons.scu.edu/cseng_senior/114

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 6, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

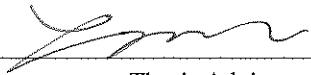
Manoj Adhikari
Colby Harper
Sean Karstein

ENTITLED

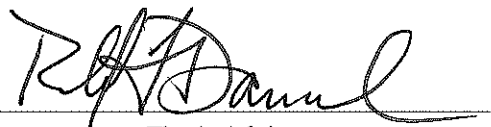
Perfect Snap

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING




Thesis Advisor



Thesis Advisor

Department Chair



Department Chair

Perfect Snap

by

Manoj Adhikari
Colby Harper
Sean Karstein

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 12, 2018

Perfect Snap

Manoj Adhikari
Colby Harper
Sean Karstein

Department of Computer Engineering
Santa Clara University
June 12, 2018

ABSTRACT

Taking group photos during important events is a common practice. Group photos are taken to remember cheerful times when people had an opportunity to meet many other people. However, an unappealing facial expression of one person can easily ruin the entire photo. Capturing the wrong moments when a person doesn't look attractive can leave him/her displeased from the complete event experience. A solution is to develop a mobile app that captures the moment when everyone is smiling with eyes wide open. Our solution aims to develop an iPhone app that will preclude users from worrying about not having a great group picture.

Table of Contents

1	Introduction	1
2	State of the Field	2
2.1	Related Research	2
2.2	Related Patents	3
2.3	Existing Solutions	3
3	Requirements	4
3.1	Functional Requirements	4
3.1.1	Critical:	4
3.1.2	Recommended:	4
3.1.3	Suggested:	4
3.2	Non-Functional Requirements	4
3.2.1	Critical:	4
3.2.2	Recommended:	4
3.2.3	Suggested:	5
3.3	Design Constraint	5
4	Use Cases	6
5	Activity Diagram	10
6	Conceptual Model	12
7	Architectural Diagram	15
8	Technologies Used	16
9	Design Rationale	17
10	Test Plan	19
11	Risk Analysis	20
12	Ethical Analysis	22
13	User Manual	23
13.1	Installation	23
13.2	Normal Use	23
14	Conclusion	24
14.1	Results	24
14.2	Lessons Learned	25
14.3	Future Work	25

List of Figures

4.1	Use Cases	7
5.1	Work flow of user interacting with system	11
6.1	Screenshot of camera user interface	12
6.2	Screenshot demonstrating the approval dialogue	13
6.3	Screenshot demonstrating the approval dialogue	13
7.1	Data Flow Architecture	15

Chapter 1

Introduction

Taking pictures to save our best memories is a common practice. During group pictures, photos can be entirely ruined by a single person's expression. We all have seen photos of people who did not smile/looked away, or worse, been that person. The experiences at big events and gatherings would be much more fulfilling if we didn't have to take many pictures just to get a nice smiling photo. Modern cameras and phones have beautiful design incorporated with advanced features. However, they have forgotten to address this persistent issue that would uplift everyone's picture taking moment.

Currently, the only group photo technology to be created is the automatic photo with timer. The purpose of this is to eliminate the need for someone to click the button in order to take the photo. However this is not a smart technology. It only has one metric which is the timer, and will take the photo when the timer has reached zero regardless of how the picture may turn out. Therefore, this does not have any effect on the quality of the photo. Currently, the only solutions affecting the quality of group photos are after effect softwares such as photoshop. However, these often require a large amount of money, and a level of skill in using the product, which most people don't have.

We propose to build an iOS app that utilizes the iPhone's camera and adds the feature of taking pictures automatically when everyone in the group is looking at the camera with their eyes open and smiling. The app will automate the tedious task of capturing the right moment saving people's time for actually enjoying the moment. Computer vision combined with machine learning will be used to narrow the focus of the image, to then allow a unique algorithm to determine when the right moment to snap the photo is.

