Technological Feasibility

SmartTalk

October 23, 2020

Sponsor: Dr. Okim Kang

Mentor: Fabio Santos

Team members: Joseph Vargovich (Lead), Andrew Munoz, Kehan Cao, Christian Bito-on, Malik Jones

Table of Contents

1.0 Introduction

SmartTalk is a team of 2020-2021 Northern Arizona University Computer Science capstone students working toward the development of a "Gamified Mobile Pronunciation Tutor for Language Learners." Our team consists of five members:

- ➢ Joseph Vargovich, Project Leader
- ➢ Andrew Munoz, Customer Communicator
- ➢ Malik Jones, Team Recorder
- ➢ Kehan Cao, Release Manager
- ➢ Christian Bito-on, Team Architect

Our mentor for this project is Fabio Santos, who will guide us and assist us if needed in developing a mobile application for language students to practice their pronunciation for their target language. Our sponsor, Dr. Okim Kang, notes that most students learning a foregign language have little to no capacity to focus on pronunciation, and that there is also a wide gap in innovation for mobile devices which provide a gamified platform for language learners to practice their pronunciation. The main problem that arises from lack of focus on pronunciation in many language learning programs is inability to communicate to others in the target language. In essence this means a lower quality of communications. Another issue to consider are the ethics of language learning applications. Some claim to help you speak like a native, however depending on the region of a country, a native speaker's accent and pronunciation can vastly differ. After considering these issues, we have come to consider several features to implement into our project to best tackle this problem for our sponsor:

- ➢ Web Framework
- ➢ Mobile Framework
- ➢ Multi-platform shared database (store your history and can share your progress)
- ➢ Automated Speech Recognition Technology (ASR)
- ➢ Gamification

We hope to use these ideas to create a better, more entertaining learning experience for our users through gamification, and we also hope to assist and encourage people to learn a new language in a different way.

2.0 Technological Challenges

The objective of this mobile app is to provide a convenient way for users to sharpen their pronunciation skills that will, at the same time, use gamified elements to intrinsically motivate them to learn their target language and help them find enjoyment in the activity itself. Considering these key goals, we have identified the following challenges that we will be facing while developing this app:

- ***Gamified*** - Ideally, we will include gamified elements to attract users and increase their interest in learning.
- ***Mobile frame-work*** **-** In theory, our solution will harvest both iOS and Android applications that will be usable in a cross platform supporting both mobile and web frameworks.
- ***Web frame-work*** **-** Ideally, we will have administrators be able to modify and update the data in the database on the webpage.
- ***Multi-platform shared database*** **-** Having a shared database on the mobile app will allow users to access the content in this database, and having a web application will allow the administrator to modify database content to update knowledge.
- ***Automatic Speech Recognition*** **-** ASR technology, which converts spoken words into text, is necessary to ensure the accuracy of speech recognition.

These are the most crucial technological challenges of the development process. Given this, we will need to critically consider alternative approaches and test different solutions to determine what is most ideal for this particular project; once those ideal solutions are determined, we will need to identify the best way for all these components to integrate with one another. By keeping these technological challenges in mind throughout each step of the development process, we will ultimately create a suitable design for the final iteration of the "Gamified Mobile Pronunciation Tutor for Language Learners."

3.0 Technology Analysis

Concerning the technological tools we are thinking about implementing into this project, we have done some research on each component to narrow our original list down to several options. As our primary goal is to develop a product that best fits the needs of our client, our primary criteria for analysis is how well each tool will help the final product meet these needs. Other criteria include ease of use, integration of features, maintainability. Analyzing each technology based on these criteria allows us to identify those which are best for this project. After analyzing each tool based on these criteria, we are going to rate each technology generally on a scale 1-5 to further determine which will be the best option to implement into our project.

The first technological tool we are going to analyze is the mobile application frame-work. This is the core of our project. Other technological tools will need to satisfy this primary one, and so analyzing this primary tool first will greatly assist the evaluation of the other tools.

3.1 Mobile Framework

The Issue

One of the main interests of the client is to have an effective mobile application that is cross-platform and interfaceable with a website dashboard. We require an adaptable, cross-platform framework that is easy to develop on and is able to utilize and communicate with a cloud-hosted database and a website. A cross-platform mobile application will allow a wide variety of users to take advantage of our app and therefore provide more audio analysis samples for Dr. Kang's team and linguistics researchers. The mobile app must be easy to use and aesthetically pleasing to be viable for the final product.

