SENIOR CAPSTONE FINAL REPORT

MAY 29,2020

GESTURE RECOGNITION KEYBOARD

OREGON STATE UNIVERSITY
CS 463, SPRING 2020

PREPARED FOR

SCOTT FAIRBANKS _____ Signature _____ Date

PREPARED BY

GROUP 11

JEREMIAH KRAMER _____ Signature _____ Date

ZACHARY HORINE _____ Signature _____ Date

LAUREN SUNAMOTO _____ Signature _____ Date

CHANGKUAN LI _____ Signature _____ Date

Abstract

This document outlines the final report of the gesture recognition keyboard application. The gesture recognition keyboard introduces a new method of keyboard input on iOS devices using gestures. The purpose of this report is to describe the project and its details, provide the initial requirements, design decisions, and updated documentation. This report also includes each team member's individual reflection, conclusions on the viability of our project, and advice for future improvements.

CONTENTS

1. Foreword: Release Notes 1.0

This project is a proof of concept and an exploration of a new idea. The outcome is not perfect, and does not work for all people or all devices, but it does demonstrate the concept and idea we set out to explore. The current version is limited to a limited set of iOS devices (due to the lack of a developer account) and only works best when the training data input into the model was done by the same person who is using the app. Future modifications could change that, but we were limited by external factors, such as time and prior AI experience. A future team that has a better range of skills would be able to generalize our model, and make it work more reliably for more people. See the 'Future Changes' section below for a more detailed description.

2. Project Introduction

The project was requested by our client, Scott Fairbanks, who is one of the project instructors in the Computer Science department at Oregon State University. The purpose of building the gesture recognition keyboard was to explore a new method of keyboard input, and introduce a new way of controlling our devices. Our project chose to focus on iOS devices, and using gestures to type out words. Our team is made up of 4 members, Jeremiah Kramer, Zachary Horine, Lauren Sunamoto, and Changkuan Li. Jeremiah was in charge of host app design, Changkuan was in charge of keyboard design and interface, Zachary was in charge of model training and Lauren was in charge of data creation and management. Scott was mostly supervising, he gave feedback and advice based on how exactly he wanted the product to turn out. The changes in the spring term did not affect our deliverable too much although it did make it difficult to make final changes and perform testing as we were not able to meet in person to test and evaluate our app on physical devices. The basic configuration and design of our app was mostly done at the end of the winter term, and we had to focus on refining data, improving accuracy and making sure everything functioned as expected for the final delivery. Our app is a proof of concept for this kind of language input, there are some things that could be improved. The most notable of this is the accuracy and speed at which the user is able to type. We have detailed the changes required to accomplish this below, in the 'Future Changes' section.

3. Requirements Document

A. Introduction

1) Purpose: The purpose of this document is to clearly outline what our group is doing for our project and determine what our client will expect when the project reaches its deadline. More importantly, this document acts as a legal contract for us, our client, instructors, and teaching assistants.

2) Scope: This document will clearly define what our group will accomplish for our project. Also, this document will detail specific expectations for our client, instructors, and teaching assistants so that everyone has the same expectations. However, this document will not provide a detailed solution of the problem at hand.

3) Definitions, Acronyms, Abbreviations:

Term	Definition
Algorithm	a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
CreateML	framework used to create and train our machine learning model; automatically computes precision and recall metrics on selected datasets
Motion Gesture	the use of motions of the limbs or body as a means of expression.
SensorLog	an application used to read out accelerometer data from an iPhone and save it as CSV files
XCode	an integrated development environment for macOS containing a suite of software development tools developed by Apple for developing software iOS, macOS, etc.

4) Overview: This paper is structured such that the project's description comes first. Next, requirements for the project are discussed in detail. This includes specific deliverables with corresponding metrics that determine whether or not the project has been completed with satisfaction. Then, the project milestone outline is described with a Gantt chart. Finally, we wrap up the document with a summary of what was discussed.

B. Overall Description

The project goal is to create a gesture recognition keyboard for an iPhone. An application like this is designed to improve accessibility in text input, and will allow users with limited dexterity or range of motion to input text in a less standard way. By allowing users to move their phone in a series of distinct motion gestures, we will be able to determine what letter they are attempting to type. This service will be available from any text input field the user attempts to use it in, and will use a CoreML machine learning model to provide real-time gesture recognition and processing.

C. Specific Requirements

1) Datasets: We will use three datasets: training, validation, and testing. The training dataset will include 520 CSV files. The validation dataset will be automatically generated by CreateML. The testing dataset will be a subset of the training dataset. Our datasets will consist of 3-acceleration sequences of gesture classes for a complete alphabet set (i.e. 26 characters). Our data will be collected with the application, SensorLog, at 30Hz and we will use around 1-2 seconds of data samples. Data will be separated into gesture-specific folders, each folder containing multiple CSV files.

2) Gesture/Character Mappings: Our dataset consists of 26 unique motion-based gestures/characters which are defined by the mappings shown in the figure below (Figure 1). Each gesture is fairly easy to quickly complete and replicate, and can be accurately differentiated by our machine learning model.

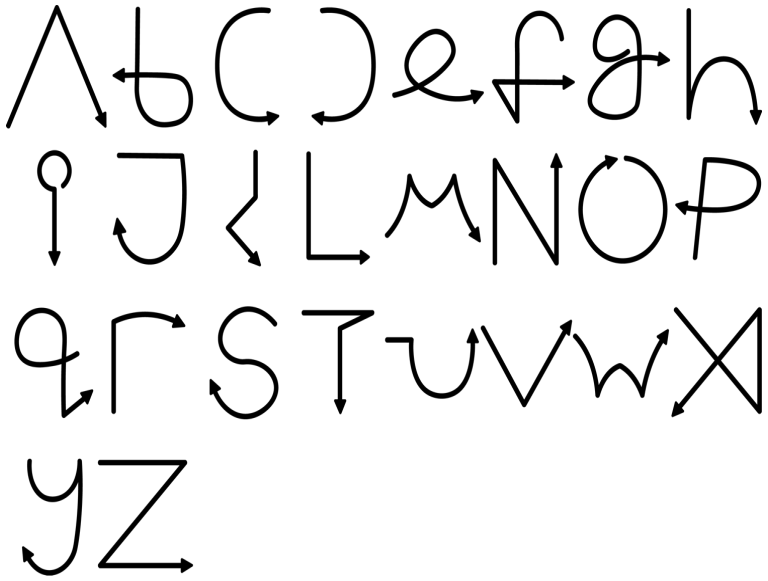


Fig. 1. Gesture Character Mappings for Gesture Recognition

3) Machine Learning Model: We will use Apple's CreateML tool to create and train a CoreML machine learning model. This will allow easy integration into our application through Xcode. CreateML automatically computes precision and recall metrics on 3 datasets: training, validation, and testing. Precision metrics indicate how effective our model is at labelling only when appropriate for a given gesture (few false positives). Recall metrics indicate how effective our model is at finding all of the relevant examples of a gesture (few false negatives). For each dataset we will achieve at least 85 percent for precision and recall.

D. Gantt Chart

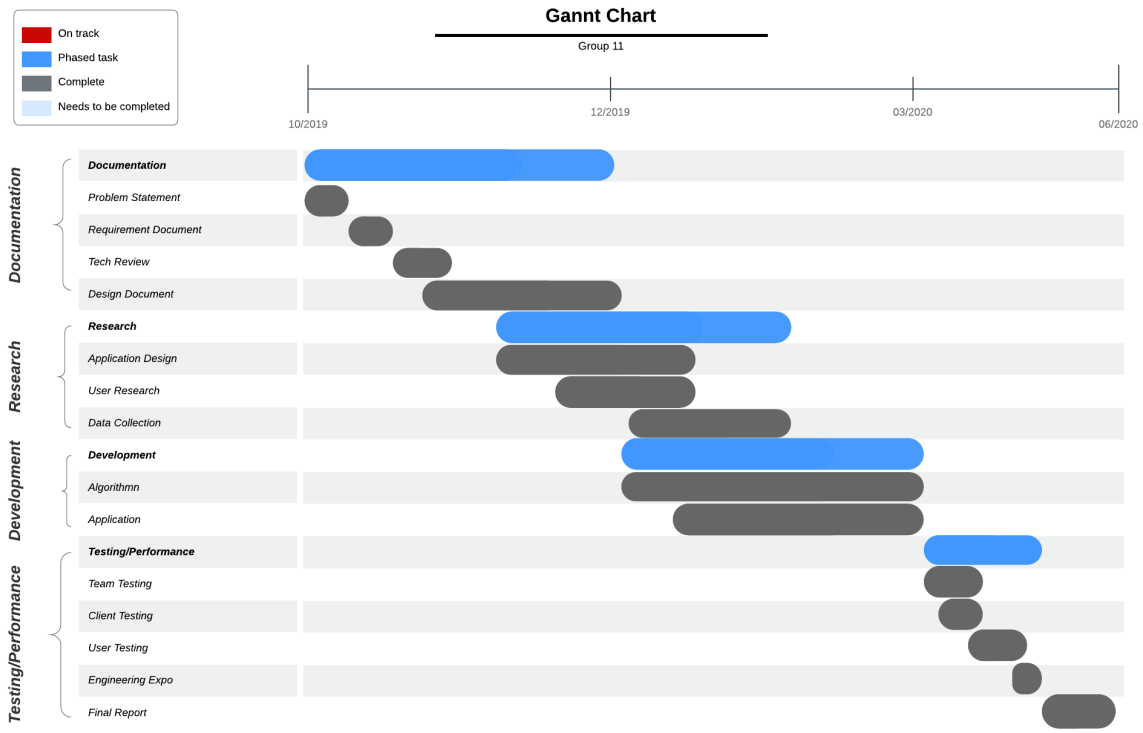


Fig. 2. Gantt Chart

4. Design Document

A. Introduction

1) Purpose: This document will outline the development design and considerations of our project. Each of the following sections discuss the specific technologies and methods that we will use.

2) Scope: This document will cover the overall design of our project, what systems and technologies we will use, as well as some of the research topics and theories that will help us as we build our project. This document is a plan and a snapshot of the project at its beginnings. The final project may use different tools, and may not employ the use of the same theories. This document has the ability to change.

3) Overview: This document outlines the various pieces of our project. For each section, design decisions are made to identify the most viable option for the project, in terms of overall quality, stability, and ease of development. The topics covered include: platform, language, data storage, categorization and prediction engine, sensors, human computer interaction and character mapping. This document also covers research topics that will help with development, including: auto-complete, gesture studies and user memorization, ground truth finding, letter statistics, grammar statistics, and computer accessibility.

B. Mobile Platforms

For our mobile application, there were two options we were considering for its Platform: iOS and Android. These are currently the two most prominent mobile platforms that exist. By examining each platform and considering key difference between them we determined that iOS was better suited to our requirements.

1) iOS: iOS is a closed platform, with open source components and so, customization is limited. And yet, it is known to be very secure with the prioritization of privacy, security, and reduced risk of malware [1]. The platform's reputation for better security contributes to iOS having more penetration in the enterprise market [2]. All applications are purchased from the Apple App Store and software updates are available for older iOS devices. iOS has a higher revenue per user. iOS applications take less time to develop as code is written using Swift, Apple's official programming language. However, the process of publishing an application is time consuming because of strict controls put in place by Apple. Also, development and testing must be conducted on an iPhone. Lastly, this platform was requested by our client to be used [1].

2) Selection: We will use the iOS platform mainly because it has been requested by our client to be used. Later, we may decide to launch the application on Android once it is established.

C. Programming Language

The first piece of this project that I will address is the language of the software. The language that our team uses depends on the platform that we choose. Since we chose iOS as the platform of choice, I will discuss iOS-compatible programming languages. .

1) iOS Research: Our options are: Swift, Objective-C, Javascript with React Native, Dart with Flutter, or C# with Xamarin [3]. Each of these languages has pros and cons. Swift is the most popular language used in most modern applications on the app store today. Further, it was developed and is maintained by Apple. Swift is also extremely robust and is always improving. Many libraries are available with Swift because it is an open source project open to anyone. Objective-C is the first language released by Apple back in 1984. This language is similar to C/C++ and many older apps are written in this language. Objective-C is well tested and is a stable language. A drawback of Objective-C is that it is losing popularity and

there aren't as many Objective-C proficient developers. Javascript with React Native is a language and library combination. Javascript is a language mainly used for full stack development and web development. React is a library heavily influenced by Facebook. There are some drawbacks to this language. First, memory management is handled in a different way and there isn't a browser that will automatically handle memory efficiently. Second, performance is lacking compared to other languages. The main goal of React was to utilize a unified code base for both iOS and Android apps. However, the language has shown to cause more problems than it solves. React is in its early stages and will need to continue to get better. Dart with Flutter is a language and library combination created by Google. Dart is a language that is very similar to Java and C++ (object oriented languages). Flutter is a library that makes is very easy to create appealing-looking apps in a short amount of time. Google claims that this language is productive, approachable, and fast. This language is a newer language that is growing rapidly. Finally, Xamarin was created eight years ago. It is open source and is able to utilize "hardware accelerations" that yields efficient performance. The drawback is that this language has decreased in popularity.

2) Recommendations: Following programming language research for iOS, there are some good options to choose. I would suggest that we use Swift as our programming language. The reason for this is because Swift is a very popular language among iOS applications. This means that there is a lot of support from Apple and online communities.

D. Data Storage Method

In order to persist gesture data inside our app, we will need to store a set of gesture point arrays that encode the base data for each gesture, and allow the system to match the gesture that the user has input, and reference it to a character. There are many ways to do this, but the most commonly used are (a) an SQLite database, (b) external files or (c) as a set of key pairs stored inside the app configuration files. Both Android and iOS support all three of these methods, but they all have their specific use.

1) External Files: External files are extremely useful when it comes to application design. sometimes, when the data that you have doesn't fit into a specific data model, it can be easier to store it in a format of your choosing and read it in yourself. There are some drawbacks to this method though. Depending on where you store your data, the data can be accessed by the user and be moved or deleted. Although on all systems, there are private data stores that can be used to keep your data safe. These files often sit close to the application code so that they share the same permissions. One of the other drawbacks is that it is more difficult to keep track of your data, and where it is stored. This could be remedied by using one of the other data storage methods though, as it would provide a 'link' to the location of the data.

Overall, the concept of storing our gesture models in their own files makes the process of reading just one gesture in more efficient, and allows us to more easily add and remove data throughout the development process.

2) Key Pairs: Key pairs are an easy way to store small amounts of data, and are useful for things like usernames or device keys. They are very specific on the types of data that they can store, and generally allow for personalized data to be stored without resorting to creating a whole database or file system.

On iOS, these keys are stored in a '.plist' file within the app package. Because of the way the file is stored, there is a maximum file size of 4GB (file-system restriction). [4] This, along with the fact that the file has to be read in every time the app loads, or the first time data is requested, means that the amount of data needs to be limited, and shouldn't be used to store large data structures. The only difference with Android devices is that the 'Shared Preferences' objects are stored in an xml file within the system. [5] Otherwise they act the same as an iOS device.