To conclude, our design will accomplish two things: firstly, it will save time for both people who are getting their photo taken as well as the person taking the photo, as the camera will know exactly when to take the right photo, and secondly, will improve the quality of photos, allowing for memories to be better cherished by all.

Chapter 2

State of the Field

The field of face recognition is newly emerging. However, we have been taking pictures increasingly since the invention of cameras in 1816. Utilizing face recognition features in order to capture photos is a very modern approach. Therefore, there is not a lot of scientific research that describes the importance or state of this technology. As the use of face recognition features become more prevalent in our lives, we will be able to clearly define the role of our application within the bigger framework of photo capturing technologies.

2.1 Related Research

A common methodology that contributed to the advancement of Face Recognition Technology from 1994 to 2000 is the Face Recognition Technology (FERET) method. During those times, most of the common face recognition algorithms were approached in a similar way. All face recognition algorithms known to the authors consisted of two parts. 1) Face localization and normalization and 2) Face identification (1). The first FERET test took place in 1994 and established a performance baseline for face recognition algorithms. This test measured how well different algorithms could measure faces from a website. The second FERET test which was done in 1995 performed the first Alarm test. This helped measure how well algorithms did on the faces that they were not trained in. A total of 14,126 images were taken of different people from various angles which were used in FERET database to evaluate various algorithms. All of these algorithms were designed to run on desktop computers because the phones were not yet advanced enough to run these algorithms.

Linear Regression For Face Recognition research, which was completed in 2007, was a unique approach to tackle a pattern recognition problem by transforming it into a linear regression problem. Researchers utilized an underlying concept that patterns from a single object class lie on a linear subspace, in order to construct a linear model that could represent an image as a linear combination of class-specific galleries (2). This algorithm is in the class of nearest subspace classification.

Face recognition feature distinction has reached a new level due to the utilization of deep learning techniques

such as Neural Networks. Neural networks are currently at the center of object and face recognition features because they can learn and predict with much higher accuracy and efficiency than other models. Many new Neural Network Architectures are being created that are focused toward enhancing face recognition qualities. DeepID3 is one of such architecture built from stacked convolution and inception layers. This architecture can provide up to 99 percent accuracy of detecting a face.

2.2 Related Patents

There are many face recognition services patents that are granted in the U.S. These patents are based on the originality and functionality of different types of Algorithms.

Low Threshold Face Recognition compares a captured image with a stored image in order to detect if they are the same. This patent was granted to an Apple employee named Robert Mikio Free in 2011 (3).

Face Recognition Based on Spatial and Temporal Proximity is a social networking system that determines the matching faces of multiple individuals in an image or a video file. Then it presents those faces to the user who uploaded the photo or video. The patent for this invention is granted to Phaedra Papakikos and Matthew Nicholas Papakikos.

2.3 Existing Solutions

Traditional timing feature that is available on cameras and smartphones is a common way we take group photos. We can set timers for 10-15 seconds and try smiling at the right time in order to get a great photo. This approach is very inconvenient because it requires each person to be smiling at the same time as when the timer is done. Another strategy that can help us fix a photo after it is taken is to utilize photo editing softwares. Photoshop is a common photo editing tool that is widely used. Photo editing software are also inconvenient because they require technical photo editing skills and subscription fee.

Social media such as Facebook and Snapchat have utilized the face recognition features. Facebook uses face recognition in order to identify faces in a photo or a video. It then provides a feature of tagging people in a photo. Snapchat also uses face feature recognition to use filters. Snapchat's facial feature recognition has the ability to register when a person opens or closes their mouth, and can track a face as it moves in the frame. Though this technology is not used for the purposes of taking the perfect group photo, it clearly showed us that the technology to accomplish or design does exist.

Chapter 3

Requirements

3.1 Functional Requirements

3.1.1 Critical:

- Users can take pictures using iPhone camera.
- Users can choose to save or disregard the photo captured.
- The app can identify if everyone in the group is smiling with eyes wide open or not.
- Photos are automatically captured without a user's direct guidance.