Desired Characteristics

The ideal mobile application framework will perform several core functions. First, it will be easy to use and learn for members that are not familiar with it. Secondly, the framework will be able to produce professional user interfaces (UIs) that allow users to perform key functions of the app quickly and without confusion. Thirdly, the framework will have plenty of built-in libraries and tools that allow us to effectively integrate the mobile app with the other parts of our final system. This includes being able to easily query and modify a cloud-based database in order to reflect information gathered to user accounts stored remotely. This database will allow the web and mobile applications to effectively communicate.

Alternatives

There are several popular choices for cross-platform app development. It is important to explore how voice recognition and integration with ASR will work with these frameworks. Three top options are listed below.

Option 1: Flutter

Flutter is an app development framework developed by Google. Flutter has a similar UI design model to HTML, so it is easy to learn and customize UI features. Flutter is lightweight and highly portable, so we can develop with Flutter using a variety of IDEs and emulators. The core programming language behind Flutter is Dart. Dart is a C-like programming language that is designed to be easy for developers to learn, especially if they already have experience with languages such as C, Java, and Python. Thus, we should be able to learn and use Dart without too much effort.

Option 2: Qt (C++)

Qt is a mobile application development framework that also translates to many different types of programs and user interfaces. The backend and development is powered by C++. Qt may be our best option if we need to do difficult parsing and analysis of data returned to us by the ASR analysis. However, C++ is not a commonly taught language at NAU, so we must consider the

learning curve for the language so all of our members can contribute effectively. Several C++ libraries, such as Asio, can be combined with Qt to easily create user interfaces for applications. Some sources talking about Qt say that it can be difficult to test.

Option 3: Ionic

Ionic is a JavaScript, HTML5/CSS based mobile development framework developed by Drifty. Several of us are already familiar with these languages, so the transition should be relatively painless. This library has been used quite often on previous capstone projects, so we will have plenty of related work and examples to look at and learn from.

(1-5) rating	***Flutter***	***Ionic***	***Qt***
Ease of use and learnability	4	4	3
Reasons	Flutter is modeled off of trees of widgets that are similar to HTML and markup. Dart is a C-like language that was designed to be easy to learn with prior programming experience. Several members have used and learned Flutter.	JavaScript and HTML are well known tools that we have all used here at NAU. These comprise most of Ionic.	C++ can be a difficult language to learn initially as it deviates from common languages like Java and Python quite significantly.
Core Libraries and UI Design	4	4	3
Reasons	There are gamification libraries available and ready to use for free. Plenty of good tutorials for UI design. Easy to learn. Good integration and emulator support in VSCode and Android Studio. Dart is not a familiar language for most members, so learning must be done.	Good support and documentation. Free and open source framework. Some gamification is available, but not as much as Flutter. Lack of clear cut gamification libraries with less tutorials in than Flutter in this regard.	Qt is a mix of proprietary and open source options. Not necessary for this project as we should be able to have free, open source software to use. Little to no open source gamification is available.

Integration with other features of the project	5	4	2
Reasons	Decent tutorials for database and web app integration. Integrates well with Google Firebase, our top database choice. Flutter is also a web framework, so we can share knowledge to complete these two stages of the project.	Usable documentation of integration with a Firebase backend, but may be more issue prone than Flutter as it is not a direct Google product. Not many straightforward tutorials available.	Few easily accessible and well documented tutorials for integrating with Google Firebase's API. Much of Qt's tutorials are poorly
Total	13	12	8
Summary	Flutter is a relatively easy to learn and use mobile development framework that has plugins to integrate with our ASR technology.	Ionic is a well-documented framework that will be a great second choice if Flutter does not work out.	Qt is a more difficult to use and harder to learn framework for mobile development. Qt has less open-source and accessible options to our team.

Chosen Approach

Currently we are going forward with Flutter. Flutter is an easy to learn and use mobile application framework that will facilitate cross platform testing and development. Its associated programming language, Dart, is campaigned as an easy to learn programming language that is similar to popular languages such as C and Java. There will be a learning curve for the team members who have not used Flutter/Dart before, but there is a variety of good starting tutorials for beginners. These tutorials will be a vital part of our preliminary development of a functioning demo program. Ionic is another very tempting second option that we could pivot to if Flutter does not prove to be feasible.