3) Selection: For our application, the best way to store our data actually comes as a combination of two methods. We will be using both external files, as well as key pairs stored inside the application executable. The advantage to this is that we can store our gesture models in their own, individual files, which allows us to make them modular, and enables the easy creation of user gesture files. We will then use key pairs to store the association between the character and their gesture file. This also allows us to avoid some of the downfalls of external files.

E. Categorization engine - how to find a ground truth

1) What is a Ground Truth?: In general, a ground truth is a data set that defines the base observations that all observations made after its establishment are based on. The concept of a ground truth is used extensively in image recognition, and is heavily related to a scientific assumption. In this way, it can be used as a basis of observation, and allows the observer to determine whether or not the data that they have gathered fits the model or not. [6]

2) What is included in a ground truth?: At its base, the ground truth is the simplest, most basic, and neutral form of the data expected. However, one set of perfect data is not enough to define the base case of the model. A ground truth has to not only include the optimal solution, but it also has 'sub-optimal' solutions that are still in the threshold. [7] It also includes solutions that are not in the data-set, which help to filter out data that doesn't fall within the confines of the desired outcome.

In image recognition, the ground truth also has multiple layers. All of the layers help to filter the data, and make it easier to pinpoint what is different from the ground truth. By looking at things such as the diffusion, reflectivity, shading and the specular components of an image, it is easier to see what part of the image differs from the ground truth, which makes it easier to correct later. [8]

3) How can we apply this?: Detecting gestures is difficult for many reasons, but they all lead to one thing: no two movements are ever the same. A lot of this has to do with sensor accuracy, as the accelerometer in a phone is 'precise' but not very accurate. The little irregularities in motion sensing stack up, and very quickly make pulling the true motion out of the noise very difficult. The other complication is that humans never recreate the same exact gesture, and no two humans will perform the same gesture the same way. because of this, the motion itself is not accurate to the model. This is why a simple 'perfect' gesture can't be used as the ground truth. By using a ground truth in multiple axis, as well as tolerances and negatives, we can eliminate bad data, and generate the most probable path.

F. Prediction Engine

The goal of the prediction engine is to intelligently match specific user input to its corresponding character mapping. In order to implement this engine, we need to select and develop an algorithm.

1) Algorithm Background Information: This algorithm will be a machine learning algorithm. This means that, the algorithm needs to predict a possible output based on user input. Of the types of machine learning algorithms, this algorithm that our team needs is a "supervised learning" algorithm as we need it to find a target or outcome variable predicted from independent variables generated by user input [9]. The independent variables will be inputted into this algorithm that generates desired outputs. This algorithm model will need to be trained to generated the desired outcome until we reach a high level of accuracy.

2) Motion Gesture Algorithms: Since there are multiple different supervised learning algorithms that we can use, we need to find possible algorithms that relate to motion gestures the closest. In an academic journal named "Motion-Based Gesture Recognition Algorithms for Robot Manipulation," there are three algorithms that are discussed. These algorithms include: Dynamic Time Warping, Hidden Markov Model, and Distance Metric Learning [10]. The journal assesses the accuracy of these accelerometer-based algorithms. Dynamic time warping and hidden Markov model algorithms are two classical pattern recognition methods.

a) Dynamic Time Warping: The dynamic time warping algorithm is a geometric approach to predicting an output. Fundamentally, the algorithm uses two different time series and will "warp" one time series to make it resemble the other series. When warping, the goal is to make the accumulative distance between them minimal. The more the algorithm succeeds in its goal, the more accurate the algorithm is at predicting the desired outcome. With this algorithm, dynamic programming is used. The output is a matrix that indicates how much the two different time series differ when they are perfectly aligned. This algorithm fits perfectly with motion gesture application because it can use a set of known pattern templates and recognize an unknown pattern based on the known templates. It chooses the template that best fits one of the known patterns. Our team would leverage this by mapping specific gestures to a corresponding character.

b) Hidden Markov Model: The hidden Markov model algorithm is a statistical approach to prediction. The hidden Markov model is an extension of the observable Markov model. An observable Markov model is a discrete, first order Markov chain that has: a set of states, a set of state transition probabilities, and a specification of an initial state. "A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules" [11]. The output of the model's process is a set of states at an instant of time and each state corresponds to a certain physical event that is observed [10]. The process becomes "hidden" when the observation is a probability of the state. Therefore, the state isn't directly observed. In a nutshell, based on specific probabilities of states at given times, a physical event can be predicted. This fits well with motion gesture recognition. Since the hidden Markov model can have different typologies and models, the model that fits the best with motion gesture recognition is a model that describes observations in a chronological sequence. This model is named the left-to-right model. With this algorithm, our team would save specific patterns that correspond to a character mapping by setting precise probabilities of space and time.

c) Distance Metric Learning: The distance metric learning algorithm is a newer technique that recognizes patterns based on a group of known patterns. The unknown pattern is then assigned to the closest known pattern. The difference in the distance metric learning algorithm is the underlying closeness metric distance from Euclidean distance to the Mahalanobis distance metric. This new metric provides more accurate results. The drawback to this algorithm is that it is very complex compared to the other algorithms.

3) Review and Recommendation: The academic journal that suggested these three algorithms performed tests on each of them separately and reviewed their accuracy. The distance metric learning performed with 98.26% overall average recognition rate. The hidden Markov model achieved 94.44% overall average recognition rate and the dynamic time warping algorithm achieved 89.89% overall average recognition rate. In order to decide the algorithm to choose and implement, my team needs to weigh algorithm performance versus its complexity. This means that our team may choose an algorithm that doesn't yield the highest accuracy if another algorithm is easier to implement. The reason for this is because of overall time constraints for our project. With that being said, our team should wisely start with the distance metric learning algorithm or hidden Markov model.

G. Sensors and interpretation of output data

In order for the gesture-based keyboard to work certain sensor must be utilized and inputs must be recognized reliably.

1) Option: This section outlines two sensors as well as two methods to record data from them. The first sensor is the gyroscope, the second sensor is the accelerometer. The two methods of recording data from these sensors are calibrated and non-calibrated.

2) Goals for use in design: The main goals for the use of these sensors and data recording methods is accessibility and accuracy. These methods should be able to be used on every smartphone as well as be accurate to allow for complex gestures. The speed at which the user can type should be based off their skill rather than hardware or software limitations.

3) Criteria being evaluated : The sensor must be abundant in modern smartphones and the data recording method should allow for accurate and smooth reading.

4) Comparison Breakdown:

- Accuracy, the whole project is based around gestures that need to be reliably recognized by the smartphone.
- Accessibility, the method of data recording and processing should not depend on a piece of hardware that is difficult to come by.
- Speed, hardware and software limitations should not hinder the user experience.

1) Gyroscope
   - Gyroscopes accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
   - Gyroscopes are present in near all modern smartphones.
   - Gyroscopes output raw data instantly.
2) Accelerometer
   - Accelerometers accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
   - Accelerometers are present in all modern smartphones.
   - Accelerometers output raw data instantly.
3) Uncalibrated
   - Uncalibrated data is raw and is precise but not accurate.
   - Uncalibrated data is outputted the same on all smartphone gyroscopes [12].
   - Uncalibrated data is recorded instantly.
4) Calibrated
   - Calibrated data accuracy is dependent on the fidelity of the calibration.
   - Calibrated data is dependent on factory calibration methods [12].
   - Calibrated data is recorded instantly.

5) Selection: The best option is actually to use the Gyroscope and Accelerometer in tandem and to record data uncalibrated data and manipulate it. Calibrated data recording relies heavily on the accuracy of the factory calibration methods and results in most jumpy data. This is due to the fact that it tries to account for the faults in the sensor and guess at the correct output [12]. Uncalibrated data recording allows for less sensor noise [13] which means that the data is less likely to have outliers

or jumps. This allows for a smoother reading if averages of the output data are taken over short intervals. By using both the gyroscope and accelerometer is it possible to differentiate between the device being upside down performing a gesture vs right side up doing the inverse gesture. Additionally, creating artificial zeros at the start of every recording session allows for an artificial calibration which in turn results in the best recording method for this particular application.

H. Human Computer Interaction

The Human Computer Interaction is one of the most important aspects of the project due to its nature. The user is using a gesture-based keyboard and needs to understand what they are inputting at all times.

1) Option: This section outlines three methods of Human Computer Interaction that could be used in the project. The first is haptic feedback, the second is sound queues, the third is on screen prompts.

2) Goals for use in design: The goal of Human Computer Interaction is to provide the user with quick and easily discernible feedback on what is being inputted. In this application the user is manipulating their phone in order to input characters so it is important that the user has a way of receiving feedback regardless of the position of the phone.

3) Criteria being evaluated : The most important criteria that these technologies are being critiqued by is their speed and accessibility. This method of keyboard input does not allow for the user to be looking at the smartphone screen at all times.

4) Comparison Breakdown:

- Speed: One of the requirements of the project is for a user to be able to type at an acceptable speed.
- Accessibility: The technology should have the least amount of hoops for the user to jump through in order to be used.
- Abundancy: The technology must be present in most all smartphones to allow for a broader user base.

1) Haptic Feedback
   - Haptic Feedback is instant upon user input.
   - Haptic Feedback works without the user having to access and settings or plug anything in.
   - Haptic Feedback is present in almost all smartphones as it uses the vibration motor.
2) On Screen Prompts
   - On Screen Prompts, are instant but users need time to read them.
   - On Screen Prompts, are dependent on the size of the screen.
   - On Screen Prompts, are the most accessible and are present on every smartphone.
3) Sound Queues
   - Sound Queues are instant upon user input.
   - Sound Queues require the user to be in an area that allows it and or the use of headphones.
   - Sound Queues are present on every smartphone.

5) Selection: The best option for this project is Haptic Feedback. Haptic Feedback is perfect for this application as the users hands must be on the device to begin with which guarantees that the user will receive the vibration. This also allows for the phone to be in an orientation in which the user cannot see the screen which opens doors to more gestures. Haptic Feedback solely relies on the vibration motor of a smartphone and can be fired instantly to correspond with a specific gesture [14].

I. Human Computer Interaction: Computer Accessibility

Computer accessibility in human-computer interaction refers to the accessibility of a computer system to people despite disability type or severity of impairment. In developing our application we hope to improve the accessibility of text input by creating features that prioritize individuals with specific visual, hearing, and motor or dexterity impairments.

1) Visual Impairments: Visual impairments include blindness, low vision, and color blindness [15]. In developing our application we can build features that rely on other senses, such as hearing and touch. We can use haptic technology which involves sending a user feedback by creating vibrations in a mobile phone. Such feedback is important in recall and recognition of users as they learn to use the application. For example, a vibration can alert a user to the generation of a letter and aid the speed in which they ”type”.

2) Hearing Disabilities: Hearing disabilities range in severity from total deafness to slight loss of hearing [15]. As mentioned above, sound queues would not be very effective. Therefore, haptic technology would be a better alternative method to provide user feedback.

3) Motor or dexterity impairments: Motor or dexterity impairments encompass a large variety impairments including total absence of limbs or digits, paralysis, lack of fine control, instability or pain in the use of fingers, hands, wrists, or arms [15]. Some of these individuals would especially benefit from alternatives to standard input methods (i.e. keyboard). Although, not all individuals that fall under such category will be able to use our application because it requires some fine motor skills for precise generation of characters as well as gross motor skills with wrist movement. But, there are specific elements of our application that we can do differently from traditional keyboards.

a) Touchscreen Manipulation: Traditional mobile keyboard require steady and precise tapping on small key spaces which can be difficult for certain users, such as people with Arthritis or those with uncontrollable hand tremors from Parkinson’s disease or Multiple sclerosis [16]. Therefore, our application should not require complex use of the screen. For example, in addition to defining gestures for alphabet letters we should define a gestures for deleting and shifting. It should also be error-tolerant (e.g. deletion should not necessarily be easy to do).

b) Customization: To accommodate the variation in impairments we should allow for some customization of our application. For example, the typing speed should be easily adjustable. The transition between characters/gestures can be adjusted as well as the recognition and translation of characters/gestures.

4) Conclusion: In developing our application we will consider the importance of computer accessibility by creating features that meet the needs of certain individuals with specific disabilities. The functionality of our application will rely mostly on gesturing and user feedback will include haptic feedback which usable for a wide range of users.

J. Gesture Studies and User Memorization

Gesture Studies and user Memorization outlines the user experience in terms of the speed of learning. This differs from the Human Computer Interaction Section as it is a system that is not necessary once the keyboard has been learned rather than one core to the project.

1) Option: This section outlines the two different methods of feedback that will help users memorize the gesture-based keyboard and increase their typing speed.The first is sound queues and the second is on screen visuals.

2) Goals for use in design: The goals for the gestures for the project is for them to be easily differentiated and simple to execute. This means that there has to be clear feedback to the user as to what they are inputting and the options that they have going forward.

3) Criteria being evaluated : The most important aspect of Gesture studies and user memorization is the speed of learning and simplicity. This means that the speed, learning speed, and accessibility are all criteria that will be used to evaluate it.

4) Comparison Breakdown:

- Speed, the feedback to aid memorization must feel that it is tied with user inputs so it must be instant.
- Learning Speed, the system must be easy to understand and memorable.
- Accessibility, the system must be within the capabilities of the modern smartphone.

1) Sound Queues
   - Sound Queues are instant.
   - Sound Queues are among the most easily recognized feedback methods and are highly recognizable [17].
   - Sound Queues can be played by all modern smartphones.

2) Visual Aid
   - Visual Aids are instant but take time for the user to process.
   - Visual Aids can be easily learned.
   - Visual Aids can be displayed on all modern smartphones.

5) Selection: The best option for this project is sound queues. Though the use of haptic feedback is ideal for the core elements of the project, there is still the aspect of auto-complete and predictions that have to be conveyed to the user in an efficient manner. This can be best achieved by using sound clues. RCP Tones offers free use of their sound effects for app developers but it is extremely important that the selection of the sounds is strenuous [18]. There is a whole science behind how the brain memorizes things but the field of sound is the most straightforward. The brain memorizes melodic patterns the easiest, after just 10 playbacks of a patterned sound the average person can pick out the sound for other similar ones [17]. By choosing distinct, short, pattern-like sounds users can easily understand what the program is trying to convey to them. The applications of this include letting the user know that there is an auto-complete ready for them, as well as telling the user what group of letters they are selecting. This paired with an auditory output upon the selection of a letter should create a flow in the user experience which allows for fast learning.

K. Auto-complete

1) Option: This section outlines two options for auto-complete. The first being word based, the second be letter based.

2) Goals for use in design: The goal for auto-complete is to increase user input speed without hindering user experience in anyway through perfect implementation.

3) Criteria being evaluated : The most important aspects of the auto-complete method is speed and the lack of unwanted prompts. The user must feel like they have the option to accept the auto-complete without it getting in the way of typing the next letter.