3.1.2 Recommended:

- Users can share pictures with their friends directly within the app.

3.1.3 Suggested:

- Users can give feedback on whether the app captured good pictures.
- The system will offer photo editing features

3.2 Non-Functional Requirements

3.2.1 Critical:

- The user interface will be simple and intuitive
- The system will be easy to install and use.

3.2.2 Recommended:

- The system will be secure from unauthorized users.

3.2.3 Suggested:

- The system will be high performance and will require minimal system resources.

3.3 Design Constraint

- The app can only run on iPhones with iOS versions starting iOS at 11.1.

Chapter 4

Use Cases

Use case diagrams demonstrate how users will interact with our system. In our case, users will be able to take photos, then accept or reject them. Our application will only have one system user at a time, as the expected user is the person holding the phone at the time of use. The subjects of the photo will not be considered direct users, only beneficiaries of the application. Also, though the user can interact with our system in three ways, the last two options are direct results of the first, and so cannot occur unless a photo has already been taken. The system we have designed was done so with the intention of being as intuitive and simple to use as possible, and maintaining a minimal use case scenario has allowed us to achieve this.

Use Cases

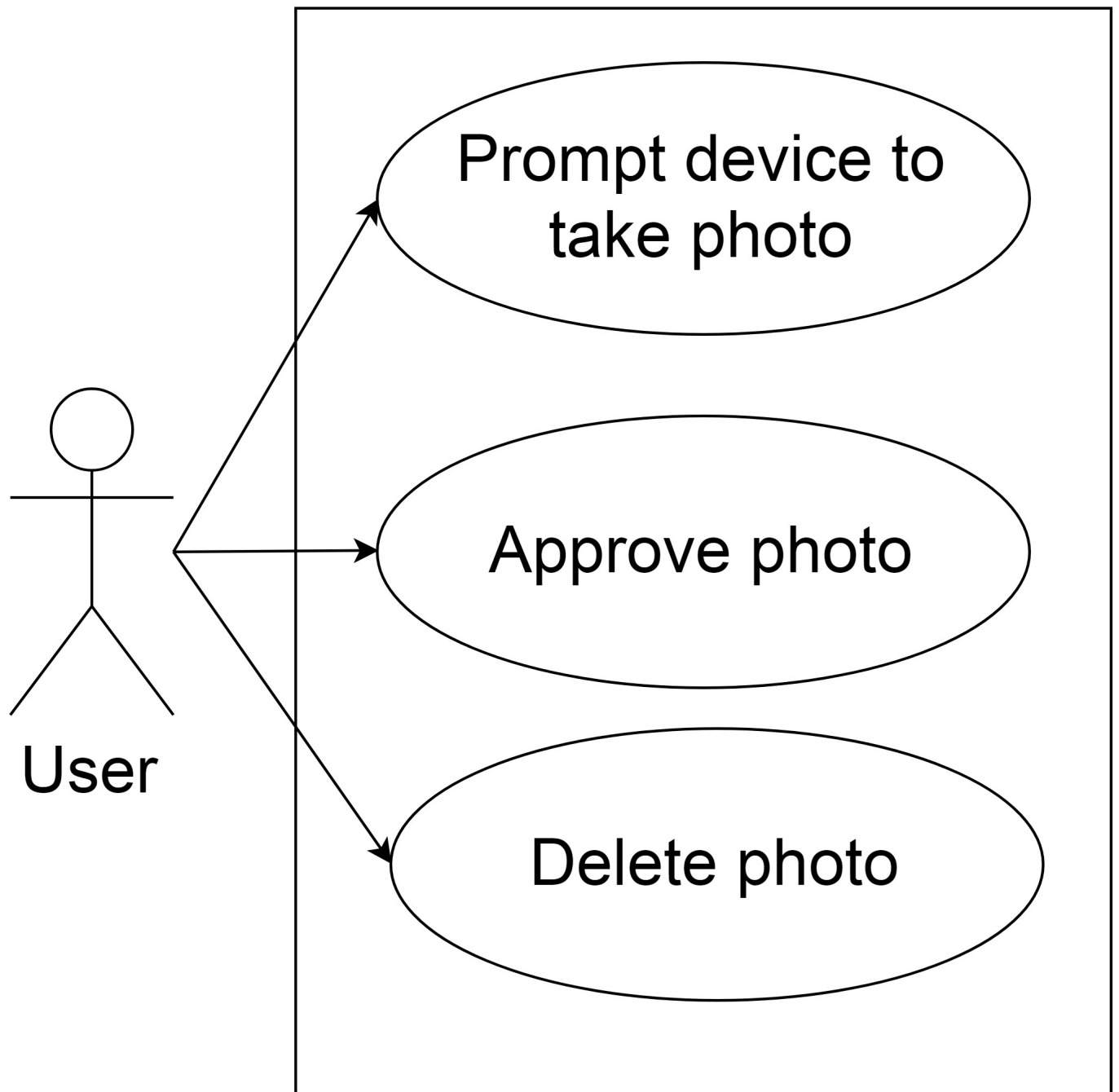


Figure 4.1: Use Cases

1. Prompt device to take photo

(a) Goal: Take photo

- (b) Actors: User
- (c) Pre-conditions:
 - i. Application must be open on device
- (d) Steps:
 - i. User presses button to prompt device to take photo
- (e) Post-conditions:
 - i. None
- (f) Exceptions:
 - i. Application must have device permission to use camera

1. Approve photo

- (a) Goal: Save photo to phone storage
- (b) Actors: User
- (c) Pre-conditions:
 - i. Must have taken a photo
- (d) Steps:
 - i. User presses accept button when prompted after taking a photo
- (e) Post-conditions:
 - i. None
- (f) Exceptions:
 - i. None

1. Reject photo

- (a) Goal: Delete photo and try again
- (b) Actors: User
- (c) Pre-conditions:
 - i. Must have taken a photo
- (d) Steps:
 - i. User presses reject button when prompted after taking a photo

(e) Post-conditions:

i. None

(f) Exceptions:

i. None

Chapter 5

Activity Diagram

The flowchart outlines the workflow of all user actions and interactions with the system. The diagram does not show system actions, such as processing the image stream and deciding when to capture an image.