Proving Feasibility

Proving the feasibility or mobile development frameworks will be multi-faceted. Firstly, we will test how convenient it is to develop user interfaces and basic logic with each of the three proposed frameworks. This will be done by creating a preliminary wireframe and implementing it across the three frameworks. We will keep track of how difficult it is to produce professional UIs, as well as the amount of detail we can specify within each of the three options. This initial wireframe will be simple, and will serve as a simple “hello world” like example.

The next testing step will involve creating a more involved demo that can query a remote database and push data to it when needed. Flutter may be easier to integrate with other Google products such as Google Firebase. This is something that will determine which mobile framework we chose. Thus, testing at this stage will be a combined effort between several facets of our project.

3.2 Web Framework

The Issue

In order to ensure the update of the app content, the client requires us to make a corresponding web program based on the mobile app. The administrator of the mobile app can log in to this webpage, and at the same time modify and update the data in the database on the webpage. In order to achieve this effect, we need a powerful front-end framework that can apply multiple libraries.

Alternatives

There are many front-end framework choices, but we need a framework that can apply multiple libraries, and this framework can use several cross-platform databases of our choice. In addition, it should also have the ability to use ASR technology, so that it will be more convenient for clients to maintain the program in the future.

Option1: Flutter

Flutter is a multi-platform development framework developed by Google. The reason for using Flutter in a web framework is that it is connected to a mobile development framework. Our team's preferred mobile development framework is Flutter. If our web development framework also chooses Flutter, it can reduce our workload and reduce maintenance costs. It can bring great convenience to use with mobile phone programs.

Option 2: Angular

AngularJS was born in 2009, created by Misko Hevery and others, and later acquired by Google. It is an excellent front-end JS framework that has been used in many Google products. AngularJS has many features, the core ones are: Model-View-ViewMode (MVVM), modularization, automatic two-way data binding, semantic labeling, dependency injection, etc.

Option 3: React

React is a JAVASCRIPT library for building user interfaces. React is mainly used to build UI. Many people think that React is the V (view) in Model View Controller(MVC). React originated from Facebook's internal project to build an Instagram website, and it was open sourced in May 2013. React has high performance, code logic is very simple, more and more people have begun to pay attention to and use it.

Option 4: Vue

Vue is a progressive framework for building user interfaces. Unlike other large frameworks, Vue is designed to be applied layer by layer from the bottom up. Vue's core library only focuses on the view layer, which is not only easy to use, but also easy to integrate with third-party libraries or existing projects. On the other hand, when combined with modern tool chains and various supporting libraries, Vue can also provide drivers for complex single-page applications.

(1-5) rating	*Flutter*	*Angular*	*React*	*Vue*
Ease of use and learnability	5	4	4	4
Reasons	Reduce our workload and reduce maintenance costs.bring great convenience to use with mobile phone programs.	Have a longer history of using Google Firebase. You can refer to more existing projects to improve the efficiency of development.	React can design a more user-friendly UI, improve customer experience, and simplify customer maintenance operations in the future.	Vue has advantages in the use of third-party libraries, and this feature may play a role in the application of ASR technology on the web side.
Core Libraries and UI Design	2+2	2+2	1+3	3+1
Reasons	Flutter can call many libraries, including how to use Firebase and ASR technology. These two functions are the key to our project. In addition, as a cross-platform development framework, Flutter can more intuitively see the results of writing code on different platforms, which is of great help to our UI design.	Angular is also a product of Google, and it has achieved good results with the use of Firebase. In addition, Angular can call many components to design the UI, we can refer to the existing UI design.	React is mainly used to build UI.	Vue is easy to integrate with third-party libraries or existing projects.
Total	9	8	8	8

Chosen Approach

Now our first choice is Flutter. Because Flutter is a multi-platform development framework, it can meet our web development needs. At the same time, if we choose Flutter in both mobile and web development, then we can save a lot of time to learn a new front-end framework. In addition, because some members of the group have learned Flutter, our work can start faster.

Proving Feasibility

In terms of proving feasibility, the first thing to consider is whether the front-end framework we choose can use the multi-platform database well, because the main function of the web-side program is to update the data in the database to serve the mobile-side program. We will test the performance of different front-end frameworks when using Google Firebase. We will create some data in Google Firebase, and then use different front-end frameworks to call and modify these data, and finally we will select a good applicable front-end framework.