4) Comparison Breakdown:

- Speed, the auto-complete must output a suggestion before the user can input the next letter.
- Accuracy, the auto-complete must be able to adapt to the user and not repeatedly suggest a word that the user does not use often.
- Learning Speed, the user must be able to predict what the auto-complete will suggest through typing experience.

1) Word Based Auto-complete

- Word Based Auto-complete is marginally slower than letter based.
- Word Based Auto-complete is accurate for common words.
- Word Based Auto-complete is fairly unpredictable unless a longer word is being typed.

2) Letter Based Auto-complete

- Letter Based Auto-complete is near instant.
- Letter Based Auto-complete is based off the probability of what the next letter will be [19].
- Letter Based Auto-complete is unchanging and predictable with practice.

5) Selection: The best option for this project is Letter Based Auto-complete due to its simplicity and ease of implementation. The gesture-based keyboard is outlined purely in letters and implementing a way to enter a word based off auto-complete could raise problems. Additionally, being as letter based auto-complete is derived from the likelihood of the next letter, it allows for the elimination of others if their probability becomes zero. By using deterministic acyclic finite state automaton (DAFSA) [19], a string of letters leads to the next which allows for the user to be guided to their next letter with more accuracy then the patterns in human speech that word based auto-complete uses. For example, DAFA outlines that if a Q is inputted by the user then the next most likely character would be U [19]. This allows for the user to also learn the auto-complete method and utilize it to master the input method.

L. Character Mapping

Character mapping is extremely important because it affects user experience and algorithm success. Since the goal of our application is to create alternative user input via motion gestures, our team needs to map motions to letters and words. Given this, we have some different options for implementing character mapping.

1) Straight Line Simple: One approach is to map 30 unique gestures to 30 different characters. The reason we chose 30 is to allow for 26 different letters in the English alphabet and four additional gestures for special characters such as periods and commas. In order to make it easier for the user to learn the gestures, the gestures would be simple straight line movements of their device. However, mapping 30 straight line gestures would be an issue because 360 degrees divided by 30 is only 12 degrees of space for the user to attempt to generate text. This would be infuriating for the user because the degree for error is much too small. To take this a step further, our application could leverage the gyroscope embedded in mobile devices and initially flip the device on its side, facing upright, or facing downward. Then only 10 movements in space would be needed. That means that users would have 36 degrees (360 degrees divided by 10) of space on the devices axis to gesture. This is one possibility. However, if users wanted to add shortcuts as new gestures, this would limit them.

2) Series of Combinations: Another approach that our team can take is to map a series of gestures to certain characters and words or phrases. In other words, if the device is moved forward, right, right in succession, that would result in a corresponding letter. With this idea, the user would only have to move their device in six different directions: forward, right, left, backward, up, and down. This would allow our algorithm to be more successful. However, the drawback is recognizing when the next ”gesture” occurs (either within the combination series, or between characters). In order words, from the previous example, the algorithm would need to know when the user stopped gesturing forward, and started gesturing to the right. Also, the algorithm needs to know when to select a character and allow for the user to gesture the next character. Further, the amount of combinations could become complex, especially if the user adds many shortcuts. This would make it difficult for the user.

3) Object Pattern Complex: Another idea is to gesture specific objects and map them to their respective character. For example, gesturing a ”circle” would result in the letter ”o.” This idea came from the academic journal that was referenced prior [10]. Although this is a valid approach, I think this would be very complex for the user. Even though this might result in a higher accuracy for the algorithm, text generation speed would suffer greatly.

4) Review and Recommendation: After review, mapping gestures to characters, words, or phrases, is difficult for users and the algorithm. As a team, we know that this will be a challenge and users and the algorithm will need to continuously learn from one another. With that being said, users should be able to map custom gestures to the application. This would result in a more user friendly application and increased usability. Our application should have default character-gesture mapping and allow for custom gestures. Also, the user should be able to edit the default mapping. The default mapping should be only moving the phone in simple straight lines with the phone's initial position altering.

M. Letter Statistics

This application must be able to generate text from motion gestures with a mobile device. We will define a set of gestures that is easy to complete and replicate, and allows for a 'typing speed' of at least ten words per minute. We will include gestures for the thirty-size standard characters, as well as a include support for various modes of punctuation, white-space and capitalization. Letter statistics can be useful in determining how we define the different gestures. Furthermore, like the optimization of keyboard layout letter statistics can be used to improve typing speed and decrease user errors.

1) Letter Frequency: To ensure our goal 'typing speed' we should prioritize the most commonly used alphabetic characters, and pair the least used characters with gestures that are more complex due to the need for distinct motion-based gestures. Based on a sample of 40,000 words, the letter 'Z' which occurred 128 times and is the least frequently used English letter should be defined in a way such that it is not easier to replicate than the most frequently used letter, 'E', which occurred 21,912 times. Also, using these letter statistics we can consider how the letters relate to one another. It is easier to identify vowels because most letters appear before and/or after them [20]. Furthermore, our application should be error-tolerant by defining the backspace character so that it cannot be misinterpreted as a common character gesture.

2) Digraph Frequency: In addition to letter frequency, we can look at the statistics for digraph frequency to improve user 'typing speed' as well as accuracy and ease of use. Digraphs are pairs of letters. For example, based on a sample of 40,000 words, the most common digraph is ”th” which occurred 5,532 times. This is consistent with the fact that the word, ”the” is the most commonly used word in the English language. This information is useful in defining motion-based gestures because it is important to consider the transition between letters, especially those that are frequently next to each other. We must consider that there are specific gestures that will be difficult to create and discern following another gesture [21].

3) Conclusion: By using letter and digraph frequency statistics in the defining of our motion-based gestures, we will be able to maximise the efficiency of typing on our keyboard. The most common alphabetic characters and digraphs will be prioritized over the least common. Overall, such considerations will help in improving user experience by prioritizing ease of use and design.

N. Grammar statistics (frequencies and common patterns)

1) Keyboards: One of the important things when designing an input device is the layout of the input 'keys'. Many traditional keyboard layouts were designed around the English language, and the letter frequencies that are used to type some of the

most common words. One of the most common examples of this is the Devorak keyboard, which prioritizes vowels and commonly used consonants. [22] we will have to do a similar thing with our keyboard, but we will also have other issues to attend to. Because of the nature of gestures, there are certain gestures that will be difficult to detect if they come after another gesture, and some that are easy and flow into one another. In this way, we are dealing with some of the issues that came along with typewriters (frequent jamming), and the supposed reason that the qwerty layout was invented. In reality, the layout was invented to slow down faster typists, which is exactly the opposite effect that we want to have. [23]

2) Frequency in Words: Instead, we should pair more readily accessible gestures with the most commonly used characters, and associate the least used characters with 'more out of the way' gestures. We also need to look at the characters that commonly come after one another, and make sure the par of gestures work well together. A study done by the College of Math at the University of Illinois shows how likely a given character is to come after another. In their study, there were a large number of characters that never appeared together, and a list of the most frequent ones, which included 'th', 'he', and 'an' as the top three. [24] This makes a lot of sense, as the three most common words in the English language are 'the', 'of' and 'and'. [25]

3) Conclusions: If we can use this information to map our gestures to letters, then we can improve the efficiency of typing on our keyboard. By taking into account all of the lessons learned in keyboard design, and incorporating them into our app, we can create a better experience for our users.

O. Conclusion

This document covered the overall design and considerations of our project including the selected technologies and methods that are best suited to our project's specific requirements.

5. Tech Review - Changkuan Li

A. Human Computer Interaction

The Human Computer Interaction is one of the most important aspects of the project due to its nature. The user is using a gesture-based keyboard and needs to understand what they are inputting at all times.

1) Option: This section outlines three methods of Human Computer Interaction that could be used in the project. The first is haptic feedback, the second is sound queues, the third is on screen prompts.

2) Goals for use in design: The goal of Human Computer Interaction is to provide the user with quick and easily discernible feedback on what is being inputted. In this application the user is manipulating their phone in order to input characters so it is important that the user has a way of receiving feedback regardless of the position of the phone.

3) Criteria being evaluated : The most important criteria that these technologies are being critiqued by is their speed and accessibility. This method of keyboard input does not allow for the user to be looking at the smartphone screen at all times.

4) Comparison Breakdown:

- Speed: One of the requirements of the project is for a user to be able to type at an acceptable speed.
- Accessibility: The technology should have the least amount of hoops for the user to jump through in order to be used.
- Abundancy: The technology must be present in most all smartphones to allow for a broader user base.

1) Haptic Feedback
   - Haptic Feedback is instant upon user input.
   - Haptic Feedback works without the user having to access and settings or plug anything in.
   - Haptic Feedback is present in almost all smartphones as it uses the vibration motor.
2) On Screen Prompts
   - On Screen Prompts, are instant but users need time to read them.
   - On Screen Prompts, are dependent on the size of the screen.
   - On Screen Prompts, are the most accessible and are present on every smartphone.
3) Sound Queues
   - Sound Queues are instant upon user input.
   - Sound Queues require the user to be in an area that allows it and or the use of headphones.
   - Sound Queues are present on every smartphone.

5) Selection: The best option for this project is Haptic Feedback. Haptic Feedback is perfect for this application as the users hands must be on the device to begin with which guarantees that the user will receive the vibration. This also allows for the phone to be in an orientation in which the user cannot see the screen which opens doors to more gestures. Haptic Feedback solely relies on the vibration motor of a smartphone and can be fired instantly to correspond with a specific gesture [14].

B. Sensors and interpretation of output data

In order for the gesture-based keyboard to work certain sensor must be utilized and inputs must be recognized reliably.

1) Option: This section outlines two sensors as well as two methods to record data from them. The first sensor is the gyroscope, the second sensor is the accelerometer. The two methods of recording data from these sensors are calibrated and non-calibrated.

2) Goals for use in design: The main goals for the use of these sensors and data recording methods is accessibility and accuracy. These methods should be able to be used on every smartphone as well as be accurate to allow for complex gestures. The speed at which the user can type should be based off their skill rather than hardware or software limitations.

3) Criteria being evaluated : The sensor must be abundant in modern smartphones and the data recording method should allow for accurate and smooth reading.

4) Comparison Breakdown:

- Accuracy, the whole project is based around gestures that need to be reliably recognized by the smartphone.
- Accessibility, the method of data recording and processing should not depend on a piece of hardware that is difficult to come by.
- Speed, hardware and software limitations should not hinder the user experience.

1) Gyroscope
   - Gyroscopes accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
   - Gyroscopes are present in near all modern smartphones.
   - Gyroscopes output raw data instantly.
2) Accelerometer
   - Accelerometers accuracy depends on what a certain smartphone manufacturer puts in the phone however, are in general reliable.
   - Accelerometers are present in all modern smartphones.
   - Accelerometers output raw data instantly.
3) Uncalibrated
   - Uncalibrated data is raw and is precise but not accurate.
   - Uncalibrated data is outputted the same on all smartphone gyroscopes [12].
   - Uncalibrated data is recorded instantly.
4) Calibrated
   - Calibrated data accuracy is dependent on the fidelity of the calibration.
   - Calibrated data is dependent on factory calibration methods [12].
   - Calibrated data is recorded instantly.

5) Selection: The best option is actually to use the Gyroscope and Accelerometer in tandem and to record data uncalibrated data and manipulate it. Calibrated data recording relies heavily on the accuracy of the factory calibration methods and results in most jumpy data. This is due to the fact that it tries to account for the faults in the sensor and guess at the correct output [12]. Uncalibrated data recording allows for less sensor noise [13] which means that the data is less likely to have outliers or jumps. This allows for a smoother reading if averages of the output data are taken over short intervals. By using both the gyroscope and accelerometer is it possible to differentiate between the device being upside down performing a gesture vs right side up doing the inverse gesture. Additionally, creating artificial zeros at the start of every recording session allows for an artificial calibration which in turn results in the best recording method for this particular application.

C. Autocomplete

1) Option: This section outlines two options for autocomplete. The first being word based, the second be letter based.

2) Goals for use in design: The goal for autocomplete is to increase user input speed without hindering user experience in anyway through perfect implementation.

3) Criteria being evaluated : The most important aspects of the autocomplete method is speed and the lack of unwanted prompts. The user must feel like they have the option to accept the autocomplete without it getting in the way of typing the next letter.

4) Comparison Breakdown:

- Speed, the autocomplete must output a suggestion before the user can input the next letter.
- Accuracy, the autocomplete must be able to adapt to the user and not repeatedly suggest a word that the user does not use often.
- Learning Speed, the user must be able to predict what the autocomplete will suggest through typing experience.

1) Word Based Autocomplete
   - Word Based Autocomplete is marginally slower than letter based.
   - Word Based Autocomplete is accurate for common words.
   - Word Based Autocomplete is fairly unpredictable unless a longer word is being typed.

2) Letter Based Autocomplete
   - Letter Based Autocomplete is near instant.
   - Letter Based Autocomplete is based off the probability of what the next letter will be [19].
   - Letter Based Autocomplete is unchanging and predictable with practice.

5) Selection: The best option for this project is Letter Based Autocomplete due to its simplicity and ease of implementation. The gesture-based keyboard is outlined purely in letters and implementing a way to enter a word based off autocomplete could raise problems. Additionally, being as letter based autocomplete is derived from the likelihood of the next letter, it allows for the elimination of others if their probability becomes zero. By using deterministic acyclic finite state automaton (DAFSA)[19], a string of letters leads to the next which allows for the user to be guided to their next letter with more accuracy then the patterns in human speech that word based autocomplete uses. For example, DAFA outlines that if a Q is inputted by the user then the next most likely character would be U [19]. This allows for the user to also learn the autocomplete method and utilize it to master the input method.

D. Gesture Studies and User Memorization

Gesture Studies and user Memorization outlines the user experience in terms of the speed of learning. This differs from the Human Computer Interaction Section as it is a system that is not necessary once the keyboard has been learned rather than one core to the project.

1) Option: This section outlines the two different methods of feedback that will help users memorize the gesture-based keyboard and increase their typing speed.The first is sound queues and the second is on screen visuals.

2) Goals for use in design: The goals for the gestures for the project is for them to be easily differentiable and simple to execute. This means that there has to be clear feedback to the user as to what they are inputting and the options that they have going forward.

3) Criteria being evaluated : The most important aspect of Gesture studies and user memorization is the speed of learning and simplicity. This means that the speed, learning speed, and accessibility are all criteria that will be used to evaluate it.

4) Comparison Breakdown:

- Speed, the feedback to aid memorization must feel that it is tied with user inputs so it must be instant.
- Learning Speed, the system must be easy to understand and memorable.
- Accessibility, the system must be within the capabilities of the modern smartphone.

1) Sound Queues
   - Sound Queues are instant.
   - Sound Queues are among the most easily recognized feedback methods and are highly recognizable [17].
   - Sound Queues can be played by all modern smartphones.
2) Visual Aid
   - Visual Aids are instant but take time for the user to process.
   - Visual Aids can be easily learned.
   - Visual Aids can be displayed on all modern smartphones.