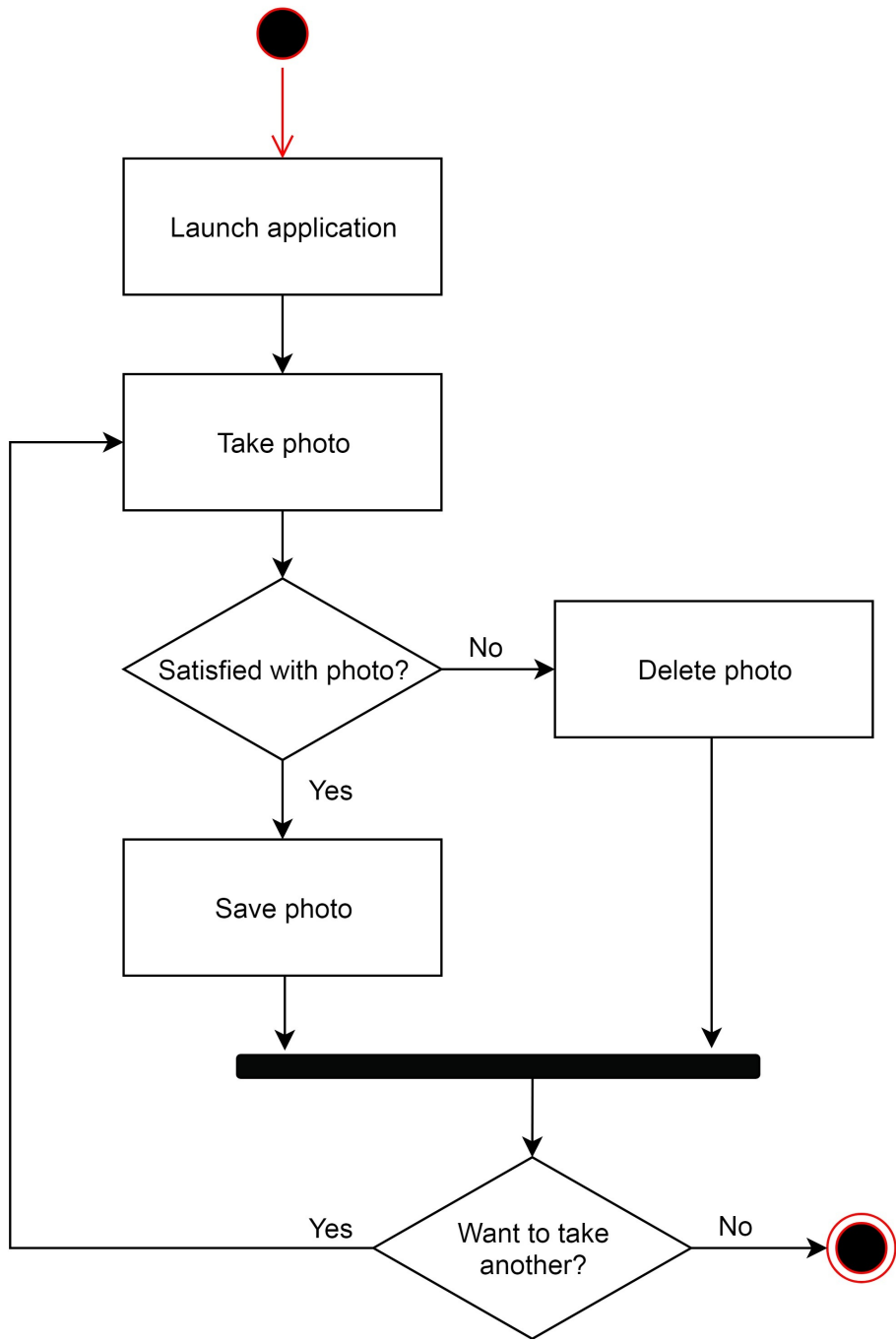


Figure 5.1: Work flow of user interacting with system

Chapter 6

Conceptual Model

Users will navigate to the application on an iPhone to begin using the system. Upon launching the app, a camera feed appears, as shown in Figure 6.1. They can change between the front and back facing camera using the button on the top right corner. Once ready, the user can touch the shutter button to enter capture mode.



Figure 6.1: Screenshot of camera user interface

Once in capture mode, as shown in Figure 6.2, the application will wait until all subjects are in frame with their eyes open to capture the image. Upon capturing the image, they will be greeted with an alert asking whether they want to save the photo or reject it, as seen in Figure 6.3.



Figure 6.2: Screenshot demonstrating the approval dialogue



Figure 6.3: Screenshot demonstrating the approval dialogue

If the user chooses to accept the photo, then it will be saved to the user's storage and they will return to the camera screen. If the user chooses to discard the photo, then it will be delete and they will return to the camera.

Chapter 7

Architectural Diagram

We utilized a data flow architecture to complete this project as shown in Figure 6.1.