3.3 Multi-platform Database

The Issue

In order for this application to work to its fullest potential we need to be able to have a database to store client information and communicate with our app. This information can consist of user information on lessons completed or instructor information that can help them lead a class.This app is very database intensive, which means we have to dwell carefully on which types of database we want to use.

Alternatives

While we can use virtually any database for our work, what we use for the database will be somewhat reliant on which mobile framework we decide to use for the app. This is due to

some databases having more integration features for different frameworks. Below are some databases which can serve as databases for all mobile frameworks.

Option 1: Google Firebase

Google Firebase is an online database that can be used with any mobile framework. While it is somewhat simple to use it does keep us from having full and complete customization of our database. However this is not to say that Google Firebase is any less powerful than any other database. Another benefit that Google Firebase provides is aid in uploading and monitoring our mobile application. This comes in the form of user analytics. Google Firebase comes in especially handy when it becomes paired with Google's mobile framework known as Flutter. This allows for better integration of features such as messaging between users. Overall Google Firebase is a simple to use yet powerful tool becoming increasingly popular in the database community.

Option 2: SQLite

SQLite is a powerful database that allows for a fully customizable database that can be stored both locally and online. Now while this database is extremely powerful and extremely customizable it will be tough to implement if there is not a significant amount of dedication put into learning this database. In order to achieve the fullest potential of this application it requires a good amount of database experience.

Option 3: Amazon Web Services

Similar to Google Firebase but hosted by Amazon. The one drawback to AWS is the complexity of the billing system and for companies trying to use more information from the database. It is slightly more expensive than Firebase but offers a nice alternative.

(1-5) rating	*Google Firebase*	*SQLite*	*AWS*
Simplicity	5	3	4
Reasons	Made for simplicity and allows for users to integrate features seamlessly	Needs more knowledge of databases but less is done for database management	Almost the same as Google Firebase but allows for a little more flexibility, increasing complications that may arise
Customizability	3	5	4
Reasons	Provides simple frameworks and ideas for the users to be able to implement into their app. Also supplies tons of API's to use.	Allows for full use of customizability, since we hand code everything that we want to do	Same as firebase without the various API's
Maintainability	5	3	4.5
Reasons	Features are updated by Google and has a simple UI for the client to continue to use even after completion of the capstone project	Beyond this capstone project it may be harder to maintain features after the project has been completed. Will need to have someone to maintain the database	Same as Google Firebase. May be a little more complex and the UI may be a tad more confusing
Total	13	11	12.5

Chosen Approach

Overall Google Firebase offers something that can integrate seamlessly into our app and sets our client up for a good way to maintain the database of the app even outside of the capstone

project. Google Firebase's app also plays a huge role in allowing for the client to monitor how the app is doing on the play store and receive user analytics.

Proving Feasibility

In order to prove feasibility for this database we need to create a sample of how we are going to handle users through the database and showcase some of the technology that makes our choice of Google Firebase appear worth it. We will need to create a way to allow users to store their information in order to login as well as have a way to check on their progress through the app. To do this we can create a base of sample users with sample usernames and passwords. We also need to find a simple way to display user task information such as monitoring where they are in a particular activity and being able to recall that information.

3.4 ASR Technology

The Issue

The amount and type of information provided by ASR technology is crucial to the development of our application. It forms the basis of our language learning system because the rest of our system revolves around the information gathered by our chosen technology. If we select an ASR library that only provides shallow information such as simple speech-to-text, it limits the potential of the other aspects of the application, as we would only have text-only results to go off of. Likewise, more sophisticated ASR technologies would provide us with more potential through more detailed information. This information can contain data such as the individual sounds the system detects versus what the system expects. However, these powerful ASR technologies often require more computer resources; sometimes more than a high-end phone can provide. Finding a balance between the level of gathered information and amount of resources consumed to do so is vital to a successful language learning application.

Desired Characteristics

The ideal ASR technology we desire is a library that can be run reasonably well on a mobile phone without consuming too much processing and battery power while also gathering a baseline of information. When gathering information, the technology should ideally collect a number of things when given an audio input: a list of phonemes (a distinct unit of sound) that the system hears, for every phoneme heard, an expected phoneme (inputted manually or through a reference file), and a rendition of what was heard by the system (something like speech-to-text).

Alternatives

There are a number of different paths we can take when selecting an ASR technology. We can lean towards more information, sacrificing more resources than what is ideal; or we can focus more on resource-friendly technologies, giving up potential facets of our application for the ability to have more phones be able to run our app.