5) Selection: The best option for this project is sound queues. Though the use of haptic feedback is ideal for the core elements of the project, there is still the aspect of autocomplete and predictions that have to be conveyed to the user in an efficient manner. This can be best achieved by using sound clues. RCP Tones offers free use of their sound effects for app developers but it is extremely important that the selection of the sounds is strenuous[18]. There is a whole science behind how the brain memorizes things but the field of sound is the most straightforward. The brain memorizes melodic patterns the easiest, after just 10 playbacks of a patterned sound the average person can pick out the sound for other similar ones[17]. By choosing distinct, short, pattern-like sounds users can easily understand what the program is trying to convey to them. The applications of this include letting the user know that there is an autocomplete ready for them, as well as telling the user what group of letters they are selecting. This paired with an auditory output upon the selection of a letter should create a flow in the user experience which allows for fast learning.

6. Tech Review - Jeremiah Kramer

A. Introduction

1) Purpose: The purpose of this document is to describe the research for the technologies or methods that might be used in my groups project and review them accordingly. This document will also identify my role in the group and a high level overview of what our group is trying to accomplish.

2) Scope: This document will be an examination of the technologies, methods, or options of three specific pieces for the project. These three pieces are only a fraction of the project and the rest of the pieces will be addressed by my other group members. Also, this document will only briefly explain what my team is trying to accomplish, since this has been discussed in more detail in a previous paper.

3) Overview: This paper is structured such that my role in my groups project and a high level view of what our group is trying to accomplish comes first. Next, the programming language being used for the software is discussed. Third, the prediction engine for how to match input data with the closest base gesture will be identified and reviewed. Then, the paper will discuss character mapping and shifting usability. Finally, I wrap up the document with a summary of what was discussed.

B. Individual Role and Project Overview

This section will identify and explain my role in the project and a high level overview of what my team is trying to accomplish.

1) Role: First, my role for the project will consist of mainly developing the engine for matching user input with a corresponding character mapping. In other words, I will help with developing and integrating the prediction algorithm. I will also be involved with app development. This includes UI design and communicating to the user how to use the app as effectively as possible.

2) Goal: The goal of this project is to provide an additional mode of input to users on their mobile devices. By doing this, we hope to improve the accessibility of text input, all while minimizing the effect on typing speed. We aim to achieve this by providing a set of easy-to-use motion gestures, and creating an intelligent engine that can decipher user inputs. Specifically, we will create an application that has a simplistic and intuitive design, provides a high degree of accuracy with minimal impact on typing speed, and enhances the users' overall text generation experience. The application will serve as a keyboard extension on the users phone so that the user can switch to our application to generate text by moving their mobile device in space.

C. Programming Language

The first piece of this project that I will address is the language of the software. The language that our team uses depends on the platform that we choose. The different platforms are discussed by another one of my group members in more detail, but we are deciding between iOS and Android, the two most commonly used platforms.

1) iOS Research: If we choose iOS as the platform for developing the app, then our options are: Swift, Objective-C, Javascript with React Native, Dart with Flutter, or C# with Xamarin [1]. Each of these languages has pros and cons. Swift is the most popular language used in most modern applications on the app store today. Further, it was developed and is maintained by Apple. Swift is also extremely robust and is always improving. Many libraries are available with Swift because

it is an open source project open to anyone. Objective-C is the first language released by Apple back in 1984. This language is similar to C/C++ and many older apps are written in this language. Objective-C is well tested and is a stable language. A drawback of Objective-C is that it is losing popularity and there aren't as many Objective-C proficient developers. Javascript with React Native is a language and library combination. Javascript is a language mainly used for full stack development and web development. React is a library heavily influenced by Facebook. There are some drawbacks to this language. First, memory management is handled in a different way and there isn't a browser that will automatically handle memory efficiently. Second, performance is lacking compared to other languages. The main goal of React was to utilize a unified code base for both iOS and Android apps. However, the language has shown to cause more problems than it solves. React is in its early stages and will need to continue to get better. Dart with Flutter is a language and library combination created by Google. Dart is a language that is very similar to Java and C++ (object oriented languages). Flutter is a library that makes is very easy to create appealing-looking apps in a short amount of time. Google claims that this language is productive, approachable, and fast. This language is a newer language that is growing rapidly. Finally, Xamarin was created eight years ago. It is open source and is able to utilize "hardware accelerations" that yields efficient performance. The drawback is that this language has decreased in popularity.

2) Android Research: If we choose Android as the platform for developing the app, then our options are: Java, Kotlin, C++, C#, Python, or Corona [2]. First, Java is the most commonly used language for Android app development. It is the official language for Android application development. Java is also the most supported language by Google due the many of the apps in the Play Store being built with Java. Java is a fairly complex language but there is a lot of support from online communities. Kotlin is an alternative to Java. The reason for Kotlin is to make it a little simpler than Java to include more beginners to Android app development. C++ cannot create an app by itself, it requires "Android Native Development Kit(NDK)." Some drawbacks of C++ include increased complexity and thus more bugs. C# is very similar to Java as well. C# has many great features as Java such as garbage collection thus less chances for memory leaks. However, C# has an easier-to-read syntax than Java, so it is easier to understand. Like C++, Python requires a tools and packages in order to create applications. A common library that is used for developing apps is named Kivy. A drawback to this is that Kivy isn't natively supported. Finally, Corona is a development kit that is paired with a programming language named Lua. Lua is also simpler and easier to learn than Java. This language is mainly used for graphic-heavy games.

3) Recommendations: Following programming language research per platform, there are some good options to choose.

4) iOS: Given that our team chooses to develop in an iOS environment, I would suggest that we use Swift as our programming language. The reason for this is because Swift is a very popular language among iOS applications. This means that there is a lot of support from Apple and online communities.

5) Android: Given that our team chooses to develop in an Android environment, I would suggest that we choose Java or C++ as our programming language. The reason to choose Java is because of its immense support due to its popularity and similarity to C++. The reason to choose C++ is because we learned programming in C++ throughout our computer science collegiate careers. This means that all group members will have a C++ foundation, which will benefit our team.

D. Prediction Engine

The second piece of the project that I will address is the prediction engine. This prediction engine will consist of an algorithm that can intelligently match specific user input to its corresponding character mapping.

1) Algorithm Background Information: This algorithm will be a machine learning algorithm. This means that, the algorithm needs to predict a possible output based on user input. Of the types of machine learning algorithms, this algorithm that our team needs is a ”supervised learning” algorithm as we need it to find a target or outcome variable predicted from independent variables generated by user input [3]. The independent variables will be inputted into this algorithm that generates desired outputs. This algorithm model will need to be trained to generated the desired outcome until we reach a high level of accuracy. Some examples of supervised learning algorithms include: Regression, Decision Tree, Random Forest, KNN, or Logistic Regression.

2) Motion Gesture Algorithms: Since there are multiple different supervised learning algorithms that we can use, we need to find possible algorithms that relate to motion gestures the closest. In an academic journal named ”Motion-Based Gesture Recognition Algorithms for Robot Manipulation,” there are three algorithms that are discussed. These algorithms include: Dynamic Time Warping, Hidden Markov Model, and Distance Metric Learning [4]. The journal assesses the accuracy of these accelerometer-based algorithms. Dynamic time warping and hidden Markov model algorithms are two classical pattern recognition methods.

3) Dynamic Time Warping: The dynamic time warping algorithm is a geometric approach to predicting an output. Fundamentally, the algorithm uses two different time series and will ”warp” one time series to make it resemble the other series. When warping, the goal is to make the accumulative distance between them minimal. The more the algorithm succeeds in its goal, the more accurate the algorithm is at predicting the desired outcome. With this algorithm, dynamic programming is used. The output is a matrix that indicates how much the two different time series differ when they are perfectly aligned. This algorithm fits perfectly with motion gesture application because it can use a set of known pattern templates and recognize an unknown pattern based on the known templates. It chooses the template that best fits one of the known patterns. Our team would leverage this by mapping specific gestures to a corresponding character.

4) Hidden Markov Model: The hidden Markov model algorithm is a statistical approach to prediction. The hidden Markov model is an extension of the observable Markov model. An observable Markov model is a discrete, first order Markov chain that has: a set of states, a set of state transition probabilities, and a specification of an initial state. ”A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules” [5]. The output of the model’s process is a set of states at an instant of time and each state corresponds to a certain physical event that is observed [4]. The process becomes ”hidden” when the observation is a probability of the state. Therefore, the state isn’t directly observed. In a nutshell, based on specific probabilities of states at given times, a physical event can be predicted. This fits well with motion gesture recognition. Since the hidden Markov model can have different typologies and models, the model that fits the best with motion gesture recognition is a model that describes observations in a chronological sequence. This model is named the left-to-right model. With this algorithm, our team would save specific patterns that correspond to a character mapping by setting precise probabilities of space and time.

5) Distance Metric Learning: The distance metric learning algorithm is a newer technique that recognizes patterns based on a group of known patterns. The unknown pattern is then assigned to the closest known pattern. The difference in the distance metric learning algorithm is the underlying closeness metric distance from Euclidean distance to the Mahalanobis distance metric. This new metric provides more accurate results. The drawback to this algorithm is that it is very complex compared to the other algorithms.

6) Review and Recommendation: The academic journal that suggested these three algorithms performed tests on each of them separately and reviewed their accuracy. The distance metric learning performed with 98.26% overall average recognition rate. The hidden Markov model achieved 94.44% overall average recognition rate and the dynamic time warping algorithm achieved 89.89% overall average recognition rate. In order to decide the algorithm to choose and implement, my team needs to weigh algorithm performance versus its complexity. This means that our team may choose an algorithm that doesn't yield the highest accuracy if another algorithm is easier to implement. The reason for this is because of overall time constraints for our project. With that being said, our team should wisely start with the distance metric learning algorithm or hidden Markov model.

E. Character Mapping

The third and final piece of the project that I will address is character mapping. Character mapping is extremely important because it affects user experience and algorithm success. Since the goal of our application is to create alternative user input via motion gestures, our team needs to map motions to letters and words. Given this, we have some different options for implementing character mapping.

1) Straight Line Simple: One approach is to map 30 unique gestures to 30 different characters. The reason we chose 30 is to allow for 26 different letters in the English alphabet and 4 additional gestures for special characters such as periods and commas. In order to make it easier for the user to learn the gestures, the gestures would be simple straight line movements of their device. However, mapping 30 straight line gestures would be an issue because 360 degrees divided by 30 is only 12 degrees of space for the user to attempt to generate text. This would be infuriating for the user because the degree for error is much too small. To take this a step further, our application could leverage the gyroscope embedded in mobile devices and initially flip the device on its side, facing upright, or facing downward. Then only 10 movements in space would be needed. That means that users would have 36 degrees (360 degrees divided by 10) of space on the devices axis to gesture. This is one possibility. However, if users wanted to add shortcuts as new gestures, this would limit them.

2) Series of Combinations: Another approach that our team can take is to map a series of gestures to certain characters and words or phrases. In other words, if the device is moved forward, right, right in succession, that would result in a corresponding letter. With this idea, the user would only have to move their device in six different directions: forward, right, left, backward, up, and down. This would allow our algorithm to be more successful. However, the drawback is recognizing when the next "gesture" occurs (either within the combination series, or between characters). In order words, from the previous example, the algorithm would need to know when the user stopped gesturing forward, and started gesturing to the right. Also, the algorithm needs to know when to select a character and allow for the user to gesture the next character. Further, the amount of combinations could become complex, especially if the user adds many shortcuts. This would make it difficult for the user.

3) Object Pattern Complex: Another idea is to gesture specific objects and map them to their respective character. For example, gesturing a "circle" would result in the letter "o." This idea came from the academic journal that was referenced prior [4]. Although this is a valid approach, I think this would be very complex for the user. Even though this might result in a higher accuracy for the algorithm, text generation speed would suffer greatly.

4) Review and Recommendation: After review, mapping gestures to characters, words, or phrases, is difficult for users and the algorithm. As a team, we know that this will be a challenge and users and the algorithm will need to continuously learn

from one another. With that being said, users should be able to map custom gestures to the application. This would result in a more user friendly application and increased usability. Our application should have default character-gesture mapping and allow for custom gestures. Also, the user should be able to edit the default mapping. The default mapping should be only moving the phone in simple straight lines with the phone's initial position altering.

F. Conclusion

This article discussed three pieces of my groups project and their related technologies or methods. These pieces don't comprise the entire project. They were researched and reviewed in detail. In addition, this document discussed my role in my group for this project as well as a high level overview of my groups goal for the project.

7. Tech Review - Lauren Sunamoto

A. Introduction

The goal of this project is to provide an additional mode of input to mobile phone users including those with motor function or dexterity issues. We hope to improve the accessibility of text input, all while minimizing the effect on typing speed. My responsibilities include reviewing mobile platforms, researching computer accessibility and statistics of letters. Each topic provides unique insight that our group will consider in selecting the specific technologies and methods we will use.

B. Mobile Platforms

For our mobile application, there are two options we are considering for its Platform: iOS and Android. These are currently the two most prominent mobile platforms that exist. To determine which is better suited to our requirements we must examine each platform and consider key differences between them.

1) iOS: iOS is a closed platform with open source components and so, customization is limited. And yet, it is known to be very secure with a prioritization of privacy, security, and reduced risk of malware. [26]. The platform's reputation for better security contributes to iOS having more penetration in the enterprise market. [27]. All applications are purchased from the Apple App Store and software updates are available for older iOS devices. iOS has a higher revenue per user. iOS applications take less time to develop as code is written using Swift, Apple's official programming language. However, the process of publishing an application is time consuming because of strict controls put in place by Apple. Also, development and testing must be conducted on an iPhone. Lastly, this platform was requested by our client to be used. [26].

2) Android: In contrast to the iOS platform, the Android platform is notably open source and so, there is more flexibility in which customization is much easier. But, this comes at the cost of low security and privacy controls. Android application development is known to be more complex than iOS with its applications needing to be written using Java. Some estimates put Android app development as thirty to forty percent slower than iOS on average. And yet, Android applications can be published more easily and quickly that iOS applications. There also many different types of Android smartphones that exist with differences such as screen sizes. [26]. Also, Androids are known to reach a more broad global audience, whereas iOS' audience is mostly limited to Western Europe, Australia, and North America. [27].

3) Selection: Because our client prefers iOS, we will use iOS rather than Android. Later, we may decide to launch the application on Android once it is established.

C. Human Computer Interaction: Computer Accessibility

Computer accessibility in human-computer interaction refers to the accessibility of a computer system to people despite disability type or severity of impairment. To improve the accessibility of text input, we will hopefully create features that prioritize individuals with specific visual, hearing, motor or dexterity impairments.

1) Visual Impairments: Visual impairments include blindness, low vision, and color blindness. [15]. In developing our application we can build features that rely on other senses such as hearing and touch. We can use haptic technology which involves sending a user feedback by creating vibrations in a mobile phone. Such feedback is important in recall and recognition of users as they learn to use the application. For example, a vibration can alert a user to the generation of a letter, and aid the speed in which they ”type”. Sound queues could also be used but may not be as useful for those with hearing disabilities.