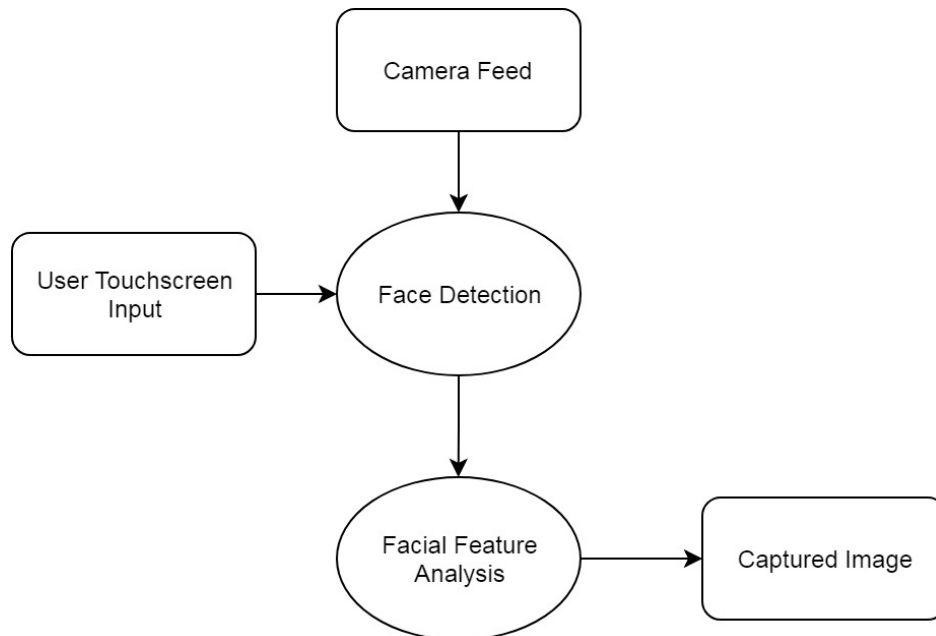


Figure 7.1: Data Flow Architecture

The first main source of data is input from the phones camera feed. These are passed as a continuous feed of images and can come from either the front or back facing camera. The second source of data is the users touchscreen input which prompts the device to enter capture mode. In capture mode, our face detection algorithm scans the frame for faces and adds them to an array. Then, once faces are detected, facial feature analysis is run which scans each face for open eyes and smiles. Once the number of faces with open eyes and smiles matches the total number of detected faces, the image is captured and saved to the camera roll.

Chapter 8

Technologies Used

- Hardware
 - Development
 - * Macbook Pro
 - * ThinkPad
 - Application Testing
 - * Iphone 7
 - * ThinkPad with Iphone emulator
- Programming Languages
 - Swift
 - * For iOS Programming
- IDEs
 - XCode
- APIs
 - AVFoundation
 - Vision

Chapter 9

Design Rationale

We are using a data flow model to increase the speed at which the application will operate. We want the device to recognize when the right moment to take a picture is as fast as possible so that the perfect photo is taken. Because of this, we decided not to have any communication with a server, and instead keep all logic on the local device.

We are using iPhones and the associated iOS devices and software for testing and development because that is what most of our team uses, as well as most of the people we know at Santa Clara University. This will allow for a potentially large alpha testing process during the later stages, as well as eliminating the need to buy iOS phones for beta testing.

Additionally, using iOS devices and software allowed us to leverage some existing frameworks that worked well for the purposes of our design. Two of these frameworks were AVFoundation and Vision.

- **AVFoundation Framework** An iOS framework that allows access to audio visual media on Apple devices(4). We used AVFoundation to access both the front and back facing cameras along with their input streams. AVFoundation also enabled us to save the captured photos directly to the user's photo library.
- **Vision Framework** An iOS framework that uses a machine learning backend to apply high-performance image analysis and computer vision technology to images and videos(5). It can automatically detect faces, objects, or classify scenes, making it an incredible tool for processing large numbers of image files and videos.

The vision framework is utilized as follows: A vision request is initialized which is an abstract superclass for image analysis requests. The initialization of this request will include the constraints and requirements that dictate what the framework will be looking for in each image. Next, a request completion handler will be used to execute the request based on the desired media input. Lastly, the result of the executed request will be an observation array, which will be populated with abstract superclasses for image analysis results, constructed based on the original request. For the purposes of our application, we first create a request to detect faces and return an observation class that provides rectangular coordinates that enclose each face. We then use these coordinates to narrow down the image, and create a

new request that will be executed just on each portion of the image enclosed by rectangular coordinates. This request is a detect landmarks request and will result in coordinates that outline the mouth and eyes of each face.