Option 1 & 2: Kaldi

A recommendation from a member of Dr. Kang's team, Kevin Hirschi, Kaldi is a toolkit for speech recognition. Originally written in C++ and developed by Daniel Povey, Kaldi has now expanded its utility into other languages such as plain C, Java, Python in the form of APIs written by other developers to provide each platform a way to interface with Kaldi. Kaldi does a number of positives: it gives us what it hears as well as expected sounds in addition to start and end times for each phoneme. It can also be trained for an individual if need be. However, it is very processor intensive if we choose to have every piece of information listed above. There are two different types of Kaldi:

Option 1: Server-side Kaldi

Server-side Kaldi allows for users to have their audio input be analyzed by a server running Kaldi. This version of Kaldi offloads the high resource consumption to the server, allowing the mobile application to use up less resources. However, this requires the user to be online when using the application to send their input to the server and to receive the results. While nowadays devices have become more consistently online through the means of wifi or data, there are still possibilities that users are not able to use the application due to weak or no

internet. Overall, while using Kaldi on a server provides us with the information we need while not overusing a phone's resources, the problem shifts towards the type of server we can acquire and if it can handle a certain number of requests.

Option 2: On-device Kaldi

On-device Kaldi is the more straightforward version of Kaldi; it runs Kaldi on your phone, through some sort of interface API. This option accrues all the resource issues that Kaldi has, and with the limited resources a mobile device has, only allows for either a limited version of information gathering or full-power capabilities at the cost of extreme battery usage. However, being able to utilize Kaldi on mobile allows the user to use the application even when not connected to the internet. This also eliminates the difficulties of setting up a server to handle Kaldi requests, keeping everything on the phone.

Option 3: Pocketsphinx through CMUSphinx

CMUSphinx is a similar technology to Kaldi developed by Carnegie Mellon University. Much like Kaldi, CMUSphinx is available for both server-side and on-device use. CMUSphinx's libraries are simpler and easier to use in comparison to Kaldi's at the cost of less versatility and accuracy in some situations. While CMUSphinx has less potential than Kaldi on the server-side, Pocketsphinx, CMUSphinx's version for mobile, could be a good alternative for mobile Kaldi, sacrificing some information for less resource consumption. However, Pocketsphinx does not have well-developed end-user tools when compared to Kaldi's vast amount of developer-created interfaces.

(1-5) rating	***Server Kaldi***	***Mobile Kaldi***	***Server CMUSphinx***	***Pocketsphinx***
Information Gathering	5	5	4	4
Reasons	Provides analysis for nearly every aspect of language, from general speech to individual sounds.	Same as Server Kaldi	Provides the same information as Kaldi, but does so at a lower accuracy rate.	Same as CMU Sphinx
Ease-of-Use	1 + 1	2+ 1	3	4
Reasons (Additional +1 for Kaldi due to having a member with previous experience)	The complete library is overwhelming for first-time users and requires the setup of a server	Overwhelming library but no need to setup a server	Simplified library but also requires the setup of a server.	Simplified library without the need for server setup. Also already has demos available for android and IOS
Other Additionals	5	3	5	4
Reasons	Has flexibility in training the system or simply using preset information	Has the same capabilities as Kaldi, but memory requirement can be costly	Has the same capabilities as Kaldi, however it uses different models not as readily available.	Has the same capabilities as CMUSphinx, and is optimized for mobile compared to Kaldi
Total	12	11	12	12

Chosen Approach

Currently, we are leaning towards Pocketsphinx as our first choice, followed by a server implementation of Kaldi as our second choice. While Pocketsphinx has a lower accuracy rate compared to Kaldi, it makes up for it with easier to use functionality as well as having two demos for Android and IOS readily available from the developers. While this does not necessarily make Pocketsphinx an easy integration into Flutter, our first choice for mobile development, the platform's ability to use android or IOS-specific code means that there is less difficulty compared to having to integrate Kaldi, which does not have as much IOS-related compatibility.

Our second choice, server-side Kaldi, has some advantages over Pocketsphinx, chiefly having a higher accuracy rate when identifying words. However, there are some glaring disadvantages that put it behind Pocketsphinx. Creating a client-server system for this application would be best accuracy-wise, but it is far more difficult to set up than an on-device version. Furthermore, there is the issue of maintenance and scalability. At this point in time, the amount of people utilizing this application does not warrant the cost of keeping a dedicated server up and running. Overall, server-side Kaldi will remain our second choice unless we either encounter an unfixable problem with Pocketsphinx, or the scale of our project grows bigger.