2) Hearing Disabilities: Hearing disabilities range in severity from total deafness to slight loss of hearing. [15]. As mentioned above, sound queues would not be very effective. Therefore, haptic technology would be a better alternative method to provide user feedback. Individuals with hearing disabilities could also benefit from screen prompts but such feature would not be as useful for those with visual impairments.

3) Motor or dexterity impairments: Motor or dexterity impairments encompass a large variety impairments including total absence of limbs or digits, paralysis, lack of fine control, and instability or pain in the use of fingers, hands, wrists, or arms. [15] Some of these individuals would especially benefit from alternatives to standard input methods (i.e. keyboard). Although, not all individuals that fall under such category will be able to use our application because it requires some fine motor skills for precise generation of characters as well as gross motor skills with wrist movement. But, there are specific elements of our application that we can do differently from traditional keyboards.

a) Touchscreen Manipulation: Traditional mobile keyboard require steady and precise tapping on small key spaces which can be difficult for certain users such as people with Arthritis or those with uncontrollable hand tremors from Parkinson’s disease or Multiple sclerosis. [16]. Therefore, our application should not require complex use of the screen. For example, in addition to defining gestures for alphabet letters we should define a gestures for deleting and shifting. It should also be error-tolerant (e.g. deletion should not necessarily be easy to do).

b) Customization: To accommodate the variation in impairments, we should allow for some customization of our application. For example, the typing speed should be easily adjustable. The transition between characters/gestures can be adjusted as well as the recognition and translation of characters/gestures.

4) Conclusion: In developing our application we will consider the importance of computer accessibility by creating features that meet the needs of certain individuals with specific disabilities. The functionality of our application will rely mostly on gesturing and user feedback will include haptic feedback which usable for a wide range of users.

D. Letter Statistics

This application must be able to generate text from motion gestures with a mobile device. We will define a set of gestures that is easy to complete and replicate, and allows for a ’typing speed’ of at least ten words per minute. We will include gestures for the thirty-size standard characters as well as include support for various modes of punctuation, white-space and capitalization. Letter statistics can be useful in determining how we define the different gestures. Furthermore, like the optimization of keyboard layout letter statistics can be used to improve typing speed and decrease user errors.

1) Letter Frequency: To ensure our goal typing speed, we should prioritize the most commonly used alphabetic characters and pair the least used characters with gestures that are more complex due to the need for distinct motion-based gestures. Based on a sample of 40,000 words, the letter 'Z' occurred 128 times and is the least frequently used English letter. Therefore, it should be defined in a way such that it is not easier to replicate than the most frequently used letter 'E' which occurred 21,912 times. Also, using these letter statistics we can consider how the letters relate to one another. It is easier to identify vowels because most letters appear before and/or after them. . Furthermore, our application should be error-tolerant by defining the backspace character so that it cannot be misinterpreted as a common character gesture.

2) Digraph Frequency: In addition to letter frequency, we can look at the statistics for digraph frequency to improve user 'typing speed' as well as accuracy and ease of use. Digraphs are pairs of letters. For example, based on a sample of 40,000 words, the most common digraph is "th" which occurred 5,532 times. This is consistent with the fact that the word "the" is the most commonly used word in the English language. This information is useful in defining motion-based gestures because it is important to consider the transition between letters, especially those that are frequently next to each other. We must consider that there are specific gestures that will be difficult to create and discern following another gesture. [28].

3) Conclusion: We will be able to maximise the efficiency of typing on our keyboard by using letter and digraph frequency statistics in the defining of our motion-based gestures. The most common alphabetic characters and digraphs will be prioritized over the least common. Overall, such considerations will help in improving user experience by prioritizing ease of use and design.

8. Tech Review - Zachary Horine

A. Introduction

The purpose of this paper is to detail research topics to be focused on for our project. These topics each provide insight into the technologies and systems that will need to be incorporated into our project. This paper will cover three of the 12 total topics to be researched for our project. The topics to be covered are (ii) Data storage Methods (iii) Categorization - Finding the Ground Truth and (iv) Grammar Statistics. All of these topics provide unique insight into a particular area, and expand upon previous knowledge to create a base of understanding that our group can work from.

B. Data Storage Method

In order to persist gesture data inside our app, we will need to store a set of gesture point arrays that encode the base data for each gesture, and allow the system to match the gesture that the user has input, and reference it to a character. There are many ways to do this, but the most commonly used are (a) an SQLite database, (b) external files or (c) as a set of key pairs stored inside the app configuration files. Both Android, iOS support all three of these methods, and Windows devices can be configured to support them, although they don't support SQLite databases by default.

1) SQLite Databases: SQLite databases are an efficient way to store highly structured data. In a database, there is a set of tables, that are linked together by a set of relations. these relations allow for the data to be read, interpreted and added to efficiently. [29] SQLite in particular has the benefit of being server-less, and lightweight. because of this, it can be implemented in mobile apps and services without much concern for process, power or storage cost. [29] Because of this, SQLite runs very well in both the Android and iOS architecture, and has been made accessible and easy to use for developers wanting to store relational data in their apps. Windows based systems don't have explicit support for SQLite, although there are libraries that implement similar functions. [30]

Overall, an SQLite database would be a good option to store our data in, as they are efficient and easy to use, but we would have to come up with a database schema that fit our data, which could be difficult.

2) External Files: External files are extremely useful when it comes to application design. sometimes, when the data that you have doesn't fit into a specific data model, it can be easier to store it in a format of your choosing and read it in yourself. There are some drawbacks to this method though. Depending on where you store your data, the data can be accessed by the user and be moved or deleted. Although on all systems, there are private data stores that can be used to keep your data safe. These files often sit close to the application code so that they share the same permissions. One of the other drawbacks is that it is more difficult to keep track of your data, and where it is stored. This could be remedied by using one of the other data storage methods though, as it would provide a 'link' to the location of the data.

Overall, the concept of storing our gesture models in their own files makes the process of reading just one gesture in more efficient, and allows us to more easily add and remove data throughout the development process.

3) Key Pairs: Key pairs are an easy way to store small amounts of data, and are useful for things like usernames or device keys. They are very specific on the types of data that they can store, and generally allow for personalized data to be stored without resorting to creating a whole database or file system.

On iOS, these keys are stored in a '.plist' file within the app package. Because of the way the file is stored, there is a maximum file size of 4GB (file-system restriction). [4] This, along with the fact that the file has to be read in every time the app loads, or the first time data is requested, means that the amount of data needs to be limited, and shouldn't be used

to store large data structures. The only difference with Android devices is that the 'Shared Preferences' objects are stored in an xml file within the system. [5] Otherwise they act the same as an iOS device. [31] The case is similar with windows devices, although their data is stored more centrally, instead of being put close to the rest of the app's data. Windows devices store their data in the 'appdata' folder of the user, and can be configured to store data as key pairs or as custom files. [32] Overall, while this may be a good method for storing some user data, it is not the best way to store gesture models themselves, as they are made up of a larger table of points, and there are too many of them to efficiently store in a model like this.

4) Best Option: After reviewing these methods, I believe that there are a couple different options we can pursue. The most promising option is to store our gesture models in external files, and creating references to these files and their locations in a set of app key pairs, or as static variables in our code. Doing this would allow us to format the data in a way that makes sense to us, all while maintaining a reference and organization to our files.

C. Categorization engine - how to find a ground truth

1) What is a Ground Truth?: In general, a ground truth is a data set that defines the base observations that all observations made after its establishment are based on. The concept of a ground truth is used extensively in image recognition, and is heavily related to a scientific assumption. In this way, it can be used as a basis of observation, and allows the observer to determine whether or not the data that they have gathered fits the model or not. [6]

2) What is included in a ground truth?: At its base, the ground truth is the simplest, most basic, and neutral form of the data expected. However, one set of perfect data is not enough to define the base case of the model. A ground truth has to not only include the optimal solution, but it also has 'sub-optimal' solutions that are still in the threshold. [7] It also includes solutions that are not in the data-set, which help to filter out data that doesn't fall within the confines of the desired outcome.

In image recognition, the ground truth also has multiple layers. All of the layers help to filter the data, and make it easier to pinpoint what is different from the ground truth. By looking at things such as the diffusion, reflectance, shading and the specular components of an image, it is easier to see what part of the image differs from the ground truth, which makes it easier to correct later. [8]

3) How can we apply this?: Detecting gestures is difficult for many reasons, but they all lead to one thing: no two movements are ever the same. A lot of this has to do with sensor accuracy, as the accelerometer in a phone is 'precise' but not very accurate. The little irregularities in motion sensing stack up, and very quickly make pulling the true motion out of the noise very difficult. The other complication is that humans never recreate the same exact gesture, and no two humans will perform the same gesture the same way. because of this, the motion itself is not accurate to the model. This is why a simple 'perfect' gesture can't be used as the ground truth. By using a ground truth in multiple axis, as well as tolerances and negatives, we can eliminate bad data, and generate the most probable path.

D. Grammar statistics (frequencies and common patterns)

1) Keyboards: One of the important things when designing an input device is the layout of the input 'keys'. Many traditional keyboard layouts were designed around the English language, and the letter frequencies that are used to type some of the most common words. One of the most common examples of this is the Devorak keyboard, which prioritizes vowels and commonly used consonants. [22] we will have to do a similar thing with our keyboard, but we will also have other issues

to attend to. Because of the nature of gestures, there are certain gestures that will be difficult to detect if they come after another gesture, and some that are easy and flow into one another. In this way, we are dealing with some of the issues that came along with typewriters (frequent jamming), and the supposed reason that the qwerty layout was invented. In reality, the layout was invented to slow down faster typists, which is exactly the opposite effect that we want to have. [23]

2) Frequency in Words: Instead, we should pair more readily accessible gestures with the most commonly used characters, and associate the least used characters with 'more out of the way' gestures. We also need to look at the characters that commonly come after one another, and make sure the par of gestures work well together. A study done by the College of Math at the University of Illinois shows how likely a given character is to come after another. In their study, there were a large number of characters that never appeared together, and a list of the most frequent ones, which included 'th', 'he', and 'an' as the top three. [24] This makes a lot of sense, as the three most common words in the English language are 'the', 'of' and 'and'. [25]

3) Conclusions: If we can use this information to map our gestures to letters, then we can improve the efficiency of typing on our keyboard. By taking into account all of the lessons learned in keyboard design, and incorporating them into our app, we can create a better experience for our users.

9. Weekly Blog Posts

Fall - Week 1

Lauren: Since receiving our new project, we were able to talk with our client on multiple occasions. We discussed the scope of our project as well as some technical aspects such as hardware that allowed us to finish the individual problem statements and requirement document draft. So far, we haven't faced any significant problems. We have established Discord as our mode of communication and used Google Docs as well as Overleaf to finish each group assignment together. We meet in person before such assignments are due usually after capstone class. We plan to meet soon in person and finish the upcoming assignment due Sunday.

Zachary: We currently have an objective, and a way to accomplish it. We are going to be creating a mobile phone motion gesture recognition keyboard, and we have a general idea of the things we will need to learn to accomplish the task. We haven't had any issues or problems so far, and are planning ahead to make sure we can deliver a quality project in the time allotted.

Jeremiah: This week has been all about catching up. My group was reassigned to a different project almost a week after project selections. We had to scramble to meet our client and figure out what the new project entailed. Luckily, our client is Scott, so that helped. I am excited about this project and am learning/researching/thinking everyday about it.

Changkuan: Originally, we were going to design an input keyboard by converting the movements of a mobile phone into Morse code. After meeting with our client, we were agreed to create a gesture recognition keyboard, which will convert gestures motion from using the phone's accelerometer and gyroscopes into different specific inputs. We have the basic idea of how we are going to design and working on this project. The problem statement and a simple draft of the requirement document were finished. The main problem that we are facing at this point is to create different unique gesture motions for each specific input and what we are going to use to make the keyboard more intelligent (e.g. knowing when a word or a sentence has completed). Our plan for now is to create gesture motions for different specific inputs, including all the alphabets and special symbols. Our goal is to make this keyboard as efficiently as possible, so the gestures have to be easy to remember. Additionally, we are going to keep working on the requirement document then we will decide our next steps when we get feedback from the client.

Fall - Week 4

Lauren: The group final problem statement as well as the first draft of the requirements document were submitted on time. During our weekly meeting with our client we talked about the usability of our application and finer details such as how we will create distinct gestures and use the accelerometer and gyroscope data of mobile phones. We will submit the second draft of the requirements document tonight. No significant problems have arisen. We will continue to communicate through Discord and meet with our client every Thursday.

Zachary: We are currently on track. we completed the requirements document, and have a good understanding of what we are aiming to achieve. We have started discussing how we are going to implement our project, but we don't have the whole thing laid out yet. We have done a little bit of research about how we are going to gather data, and so fa everything seems to be working out.

Jeremiah: This week has been about further clarification for our project and figuring out the requirements. Towards the end of last week, we started a rough draft of the requirements doc. This week, we added and revised that draft. There weren't any problems this week. We are looking to work on the tech review next week.
Changkuan: We met with our client and discussed the ideas we have in mind to further build up the way the application will work. We are currently working on the second draft of our requirement document, and added more details to what we originally had as well as reworked on the Gantt chart using Lucidchart instead of Tex. We are still having the same problem from last week, which is creating different kinds of gestures. After reviewing the problem statement and requirement document draft, our client gave us some of his idea of different gestures for specific letters and symbols using the accelerometer(angles and facing positions of the phone). We are also still deciding on IOS or Andriod. Our plan for the following week is to keep adding more information to the requirement document as well as gain more knowledge of technologies that we need to build the keyboard. Additionally, come up with different gestures for inputs.

Fall - Week 5

Lauren: We discussed with our client the research we should conduct for the tech review and submitted our second draft requirements document. We have not faced any significant problems and plan to work on the tech review document.
Zachary: We have a good idea of what we are expected to complete, and are in the process of figuring out how we are going to do it. We met with Scott on Thursday, and talked about the different areas we need to focus on, and the ways in which we can approach the tech review. Currently there aren't any problems, our plan is to do research, learn as much as we can, and create a solid tech review.
Jeremiah: This week I was very sick throughout the entire week. This was an unforeseen blocker this week and it was very unfortunate. Our group luckily had the opportunity to re-do our requirements doc because it wasn't up to standard. Then, we started working on the tech review document that is due this Sunday. For me personally, it is all about catching up because of my sickness that I came down with. This gives me motivation going forward, because of the added stress.
Changkuan: We met with our client and we discussed about the important technologies that we are going to need for this project. I don't have any problem at the point. My goal is to fix issues based on the feedback from the technology review document draft. Our plan is to work on the technology review document individually at this point. We weren't quite sure how exactly the document should look like. Will revise the document once got feedback.