Chapter 10

Test Plan

Unit testing was done throughout the development process on every new functionality we implemented. This was especially important when working in a team environment, as the code one person writes could unknowingly impact code another person wrote days, weeks, or even months prior.

Also, an extremely important part of this project was testing our eye detection algorithm on a diverse group of people to determine the best threshold value for determining whether an eye is open or closed. Testing on a larger and more diverse group of people than just our own senior design group gave us more confidence when putting the final touches on the algorithm.

Once we had a working project, we started on black box testing. We gave our application to friends and had them use it without any knowledge of the code, or specific instructions other than the general purpose of the app. We used this system of testing to evaluate our product without any bias.

Chapter 11

Risk Analysis

Table 10.1, shown below, shows the risks and consequences involved with the completion of the system. The table also shows the probability that the risks will occur along with a severity ranking between 1 and 10. The probability and severity are multiplied to receive an impact measurement. Each risk has mitigation strategies to avoid the issue.

Risk	Consequences	Probability	Severity	Impact	Mitigation Strategies
Illness	Portions of development blocked.	.4	5	2	Ensure each component of the design is understood by multiple members.
Insufficient Development Knowledge	The system will not accomplish our goals.	.2	7	1.4	Use online resources and communicate with team members when a roadblock is hit.
Coordination Failure	Components overlooked and unfinished	.15	8	1.2	Keep an organized schedule of due dates and follow the development timeline.
File loss	We lose access to our files.	.05	10	.5	Use GitHub to protect files, and always push to master when updating.

Table 11.1: Risk Analysis Table

Chapter 12

Ethical Analysis

While developing our product, there were several ethical scenarios that needed to be considered and addressed. First, we needed to acknowledge any internal ethical dilemmas that we may encounter. The first of which is ensuring that every team member has a voice that is heard during the design process. We wanted to ensure no member of our team would be ignored or blocked from the development process under any circumstances. On the opposite end, no member should be forced to perform a greater amount of work than the rest of the group. The workload needed to be distributed among team members as evenly as possible. To prevent either of these ethical issues during the design, one solution implemented was to document all necessary next steps as well as all recently completed steps when all members were present, or on an online forum that all members had access to. This allowed all members to be aware of and able to add input to every task, preventing anyone from being blocked from development information. This also allowed all members to see who had done which tasks, making it easy to determine if the work load was being divided fairly.

Secondly, we needed to acknowledge external ethical dilemmas that could be pertinent to our product. One of which is ensuring that our product works for people of all races and genders. We do not want a product that is implicitly racist or sexist due to its code. In order to ensure this is not the case, we had an extensive period of testing on a diverse group of people. We used people of different races, genders, and ages when developing our algorithm to detect eyes. We also included people with glasses to ensure that they could enjoy our application as well. Moving forward, we will continue to take steps to ensure that all people will be able to share the same experience while using our product.

Chapter 13

User Manual

13.1 Installation

- Double click the CameraView.xcworkspace file to open it in Xcode.
- Connect your iPhone to the computer and select that iPhone as the device to run the application.
- Click run from the top menu to install the application and run it on the iPhone.

13.2 Normal Use

- Install and launch the app.
- Reverse camera by clicking button at top right corner.
- Enter capture mode when ready by clicking button at bottom center.
- Hold the phone still with the people in the camera view.
- Once the photo is captured, you can save or delete.
- To save, click the button that says 'save' on the top right corner.
- To delete, click the x button on the top left corner.
- With either action, you will be taken back to the main screen, where you can either repeat the process or exit the app.

Chapter 14

Conclusion

The development of our project proved to provide many obstacles which we needed to overcome in order to create a consistent and finished product. This section contains information about the degree to which our project succeeded, lessons we learned along the way, and future work planned to improve our product.