Proving Feasibility

To prove the feasibility of each of these options, we need to look at two different metrics. We have a measurable metric in performance, where we can test each option on how fast the analysis arrives as well as how accurate it is. To do this, Dr. Kang's team can help identify each analysis' level of correctness by comparing the output to an actual transcription. On the other hand, the unmeasurable metric would be how the team feels using each of these technologies. While comfort and ease-of-use cannot be easily measured, there are thresholds that should be met, such as if the team can understand the overlying structure.

3.5 Gamification

The Issue

One of the main issues to be addressed in the software mobile app is the lack of gamification. In order to make this app more enjoyable to users, and therefore to motivate the users to keep striving for a goal in an enjoyable manner, we are going to need to implement some gamified elements into the application. This will also need to be on the service level for users to interact with the mobile application, which is important to keep the users entertained.

Alternatives

The possible approaches to implementing gamification are generally based on either intrinsic motivation or extrinsic motivation. One of our challenges is to decide which type of motivation we want to implement into the project, as well which program would best help users get experience of learning a new language. In designing a game all the concepts are the same: there are goals, rules, challenges, and feedback. For this project in particular however, we need to identify a particular behaviour that we can encourage users to perform.

Motivation Techniques

Research on how gamers are motivated reveals two basic techniques: extrinsic and intrinsic motivation. Extrinsic motivation focuses on using rewards to encourage users to achieve a desired outcome or to avoid punishment. This works to some extent to motivate people in some contexts, but the value of rewards diminish over time, and bigger and better reward systems are needed to keep the users motivated. Once a reward is removed, the user will also be less likely to continue the activity.

On the other hand, intrinsic motivation is a technique to motivate users through enjoyment of the activity rather than through gaining a reward or avoiding punishment. One of the techniques that supports intrinsic motivation is Self Determination Theory. Through our research we have found that there are three psychological needs to proceed in this theory and

promote these game mechanics which are autonomy, competence, and relatedness. Autonomy is the choices that people make and why they make them. For instance, in a video game the player chooses to take on activity because they are interested rather than doing it because of a reward or punishment. Competence is the ability to be challenged appropriately; this generally happens when we are given a challenge that matches our skill level. Lastly, relatedness is our connection and our development of close personal relationships.

Another theory that promotes intrinsic motivation is the theory of flow. First introduced by Mihaly Csikszentmihalyi in 1990, flow refers to a state of total attention that brings satisfaction. It occurs when you are concentrating so intently on an activity that you are completely involved in that activity (add youtube video citation; Nakamura and Csikszentmihalyi, 2009). Incorporating this idea into the games increases motivation and engagement by providing:

- ➔ A clear goal that you are working toward
- ➔ Clear progress towards completing that goal
- ➔ Clear & immediate feedback to tell you how you are doing
- ➔ A balance of challenge and skill

Game Engines

We are leaning towards Flutter for our mobile application frame-work, specifically using the gaming libraries within Flutter such as Flame, SpriteWidget and Candy. We will be evaluating each of these gaming libraries for ease of use, functionality and performance.

The first game engine we are going to look into is Flame. Flame engine is a minimalist flutter game engine which simplifies the code you need to build projects. This game engine provides a few utilities such as images, audio and a component/object system. The developers that have created this game engine's main goal is to create solutions for common problems in every game development through flutter.

Next, SpriteWidget game engine is a toolkit for building more complex and high performance games. This game engine has the same utilities as compared to Flame but the developers main goal of producing this engine was mainly focused on the high performance

through our research. Although this game has very stable performance, complete functionality it seems has a lack of easy development.

The Candy engine is a lightweight game engine which doesn't take up that much space on your desktop. Also this engine is very easy to use for developing games and having stable interaction throughout the game. The developers of creating this game engine main goal was create features for easy development, stable performance and complete functionality. Although most of these desires are met the only flaw is this game engine within flutter is that there has been some issues of developers experiencing some memory leakage.