Fall - Week 6

Lauren: Since the last blog, I have submitted the final version of the tech review document. We were unable to meet with our client because he had a lecture scheduled during our meeting time and decided to meet next week instead. We plan to discuss the design document and meet to discuss its contents and divide responsibilities.
Zachary: Our tech review went very smoothly. I learned a great deal about topics that will help us in designing and implementing our project. We haven't run into any problems yet, and are on track to complete the Design Document.
Jeremiah: This week, I spent a lot of time working on the Tech Review paper. As the week progressed, I was able to put in more quality work. This is because I was still coming off a sickness. After the Tech Review, I spent some time taking a look at the Design document and how to start and where to go from there. I feel confident about the Tech Review and hope for the best. I really enjoyed my responsibilities, especially researching and reviewing machine learning algorithms for input pattern recognition aspect of our project. It is very fascinating.

Changkuan: This week was not too group heavy, we are working on our individual technical reviews. We met with our TA Richard and talked about the design document that's due next week and how we will be required to go into more detail concerning actual implementations, for example, the types of algorithms we might use. We did not have a chance to meet with our client this week. Since I did not get any feedback from anyone for my technical review document, I am still not exactly sure how this paper should look like. As a group, we are trying to think of any possible implementations that we need for the design document after we all finished the technical review paper. Our plan for the following week is to finish the draft of the design document, then we are meeting with the TA and client next week to make changes to the draft and work from there for the final draft.

Fall - Week 7

Lauren: Since the last blog, we have submitted the rough draft version of the design document. Also, we met with our client to share what we learned through the tech review. No significant problems have arisen. We plan to submit the final design document by next Friday.

Zachary: We have completed our design document, and I think we have enough knowledge to get started on our project. We haven't run into any problems yet, and we are starting to look at other solutions that exist, and see if we can use any of the things learned in them to improve our project. The next step is to finalize our design document, and hopefully get started!

Jeremiah: This week wasn't as time extensive as the last few weeks, which was nice. As a group, we worked on the design document draft. I think we have a decent start, but we might need to work on it a bit more. Further, the meeting with our client was really helpful as we updated him what we learned from our tech review assignment. This was awesome because he gave some good feedback.

Changkuan: We met with our Ta and went over the design document requirements and finished our draft of the design document. We also met with our client, we talked about the technologies we chose for the technology review document. We are still trying to figure out what exactly we are going to add to the design document. Our client suggested a project that's similar to what we are working on. We are trying to get that project to work on our mobile devices to see how it performs. Our first goal for next week is to first finish the design document then try to get the example project to work on our phones. Additionally, we are trying to figure out the algorithms and gestures for different inputs.

Fall - Week 8

Lauren: Since the last blog, we met with our client to talk about our progress the design document. We discussed splitting up work among team members to work on during winter break but didn't go into much detail. We have not faced any significant problems. After the meeting we split up the project's components through Discord. We plan to finish our individual parts by the end of today and then, compile and organize them on Saturday to meet the design document deadline.

Zachary: We are working on improving our design document, and are working with our client to make sure that we are formulating a project that is attainable and will satisfy the project requirements. We haven't run into any problems yet, and Scott is working with us to make sure that we have access to Mac computers to design our application with. Our plan is to finalize our design and start working on a prototype.

Jeremiah: This week, I spent my time on the design document. The meeting with our TA seemed to be helpful and we have tried to incorporate what he said in our document. The minor extension seemed to help clear up some things however. I hope for the best for this design document for grade submission!

Changkuan: We are still working on the second draft of the design document. Our client and TA went over different aspects that we need on the design document. We figured out the main components that we need to build our program. We think we have the main components we need to build our project, however, we are not sure if we covered everything. We are still working on the design document and hopefully get feedback from the TA and our client and work on it from there. Our plans will be based on the feedback from our design document. We will try to better our project guide line for us to start on the coding for next term.

Fall - Week 9

Lauren: On Saturday we submitted our final design document. We have not faced any significant problems. We plan to revise our document a bit more before our client sees it and get his approval by next Friday.

Zachary: Our project is going well, we have completed our design document, and have gotten feedback on a couple of things that we should update before we turn it in. We will get approval from our client in the coming week, and We are almost ready to start development. I believe we have the proper resources to do so, although it would be beneficial if our two group members without Apple laptops were able to access the two machines in the capstone lab remotely.

Jeremiah: This past week, we spent our time on the design document. I think we all did a good job and had massive improvement over the first iteration of the document. Our time with our TA really helped us.

Changkuan: We finished the second draft of the design document. We met with our TA Richard today and he suggested that we should fix the first half of the document to be the more like the second half, only focusing on “how” instead of “research”. We don’t have any problems at this point other than edit the first half of the design document. Our plan is to fix the first half of the design document and get it approved by our client and work from there.

Winter - Week 1

Lauren: Following the beginning of the term, we have scheduled meetings with our TA at 4pm on Thursdays and reached an understanding with our client. No significant problems have arisen. We plan to meet with our client when necessary by letting him know at least a day in advance if we need to meet with him. We plan to meet Friday to begin developing.

Zachary: We have started development on our app. We are working on generating a prototype that we can show to our client and demonstrate some of the features that our app will have. We haven’t run into issues yet, and our plan is to have an initial demo ready by the end of next week.

Jeremiah: This week, we met back up with our group members after the break. Prior to the break, we finished all the required documents. Our next step is development. We met with our client this Tuesday and he wants to see some progress (code) by next week. We have begun already and look to continue development.

Changkuan: Finished the team critique and reflection. I met up with the group and we planned out our coding goal for this term. We were trying to get the sample codes working on IOS devices, however, we still getting bugs while we were building the codes in XCode. We are trying to get the simple codes working before we start the actual coding for our project. We will talk to our client next week and hopefully get some advice from him.

Winter - Week 2

Lauren: We began developing by working with some source code provided by our client. We had some trouble because it was outdated. But, we were able to get the application on a phone and are planning to use it for reference. We updated our client on our progress with an in-person meeting. On GitHub we have created issues which we plan to complete in 2 weeks. Each group member was assigned individual issues.

Zachary: We are slowly making progress. We were able to install the example code on an iPhone 6s on Thursday, and it gave us a little bit of insight into the process and work required to build our app. From here, we can start building our app, and use the example code as a template on how to use the GRT library to collect and detect gestures. We are ready to get started building our app.

Jeremiah: This week, our team made some progress with examples from previous code that our client showed us. We found out that the code is only compatible with Swift3, not Swift4. This causes us to create more code from scratch. After our meeting with our client, we talked about splitting up specific tasks/issues for each member to work on. We plan to work on those tasks from now to next week and show progress on the application.

Changkuan: From the past week, we were able to build the gesture recognition toolkit with Swit3 on the iPhone with IOS 12.The test code of the GRT was written in Swift 3, which means that we weren't able to get it to run in iPhones that are in IOS 13. We've tried to build it with Swift 4, however, unsuccessful. Here are the next steps that our team came up: 1. get accelerometer data in real-time (swift) 2. generate keyboard buttons and get callbacks (swift) 3. figure out how to call main-app functions from keyboard 4. detect gesture delimiters (return to home) 5. generate test training data (into CSV file) 6. explore GRT library and determine the best way to detect features in data 7. be able to distinguish test data sets using GRT functions

Winter - Week 3

Lauren: I have done research regarding my assigned issue of how to call main-app functions from the application extension. I have looked at various articles detailing similar projects and continue to refer to Apple documentation. Also, I have been trying to learn more about XCode and Swift. I have not faced any significant problems. I plan to resolve my issue by the end of next week.

Zachary: Our group is on track. We are working on figuring out how to get information into and out of the various parts of the app. I am working on getting data cleaned up and being able to detect gestures. There aren't any issues, and we seem to be on track.

Jeremiah: This past week, we organized separate issues for each team member to work on separately. We discussed a time frame for finishing these issues and we should be done or close to done next week. Working on the mac machines in the lab make it somewhat difficult to develop because we need the GUI to develop our application, not just the command line. There has been a lot of learning about Xcode and how to develop iOS apps for me.

Changkuan: We've come up with specific plans and goals for each of us to work on. My goal is to be able to create buttons on the custom keyboard and make connections with the buttons to trigger specific events. I was still trying to get to know XCode a bit more and doing more research on creating button functions on the IOS keyboard. Nothing major, we are still discussing how many buttons are appropriate and the position of the buttons. My plan for this week is to create a custom keyboard and add buttons to it. I will also try to figure out how to make connections with the GRT if successful.

Winter - Week 5

Lauren: I worked on getting accelerometer data from the keyboard view controller and displaying it with text fields. I researched how we integrate the GRT library. I have not faced any significant problems. I plan to continue researching how to integrate the GRT library as well as the relationship between our app extension and its container app.

Zachary: All of our pieces are getting close to alpha stage. I am getting close to a workable solution for gesture input and output out of the GRT library. I believe we are on track for the alpha build, and everything is going well.

Jeremiah: This week was a little better than last week. Still not as much progress as I hope, but better. More development and figuring out things. I plan to get much more development done this weekend and next week.

Changkuan: I am still trying to get a better understanding of developing using Swift (custom keyboard extension to be specific). I was able to create a button to output text and the next keyboard button that came in the package. I was struggling with the positioning and styling of the buttons. My plan for this week is to create a button that has a delete function and a button to hide the keyboard.

Winter - Week 6

Lauren: I started working on the expo poster by adding to the background and problem section. I also got a new updated library working which is a much better alternative to the GRT library we were trying to use. I have not faced any problems. I plan to help in finishing the poster and prepare for our design review next Tuesday.

Zachary: We are working on alpha functionality. We decided to change the gesture recognition library that we are using from the Gesture Recognition Toolkit to the CoreML library within iOS. This should make the project easier to manage. We have our design review on Tuesday, and we are should be ready to go.

Jeremiah: This week was busy but it was really good. Our team worked on the poster and we finished it. We have been working to be ready for the design review as well, which is next Tuesday. Although one of our problems we have right now hinges on the success of the project, our client (Scott) said to go ahead with the design review anyway and we might get help there.

Changkuan: I have been working on the poster with the team. Have done much work individually this week, was just testing around with the positions and styles of the buttons, I am struggling with connecting the .xib file with the ViewController.swift. I wasn't able to have the modules that I created in .xib to display on the keyboard user interface. My plan is to try to work on the functionalities of the keyboard as much as I can before preparing for the design review next week.

Winter - Week 7

Lauren: We presented our project at last Tuesday's design review. There have been no significant problems. We plan to collect accelerometer data and work on defining distinct gestures. This data will be used to train our machine learning model in CreateML.

Zachary: We have gotten most everything at least partially working. We are now working on training our CoreML model. After this is complete, the app will be mostly done. The plan is to have at least half of the gestures done in two weeks.

Jeremiah: This week, our team had the design review. I thought it went well and we received some good feedback. We were nervous coming into the review because we had a fairly large problem that we couldn't solve with our libraries. However, just before the review, we pivoted to a new library which caused us to create a new workflow for the development of the

application. It requires additional tools that we need to implement. This means that we seem to have one hurdle solved and we will continue to develop the rest of the app to get a working version. We will have to do a lot more testing with this new workflow.

Changkuan: I have been most preparing for the tech review presentation earlier this week (which was on Tuesday). Additionally, I was writing feedback for other groups that reviewed on the same day. Don't have any problems at the moment. Our plan is to review the feedback we got from other groups and improve the functionalities of the keyboard.

Winter - Week 8

Lauren: Using the application, SensorLog, I have collected accelerometer data for a few different gestures. I have not faced any problems and plan to collect at least 20 csv files for each of the remaining gestures.

Zachary: We are making good progress towards the completion of our project. Chang is working on the keyboard view and layout, which is mostly done. Jeremiah is working on creating a user training function to go in the main containing app. I am currently working on training the CreateML model, which requires generating a Training dataset of 20 samples for each gesture. Lauren is helping me with this. We have tested the integration of the model with a barebones version of the app, and it appears as though everything but a small data-flow issue works, which I plan to have working by next week. We should have all of the Training data collected in the next two weeks, at which point our app will be fully functional, and will only need a little bit of polishing. I think that even with our switch a few weeks ago we are on track to have everything done.

Jeremiah: This week, we made more progress with the new pivot. The new workflow is working really well. We divided the work up well and are close to getting the finishing touches done. We have accrued lots of test data for training the gesture models. I have been working on the containing application for usability and learnability.

Changkuan: We are on track and making good progress. Everything is mostly done, Zach and Lauren are working on getting more training data for some remaining characters. Still waiting for everyone to complete their own task, not having any blockers so far. Our goal is to combine everything together and test the "final" product once everyone is done.

Winter - Week 9

Lauren: I generated accelerometer data using the app, SensorLog, for the rest of the Alphabet gestures. I have not faced any significant problems. I plan to work on our project's Expo poster and anything else that needs to be done.

Zachary: We are making progress towards a beta build. We are working to diagnose an with the model integration, but our app should be functional once we figure it out.

Jeremiah: This week we have been still working on the project and finishing it up. We had some exams this week and worked on presentations as well, however. This limited time to work on the project. We are set to finish next week and fix a few things with our poster.

Changkuan: Not much going on this week. Still waiting for everyone in the group to finish and combine everything together. Our plan as a group is to improve the poster and start working on the demo video.

Winter - Week 10

Lauren: I created testing data for our machine learning model with SensorLog, and finished revising our expo poster. I did not face any significant problems. I plan to help my group members finish our demo video and final report.

Zachary: This week we resolved the main issue that we had with our project. we had a "ERR_BAD_ACCESS" error when attempting to access our CoreML model. We resolved this by updating and re-building the model. We are at the point where we can put everything together and the app should work. We are collecting more data, and as we go, the prediction will only get more accurate.

Jeremiah: This week , we finished up. We worked on the video and beta functionality that is due. The coronavirus was an interesting thing that was a blocker kind of. We are excited for the expo!

Changkuan: We have been working on the poster as well as collecting gesture data for individual characters. We've updated the poster based on the feedback from the first draft. For our keyboard, everything is mostly done besides the gestures. We are trying to train and test the gestures to be as accurate as possible. Our plan is to try to put everything together before the demo video, also to work on the final report that's due next week.

COLLEGE OF ENGINEERING | Electrical Engineering and Computer Science | CS11

GESTURE RECOGNITION TYPING

An exploration into accessible typing, and using gestures to augment traditional typing on iPhone

AN END TO TYPING

Figure 1: Really old phone

In the last few years, modern devices have become capable of acquiring information in many new ways. As the traditional keyboard becomes something of the past, improvements in technology introduce new possibilities. Heading towards a post-keyboard world, more advanced methods of input are needed to further improve the accessibility of text input. Current input methods do not benefit users with phones that have broken screens that do not register touch input properly as well as users with motor function or dexterity issues.

MODEL TRAINING

- SensorLog – application used to read out accelerometer data from an iPhone, and save it as CSV files
- CreateML – framework used to create and train our machine learning model; automatically computes precision and recall metrics on selected datasets
- CoreML – format of the CreateML model; used for easy integration into our application through Xcode
- Xcode – used to get accelerometer input values from a user, pass data to our model, and display predictions to a user as text.