14.1 Results

Our implemented design proved to be effective at automatically capturing ideal photos in real time. In order to evaluate our application we conducted testing on several subjects in various conditions. We found that our application performed almost perfectly under certain conditions. The first condition was distance from the camera. The application performs best when subjects are standing within ten feet of the camera. When subjects are farther than ten feet the application often does not recognize their features and fails to capture the photo. Another condition is background noise. We found that when there are additional faces in the background either in profile or straight on, the application considers them a subject and tries to analyze their features as well. When this is the case, photos are often not captured. The final important condition was the angle of faces. When faces are tilted our eye detection algorithm is not able to detect if they are open because the height and width calculations are based on absolute x and y axes. This often causes the photo to not be captured, and sometimes captured inopportune photos.

Under ideal conditions, our application is able to capture the photo within less than 20 milliseconds of the ideal frame entering the camera feed. This time is significantly faster than a humans blink which takes approximately 100 to 400 milliseconds, so we can be sure that a blinking eye will not be inadvertently captured by the application(6).

14.2 Lessons Learned

Through research, planning, and trial-and-error, we learned a lot during the development of our project. We will take these lessons into account when conducting future work in both academic and professional settings.

1. **Research technologies before committing to them.** While a particular technology may seem ideal, it is important to research dependencies and compatibility issues before beginning implementation with them.
2. **Create a realistic schedule.** It is important to accurately estimate the amount of time each task will take when formulating a schedule. This creates a feasible schedule that will be easier to stick to when development actually begins.
3. **Seek feedback early and often.** Feedback from users helps to provide valuable information about how users expect a system to function. This feedback is important to consider when trying to create a system with good usability.

14.3 Future Work

Although our project provided fairly reliable photo capturing based on facial features, there are several ways in which the project could be improved. The following list details some of the improvements which should be added to create a more complete project.

1. **Implement implicit user feedback.** In order to improve the accuracy of our open eye detection we want to incorporate implicit user feedback based on which photos users decide to save and delete. Based on this feedback we can create a metric specific to users that can help tune the values used in open eye detection so that it will work more reliably with them.
2. **Consider tilted and rotated faces.** Currently our system assumes all subjects are looking at the camera straight on. By detecting and accounting for the tilt of faces, we can more accurately perform the eye dimension calculations that are necessary for open eye detection. In addition, by accounting for the rotation of faces, we can more accurately assess facial features and capture photos more reliably.
3. **Add gallery feature.** Our current system requires users to exit the application and enter the photos app in order to view their captured photos. We would like to add a small button in the corner of the interface which would take users directly to their photo gallery.
4. **Launch on App Store.** Finally, we see a lot of potential for this concept, especially as a feature integrated into large applications. The first step in making our project more available to interested users is to launch it in the the App Store. From there we can start researching other platforms for our application.

Bibliography

- [1] P. Jonathon Phillips, Hyeonjoon Moon, Syed Rizvi, and Patrick Rauss. *The FERET Evaluation Methodology for Face-Recognition Algorithms*. IEEE Transactions on Pattern Analysis and Machine Learning, Vol. 22, No. 10, October 2000
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=879790tag=1>

- [2] Imran Naseem, Roberto Togneri, and Mohammed Bennamoun. *Linear Regression for Face Recognition*. IEEE Transactions on Pattern Analysis and Machine Learning, Vol. 32, No. 11, November 2010
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=5506092>

- [3] Low Threshold Face Recognition Patent.
<http://www.freepatentsonline.com/y2011/0317872.html>

- [4] AVFoundation Documentation,
<https://developer.apple.com/av-foundation/>

- [5] Vision Documentation,
<https://developer.apple.com/documentation/vision>

- [6] Ramot, Daniel. *Average Duration of a Single Eye Blink*. Harvard.edu, 13 July 2001
bionumbers.hms.harvard.edu/bionumber.aspx?s=yid=100706ver=0