(1-5) rating	***Flame***	***The Candy***	***SpriteWidget***
Ease of use	5	5	4
Reasons	Easy to develop 2D flutte games.	Easy to develop 2D flutter games	Somewhat easy to develop 2D flutter games with high performance animations.
Integration features	5	5	5
Reasons	Flutter gamification library	Flutter gamification library	Flutter gamification library
Functionality	5	4	5
Reasons	Complete functionality and a solution for common problems in every game development.	Slight stable the reason being of some memory leakage, but besides that the program is functions properly	Complete functionality and high performance game.
Total	15	14	14

Chosen Approach

During the research process, we have determined that the best approach would be intrinsic motivation rather than extrinsic because of the duration of encouragement. By implementing this into our project, the user will be able to learn a new language not because of the rewards or badges that they will be achieving, but out of the enjoyment of the activity. Implementing this intrinsic motivation into our project will hopefully have the added effect of the user experiencing a theory of flow, which will make them spend more time on the content at hand and sharpen their skills.

As for choosing a game engine from all our options we are leaning towards flame. Out of all the options Flame seems to be the more stable game engine for easy development, stable performance and exceptional performance. While looking more deeply into this game engine there are many tutorials ,templates and many intriguing utilis to learn how to use this software with ease to make a gamified mobile application engangin for users to play.

Proving Feasibility

In order to best address the challenge raised by this particular project, we will be creating a technology demo towards the end of the semester to demonstrate the implementation of the key elements for intrinsic motivation for users to stay moviatiated and increase enjoyment of doing this activity of learning a new language. While creating this gamified app we are leaning towards using the Flame engine by these three aspects: easy development, stable functionality and stable performance. Also this game engine integrates with flutter and this seems like a solid base platform on developing games.

4.0 Technology Integration

As Figure 4.1 shows, our front-end mobile and web application will be developed through Flutter for cross platform on mobile applications. By having this technological component integrated into our project all users will be able to use this application, and this will increase the total size of users that will be using the application rather than limiting it to only one specific platform. Also, the gamification aspect will be on the service level to help the user find this application enjoyable and motivate them to keep learning.

The back-end development will be storing user information into a cloud database called Firebase. Having this component will provide systematic capabilities to evaluate how users are using mobile applications; if a user is having any issues with the application the administrator/staff can go examine the issue on the web application and override the user data if necessary. Firebase will be integrated so there will be no issues connecting a web and mobile abilication for front-end and back-end abilities. ASR technology will be used to receive data from the user and make sure the accuracy of the pronunciation is correct. Allowing feedback for the user is the key for the user to self-assess their progress with their target language.

Below Figure 4.1 is a system diagram that is displaying how we are planning to have all the technological components that we have chosen to interact with one another.