Figure 2: Model training workflow

Oregon State University

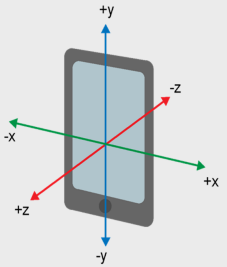


Figure 3: Mapping of Accelerometer directions to phone orientation

DATA COLLECTION & CLEANING

Our dataset consists of 3-acceleration sequences of gesture classes for a complete alphabet and punctuation set. Our training data will be collected at 30Hz and we will use 1-2 seconds of data samples.

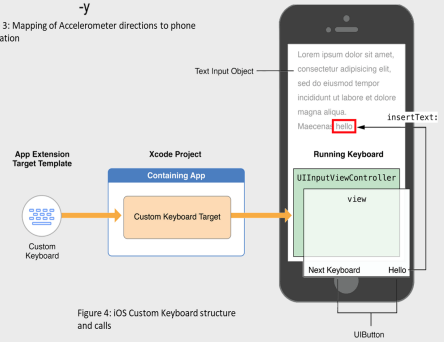


Figure 4: iOS Custom Keyboard structure and calls

LEARNING

The container app allows users to learn the gestures we have assigned to letters and test the accuracy of the gestures.

USAGE

The app extension (keyboard) allows users to input gestures, which will be interpreted against a model, and return the matching letter.

GESTURE CHARACTER SET

Our dataset consists of 26 unique motion-based gestures/characters which are defined by the mappings shown in the figure to the right (Figure 5). Each gesture is fairly easy to quickly complete and replicate, and can be accurately differentiated by our machine learning model.

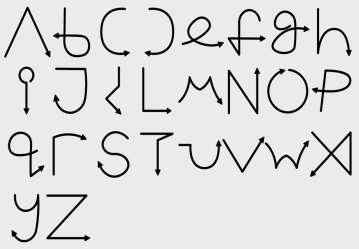


Figure 5: Gesture character mapping for gesture recognition

OUR TEAM

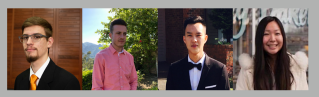


Figure 6: Project members listed below from left to right

Zachary Horine	horinez@oregonstate.edu
Jeremiah Kramer	kramerje@oregonstate.edu
Changkuan Li	lichangk@oregonstate.edu
Lauren Sunamoto	sunamotl@oregonstate.edu

ACKNOWLEDGEMENTS

Client: Scott Fairbanks
Scott is a Project Instructor in the school of Electrical Engineering and Computer Science here at OSU. He provided the idea for this project.

RESULTS AND FINDINGS

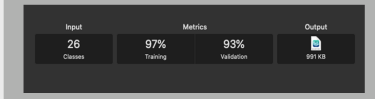


Figure 7: CreateML testing metrics

- According to CreateML's precision metrics shown above (Figure 7), our model is fairly effective at applying a label only when appropriate for a given gesture (few false positives)
- According to CreateML's recall metrics shown above (Figure 7), our model is fairly effective at finding all of the relevant examples of a gesture (few false negatives)

Fig. 3. Poster

11. Project Documentation

A Gesture Recognition Keyboard for iOS. Our app provides a system keyboard that allows the user to use a series of motion gestures to type out text in standard input fields.

This project includes:

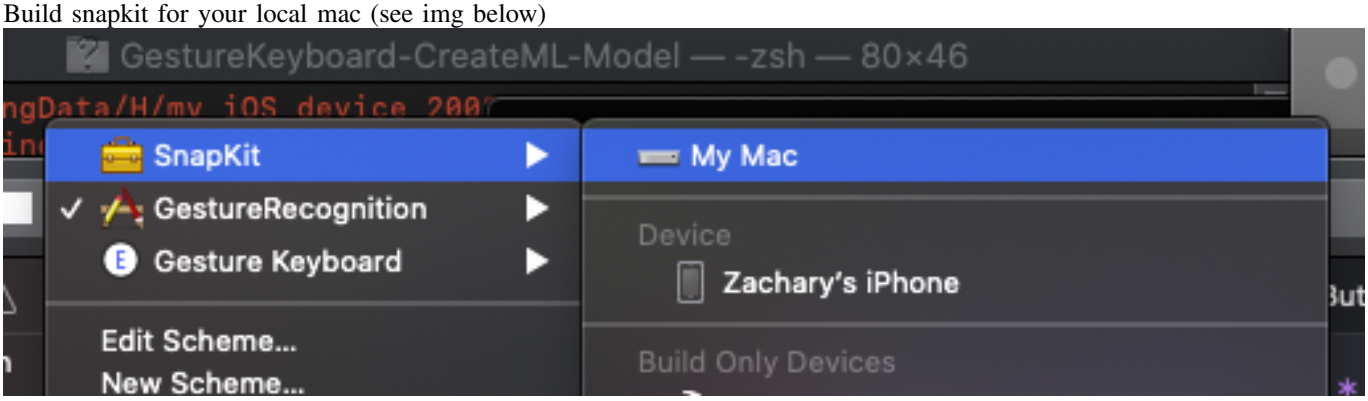
- Building a machine learning model to test user gestures against (We used Apple's CreateML and CoreML tools)
- Creating training data and using this to generate a viable model for matching
- Making a keyboard interface for users to interact with and tell us when we should be listening for gestures
- Creating a user trainer for use by the user to learn our gesture set

To build this project:

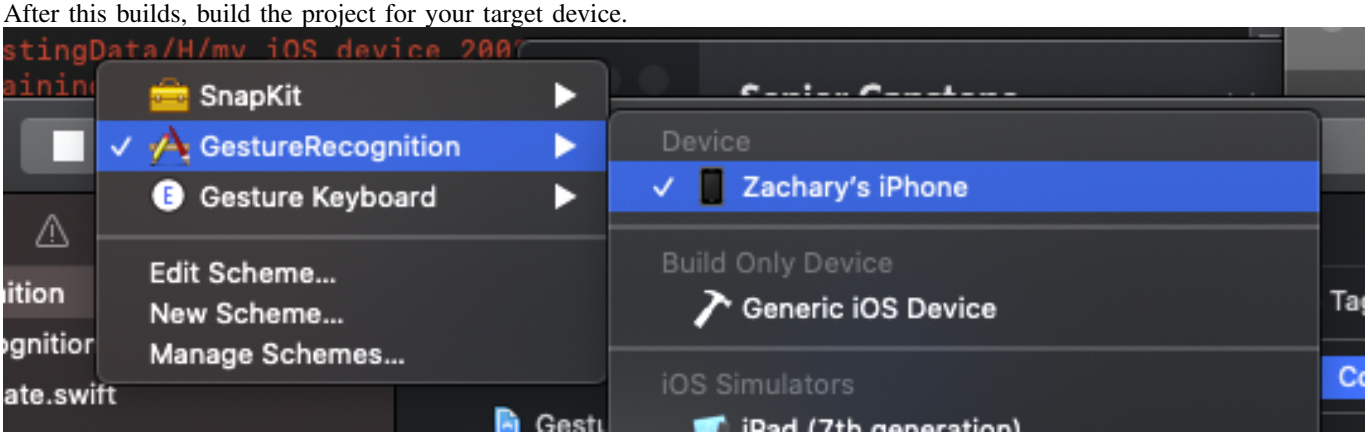
- download the git repository, XCode 11.3, and MacOS 10.15 (if you want to build your own CoreML model)
- run "git submodule init" and "git submodule update" to pull Snapkit into your repository

**This only works if you have an SSH key registered with Github. Info here:
https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)

- open the main project file (GestureRecognition.xcodeproj) in XCode
- Build snapkit for your local mac (see img below)



- After this builds, build the project for your target device.



12. Recommended Technical Resources for Learning More

The link below details how to improve the model's accuracy for CreateML. This is very important because we want our models to accurately predict the correct character mapping given the inputted gesture.
https://developer.apple.com/documentation/createml/improving_your_model_s_accuracy

The link below helped us develop the containing application for our keyboard extension. Swift is an object-oriented language, but there is a learning curve for getting used to the platform.
https://www.raywenderlich.com/ios/paths/learn

The link below describes CreateML as a whole. This helped us understand the tool that we ended up using.
https://developer.apple.com/documentation/createml

The link below describes SensorLog, an app that we utilized to gathering "training" data for our CreateML models.
https://cloudacm.com/?p=2932

13. Reflections

A. Changkuan Li

- What technical information did you learn?
  I learned to code using Swift4 and the basic techniques of iOS development in Xcode. I was able to figure out how different tool kits interact with the application and the way that they affect the overall look of the user interface. In addition, I learned to collect and gesture data using the SensorLog application as well as the basics of how to train the data using CreateML with XCode.
- What non-technical information did you learn?
  I learned the importance of how to work as team and communication between our teammates, TA, and client. Through this project, we encountered some difficulties that we did not anticipate in advance. I learned to make more detailed preparations and anticipate the worst in the future so as not to delay the progress of a project.
- What have you learned about project work?
  I learned that the project requires precise preparation and implementation, these include: planning, collating materials and documentation, implementation and testing.
- What have you learned about project management?
  I learned that reasonable planning and time management are an important part of project management. Through this project, I found that things didn't went as smoothly as we originally thought, we need to collect more information and make more preparations in advance.
- What have you learned about working in teams?
  I learned the importance of communication for working as a team. Each team member should be evenly assigned and complete tasks on time, communicate as a team in time when problems are encountered, and find out solutions together.

- If you could do it all over, what would you do differently?

  I would've not used the SnapKit tool kit instead code in the traditional way. It would make it a lot easier for the program to build and interact with other parts of the program better.

B. Jeremiah Kramer

- What technical information did you learn?

  I learned about iOS development using Swift4 in XCode. I developed the containing application for our Gesture Recognition Keyboard extension. This application was developed using Storyboards, which allows developers to create various screens via a graphical user interface. This process made it very easy as a first-time iOS developer. Although I have experience with model-view-controller (MVC) architectures, Storyboards added a new element. I learned how to connect different parts of the application together so they worked together.
- What non-technical information did you learn?

  I learned that there are a lot more background elements that affect large projects than anticipated. Our group had the exciting challenge of starting something fresh from scratch with little to no prior research or brainstorming. Our client called an audible and we switched projects toward the beginning of the Capstone experience. This new project was exciting because it gave us a lot of freedom, but the curse of this was a lot of headache and figuring things out. I learned how to deal with this reality.
- What have you learned about project work?

  I learned that projects take a lot of time and that there can be (usually are) a lot of setbacks. Our group faced quite a few blockers that stood in our way of progress. These blockers required us to change certain tools that we planned to use to complete the project. This had side effects because we found that libraries can be incompatible with versions of software, versions of programming languages, even versions of IDEs (Integrated development environment). Therefore, we had to figure out to change the project in certain areas so everything worked together in harmony.
- What have you learned about project management?

  I learned that it is extremely helpful to plan things out with specific deadlines. Taking on a year long project is one massive task. It became feasible when we split things up into chunks. To add to this, it is best to allow for extra time before the final deadline because there are probably going to be setbacks along the way. Further, communication with the client is paramount regardless if there are setbacks.
- What have you learned about working in teams?

  I learned that teams work best when there is a leader. Our group was fortunate that we had Zac Horine because we all turned to him for help in what we needed to do. I feel like our group worked well and had similar student personalities that meshed together. We all cared about one another as well.
- If you could do it all over, what would you do differently?

  Given the amount of setbacks we experienced, I would have had earlier mini-deadlines for our project. This would have given us a more balanced workload overall. Differently, I would've had added communication with the client during project proposal. Especially when facing with setbacks, interest in projects plays a huge role when investing into the project. I think that negotiating with the client regarding the project would have increased clarity and motivation. With that being said, project experience is vastly important in these situations.

C. Lauren Sunamoto

- What technical information did you learn?

  I learned how to collect and manage data using SensorLog and CreateML. Using SensorLog, we could read out accelerometer data from an iPhone and save it as CSV files for CreateML. CreateML allowed us to create our ML model along with useful metrics to improve it. Also, I learned a bit of Swift (Apple's programming language) and Xcode (IDE for Apple software development). Swift and Xcode were used to develop our keyboard app extension and container app.
- What non-technical information did you learn?

  I learned how to more effectively collaborate with others (i.e. our client and my team members). This involved good communication over email and discord as well as engaged discussion about what changes we should implement to improve our project. I also learned to overcome unforeseen challenges because of strict controls put in place by Apple.
- What have you learned about project work?

  It requires a lot more documentation and careful planning than I anticipated. Also, it involves a lot of careful research needed to find methods that provide results which fit your exact needs.
- What have you learned about project management?

  The main challenge of project management was meeting all of our project goals within our given constraints. I learned that things don't always go as planned and backtracking is a part of reality.
- What have you learned about working in teams?

  Team members are awesome allies that are great resources of knowledge and support. The success of our project heavily relied on the cooperation of each team member and each of our individual efforts.
- If you could do it all over, what would you do differently?

  I would have used CreateML from the very beginning of the project if I knew about it. The biggest challenge our group faced this term was shifting from referencing an open-source C++ machine learning library to creating our own gesture recognition system.

D. Zachary Horine

- What technical information did you learn?

  I learned a great deal about how modeling and pattern recognition work. I got to explore the factors that weigh on a model's accuracy, and although we weren't able to generalize our model, the model was still valid for the individual it was trained on.
- What non-technical information did you learn?

  I learned alot about managing a group of people, and making sure that we all were able to work together in a team and accomplish our goals. I learned that it isn't about you as a leader, but about your team and what we can accomplish together. The job of a leader is to help push others to their best.
- What have you learned about project work?

  When working on a large project, having a plan and following through with it is important. It is okay to make changes to your plan, but those changes should be discussed and documented.

- What have you learned about project management?

  It is important to keep a running log of your progress, what needs to be done, and who is working on it.
- What have you learned about working in teams?

  It is very important that everyone in the team is kept up to date with the work being done. Everyone needs to be included in decisions, and a collaborative approach should be taken to solving problems.
- If you could do it all over, what would you do differently?

  If I had known about some of the tools that were available when we started, such as CreateML, I would have started with those instead of trying to make the GRT library work with our code.

14. Conclusions

Our project explored some very interesting parts of app design, human computer interaction, and machine learning. We learned a great deal about how people use text input, and how we can make it more accessible to different groups of people. In creating our project, we quickly discovered both the things that make iOS so easy to develop for, as well as the limitations of the iOS system. In general, Apple provides plenty of tools to help you design an app that fits within the standardized ”box” that they think it should fit in. This is really useful if you are trying to create menus and views, although it gets in the way as soon as you try to install a third-party library. Apple discourages this, and doesn't allow non-paid developers to use these in the most recent version of iOS. This made it more difficult to design and test our app, as we had to redesign entire portions of our app purely so that it would work with a free account, or pay for the license, purely to use open-source code.

It was very interesting working through the design of a new mode of mobile device input, as we had to think about how the user would use the app, and what felt the most intuitive. We eventually landed on a mode of input that used a button on the keyboard to let the system know when to start and stop capturing a character. This actually proved more user friendly than an algorithm that attempted to solve this problem, as it gave the control to the user, and they were able to more accurately line up the beginning of their gesture with the start of the recording.

Initially we had difficulty determining the proper libraries and training methods to use, and discovered a framework developed by Apple called CoreML. CoreML and its companion app CreateML allowed us to create large sets of sample data and allow the framework to do the heavy lifting in creating the model and drawing inferences. Unfortunately, because of the scope of problem that we were trying to solve, a generic approach like this is not sufficient. The sheer number of model classes and the variability that is introduced with multiple users creating gestures in a 3D space became too much for the generalized model to handle.

15. Future Changes

The piece of our project that needs the most work is the model accuracy, and being able to correctly guess the user's character closer to 100% of the time. The current issue is that our model attempts to categorize the user's input into one of 26 different gestures, which creates a very 'fuzzy' model. The fix for this would be to better normalize the data. Our team did not have the skills to do this, as the problem is very mathematically complex. Our best proposed solution would be to take all of the training data prior to model training and turn each data set into a vector. From here, since all of the characters are two dimensional, the plane at which the gesture is located on can be determined, and the vector can be flattened to this plane. Finally, the resultant 2D vector can be set to a uniform scale and turned back into a set of evenly distributed points, which will make the data more consistent, eliminate angle, speed and size variance. This will make the training data more clear, and result in more distinct gestures. The same method can then be used for user generated gestures before they are compared to the model, which puts them within the same plane, time and size window, where all of the points are normalized. This will make the model's job easier, and produce more accurate results. In order to achieve this, the team that picks this project up would need to have a mathematician and an artificial intelligence focused computer scientist. These two individuals would be crucial in solving this complex problem.

16. Appendix 1: Essential Code Listings

When the keyboard button is pressed, a function is called to start recording accelerometer values and a timer is started. (if the timer expires, the capture is invalidated) When the button is released, the recording and timer are stopped, and the data that was recorded is sent to the CoreML model, which matches the recording against the model that was pre-generated. It then returns the most likely character and displays it on the screen.

```
@objc func firstRowButtonTouchDown(sender: UIButton) {

    self.sequenceTargetX = []
    self.sequenceTargetY = []
    self.sequenceTargetZ = []

    //timer determines length of acceleration data samples
    timer = Timer.scheduledTimer(timeInterval: 1.0, target: self, selector: #selector(self.updateTimer(tm:)), userInfo: nil, repeats: true)
    timer.fire()

    motionManager.startAccelerometerUpdates(to: queue, withHandler: {
        (accelerometerData, error) in
        if let e = error {
            fatalError(e.localizedDescription)
        }
        //Data consists of 3 axis accelerations provided by iOS device
        guard let data = accelerometerData else { return }
        self.sequenceTargetX.append(data.acceleration.x)
        self.sequenceTargetY.append(data.acceleration.y)
        self.sequenceTargetZ.append(data.acceleration.z)
    })
}
```

Fig. 4. Keyboard gesture capture

17. Appendix 3: Code Review and Responses

A. Build

Could you clone from Git and build using the Readme?

Reviewer's comment	Action taken
I was not able to build the project, mostly because I do not have an iOS device, and have no experience with swift.	We recognize and understand that some users cannot build this project. We clearly communicated this during the demo.
I could clone, but I failed to run it with the instruction. I don't know why.	This comment is not helpful because we don't know what error occurred. We found that one major issue with building our project is adding an ssh key to the user's GitHub so that the user has proper permissions for SnapKit.
Team was able to build during demo.	Yes.
As they said, this isn't currently possibly because they're utilizing a third party library and I don't have a developer account. However, I found the instructions very clear to follow, and I believe if I had an Apple developer account they would definitely be sufficient to get the project built.	Thank you.
Project video and demo shows that it builds properly. The readme has good documentation for building and includes screenshots.	Thank you.

B. Legibility

Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?

Reviewer's comment	Action taken
Variable names make sense, and code style is alright. The code could be made far more readable by splitting up the modules into individual files based on the functionality, rather than having one large UI doc with functionality in it.	We understand this criticism but feel that the functions we used are the best way to accomplish this, as choosing to abstract away portions of the code actually makes it less readable and harder to understand as each function relates to a specific key on the keyboard, and they all do different things.
Not very easy to follow, too many packages stuff needs a better explanation. The lighting needs to be improved, such as some lines are extreme long.	We have added more comments so the code is easier to follow. We understand that some experience with Swift4 is required to further understanding.
I would suggest renaming 'First/SecondViewController' to something clearer. The project ostly follows good coding practices, however I would suggest using more comments to clarify what you are doing.	We feel like the name 'firstviewController' is descriptive, as it is the controller for the first view in our containing app. We have also added more comments to clarify what we are doing.
The one thing that did throw me off was naming the view controllers "First/SecondViewController," which gives very little contextual information about what they're actually doing in the program. However, other than that Most of the naming conventions were consistent and the code was rather easy to follow. It definitely followed along the lines of standard Swift conventions I've seen.	We feel like the name 'firstviewController' is descriptive, as it is the controller for the first view in our containing app.
The code is well written and adheres to a common style.	Thank you.

C. Implementation

Is it shorter/easier/faster/cleaner/safer to write functionally equivalent code? Do you see useful abstractions?

Reviewer's comment	Action taken
The code has been abstracted well and makes good use of toolkits.	Thank you.
Not for all the parts, there are some duplicate uses for the swift stuff. I'm little confused about how they used the swift package to recognize the character.	We went over our code and cannot eliminate duplicates. We are not sure which duplicates the reviewer is referring to. We may not have fully explained how the CreateML toolkit functions, but we have added some additional comments to help with understanding.
Would it be possible for you to turn the code blocks in keyboardView() into a function that is repeatedly called with different parameters? Much of the code looks identical aside from the input values.	We understand that these functions do look very similar. However, they are in fact different and necessary. We have addressed this concern but adding comments. More specifically, each one is connected to a button on the keyboard, and they all have primarily unique code, which makes them an impractical candidate for functionalization.
The switch to Apple ML model was a really smart move, and it definitely cut down the code base quite a bit from what it would be. That being said, it definitely may halt the long term extensibility of the project, and I imagine sometime going forward in the future this would have to be migrated to a custom algorithm. Also I saw a ton of repeated code for the "xxxButton" lines in your KeyboardViewController. I would imagine this could be shortened quite a bit if you managed to store the information in an array, but perhaps not. Altogether the code was great though, and besides those minor differences I thought it was pretty succinct.	Thank you, we have discussed this with our client. Thank you, we do understand that some of our code looks really similar, but the functions actually serve their individual purpose and cannot be combined.
The code is organized in a logical way. Code has been properly refactored.	Thank you.

D. Maintainability

Are there unit tests? Should there be? Are the test covering interesting cases? Are they readable?

Reviewer's comment	Action taken
There are no unit tests, though I do see spaces where they could be implemented, checking for internal function responses, etc.	We communicated that our project doesn't warrant unit tests unfortunately. Our algorithm is largely affected but our training data, which we have added. The model has testing data associated with it, which we used to test the validity of our model. The UI is the only part that doesn't have unit tests, which we feel is acceptable.
Yes, there should be, because they have many datasets, the accuracy of classifying a character is worth to test with the functionality.	We have unit tests for our training data. Any other unit tests for our project are not necessary because unit tests on the UI are purely to test superficial things, which can be done manually, and doesn't have enough variability to warrant unit tests.
Unit tests do not appear to be present. They could be useful for testing the app side of the project. I would suggest looking into XCode's unit testing and UI input automation. However, much of the project is based on ML models that are not as easily tested via unit testing.	Since the app is for demoing purposes, we do not have unit tests for it. We have some unit tests for our model but the rest of the code does not make sense nor require unit tests because unit tests on the UI are purely to test superficial things, which can be done manually, and doesn't have enough variability to warrant unit tests.
It's rather hard to write unit tests for this, however basic UI tests can be written that test out, for example, pivoting between the view- controllers. It wouldn't be super helpful, though.	We agree. Since the app is for demoing purposes, we do not have unit tests for it. We have some unit tests for our model but the rest of the code does not make sense nor require unit tests because unit tests on the UI are purely to test superficial things, which can be done manually, and doesn't have enough variability to warrant unit tests.
There is a skeleton for some tests but no unit tests as of yet. It might be a good idea to add these eventually.	Since the app is for demoing purposes, we do not have unit tests for it. We have some unit tests for our model but the rest of the code does not make sense nor require unit tests because unit tests on the UI are purely to test superficial things, which can be done manually, and doesn't have enough variability to warrant unit tests.

E. Requirements

Does the code fulfill the requirements?

Reviewer's comment	Action taken
In many ways yes, the algorithm is the only thing that needs to be tuned. It is not quite up to the 85% mark yet.	We have performed more training for our data so that our algorithm increases in speed and accuracy. We haven't been able to hit 85% yet, but we have addressed this with our client, and will include the reasons why in our report.
Does not fully meet the requirements yet, the demo is failed due to some dependencies stuff.	We have fixed the minor dependency issues.
The code appears to satisfy the requirements pertaining to the app. Other requirements are based on the ML models, and thus require different evaluation.	Thank you.
Yes, besides the minor dependency issues which should be easily fixable the codebase as a whole definitely meets the requirements.	We have addressed the minor dependency issue.
Yes.	Thank you.

F. Other

Are there other things that stand out that can be improved?

Reviewer's comment	Action taken
Nope, good job everyone, If I was a TA, I would give your progress a 3 or a 4.	Thank you. We believe we have addressed the issues with our project.
Make the instructions for set up better.	We have clarified how to build the system and the requirements.
The code appears solid. I would mainly suggest adding more comments and look into condensing your code if possible.	Thank you, we have added more comments for better understanding and clarification.
Perhaps more test cases to better t your machine learning model, along with the minor fixes and usability/readability I suggested. Otherwise I think it looks great, and nothing major comes to mind from me.	We have added more training data to increase accuracy of the model.
Not submitted.	N/A

REFERENCES

[1] S. Aggareal, "Android vs ios: Which mobile platform is best for app development?" techahead. [Online]. Available: https://www.techaheadcorp.com/blog/android-vs-ios/

[2] T. Manifest, "Android vs ios: Which platform to build your app for first?" medium. [Online]. Available: https://medium.com/@the_manifest/android-vs-ios-which-platform-to-build-your-app-for-first-22ea8996abe1

[3] E. Chung, "What language are iOS apps written in?" [Online]. Available: https://www.zerotoappstore.com/what-language-are-ios-apps-written-in.html

[4] "iOS Data Storage Guidelines," 2019. [Online]. Available: https://developer.apple.com/icloud/documentation/data-storage/index.html

[5] Tarun Kumar, "Top 10 ways you should know to store data in Android," Mar. 2018. [Online]. Available: https://www.loginworks.com/blogs/top-10-ways-know-store-data-android/

[6] Danilo Pena, "Ground Truth Versus Bias," Apr. 2018. [Online]. Available: https://towardsdatascience.com/ground-truth-versus-bias-99f68e5e16b

[7] S. Krig, *Computer vision metrics: survey, taxonomy, and analysis*, ser. The expert's voice in computer vision. New York, NY: Apress, 2014, oCLC: 881828978. [Online]. Available: https://www.embedded-vision.com/sites/default/files/apress/computervisionmetrics/chapter7/9781430259299_Ch07.pdf

[8] Roger Grose, Micah K. Johnson, Edward H. Adelson, and Willian T. Freeman, "Ground truth dataset and baseline evaluations for intrinsic image algorithms," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Feb. 2009. [Online]. Available: https://www.cs.toronto.edu/~rgrosse/iccv09-intrinsic.pdf

[9] S. Ray, "Essentials of Machine Learning Algorithms (with Python and R Codes)," 09-Sep-2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

[10] Tea Marasović, Vladan Papić, and Jadranka Marasović, "Motion-Based Gesture Recognition Algorithms for Robot Manipulation," vol. 12, no. 5, p. 51, 2015.

[11] Henry Maltby, "Markov Chains," https://brilliant.org/wiki/markov-chains/, 8-November-2019.

[12] Motion sensors. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion#java/

[13] BASE. [Online]. Available: https://base.xsens.com/hc/en-us/articles/115000224125-RMS-noise-of-accelerometers-and-gyroscopes

[14] L. Critchley. How do haptic sensors work? Library Catalog: www.azom.com Section: Materials Article. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=15700

[15] "Disabilities affecting computer accessibility," Forth ICS. [Online]. Available: https://www.ics.forth.gr/hci/ua-games/disabilities.html

[16] "Motor disabilities: Types of motor disabilities," webaim. [Online]. Available: https://webaim.org/articles/motor/motordisabilities

[17] Z. Macintosh. Brain quickly remembers complex sounds. Library Catalog: www.livescience.com. [Online]. Available: https://www.livescience.com/10670-brain-quickly-remembers-complex-sounds.html

[18] Dev_tones | UI sounds for your app. [Online]. Available: https://rcptones.com/dev_tones/

[19] J. Daciuk. Comparison of construction algorithms for minimal, acyclic, deterministic, finite-state automata from sets of strings. Implementation and Application of Automata Lecture Notes in Computer Science. [Online]. Available: http://web.cs.mun.ca/~harold/Courses/Old/Ling6800.W06/Diary/q3p2qx4lv71m5vew.pdf

[20] "English letter frequency (based on a sample of 40,000 words)," Cornell. [Online]. Available: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

[21] "Digraph frequency (based on a sample of 40,000 words)," Cornell. [Online]. Available: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/digraphs.html

[22] Randy Cassingham, "The Dvorak Keyboard," 2019. [Online]. Available: https://www.dvorak-keyboard.com/

[23] Leah Welborn, "Why QWERTY?" Jun. 2011. [Online]. Available: https://www.mentalfloss.com/article/27938/why-qwerty

[24] Jeffrey S. Leon, "Frequency of Character Pairs in English Language Text," University of Illinois, Tech. Rep., 2008. [Online]. Available: http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/char_freq2.pdf

[25] "100 Most Common English Words (Learn 85% of English in 100 Days)," 2019. [Online]. Available: https://www.rypeapp.com/most-common-english-words/

[26] S. Aggarwal. Android vs iOS - which mobile platform is better in 2020? Library Catalog: www.techaheadcorp.com Section: Mobile Applications. [Online]. Available: https://www.techaheadcorp.com/blog/android-vs-ios/

[27] T. Manifest. Android vs iOS: Which platform to build your app for first? Library Catalog: medium.com. [Online]. Available: https://medium.com/@the_manifest/android-vs-ios-which-platform-to-build-your-app-for-first-22ea8996abe1

[28] Frequency table. [Online]. Available: http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/digraphs.html

[29] "What is SQLite? Top SQLite Features You Should Know," 2019. [Online]. Available: https://www.sqlitetutorial.net/what-is-sqlite/

[30] “Use a SQLite database in a UWP app - Windows UWP applications,” Nov. 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/data-access/sqlite-databases

[31] Obaro Ogbo, “How to store data locally in an Android app,” Sep. 2016. [Online]. Available: https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/

[32] “ApplicationData Class (Windows.Storage) - Windows UWP applications,” May 2019. [Online]. Available: https://docs.microsoft.com/en-us/uwp/api/windows.storage.applicationdata