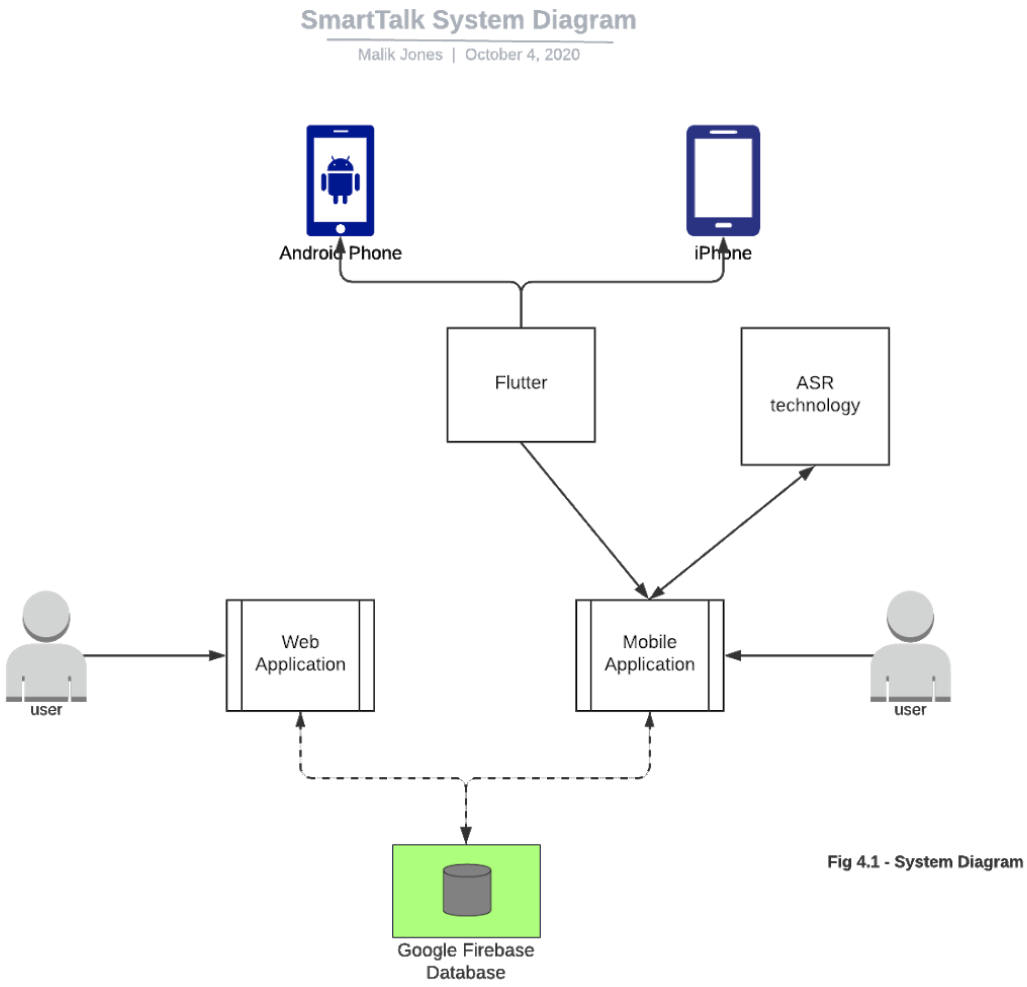

Fig 4.1 - System Diagram

Conclusion

The overall significance of this document is to discuss the technological feasibility of the "Gamified Mobile Pronunciation Tutor for Language Learners" and address specific technical challenges that we will face during the development process. The main problem that arises from lack of focus on pronunciation in many language learning programs is an inability to successfully communicate to others in the target language. In essence this means a lower quality of communications. The second issue concerns itself with ethics, as pronunciation plays a large part in learning the language, and different pronunciations can be attributed to different native accents. With these concerns in mind, SmartTalk's primary goal for developing this product is to create a convenient way to perform second language assessment through a gamified mobile app that encourages language learners to sharpen their pronunciation skills through gamification.

SmartTalk is thrilled to be a part of the project working on "Gamified Mobile Pronunciation Tutor for Language Learners" with our sponsor and her team. While developing this product may take time, we will work as a team to respond to any new challenges we encounter during development, and we are confident that we will find a reliable solution to each one.

Below is a chart of our research that we believe will lead to a successful, functioning product for the objective for this project. As we move forward with this project we will continue to look out for potentially better solutions to fit the needs of our sponsor.

Technical Challenge	***Proposed Solution***	***Confidence level (1-10)***
Web frame-work	Flutter	8
Mobile frame-work	Flutter	8
Multi-platform Database	Google Firebase	8
ASR Technology	Pocketsphinx	7
Gamification	Implement Flame to gamify the mobile app. Intertwine intrinsic and extrinsic to engage the user the in the activity.	8

Citations:

"5 Best Flutter Game Engine." *Dunebook*, 7 Feb. 2020, www.dunebook.com/best-flutter-game-engine/.

Authors, Various. "Your First Ionic App: Angular - Ionic Documentation." *Ionic Docs*, Ionic, Apr. 2020, ionicframework.com/docs/angular/your-first-app.

"Asio C++ Library ." *Asio C++ Library*, think-async.com/Asio/.

bakis , Genel. "Smartphone Apps That Gamify Language Learning." *Telc - Smartphone Apps That Gamify Language Learning*, Nov. 2018, www.telc.net/tr/hakkimizda/guencel/detail/smartphone-apps-that-gamify-language-learning-1.html.

"Building Games Using Flame." Performance by Luan Nico, and Renan C Araujo, *Building Games Using Flame - Luan Nico & Renan C. Araújo | Flutter Europe*, Flutter Europe , 8 Apr. 2020, www.youtube.com/watch?v=sFpjEH-ok2s.

"Cereal - A C++11 Library for Serialization." *Cereal Docs - Main*, uscilab.github.io/cereal/.

Flores, Jorge F. "Using Gamification to Enhance Second Language Learning ." *Digital Education* , 2015, files.eric.ed.gov/fulltext/EJ1080742.pdf .

Gilakjani, Abbas Pourhossein. "Why Is Pronunciation So Difficult to Learn? ." *English Language Teaching* , 3 Nov. 2011, files.eric.ed.gov/fulltext/EJ1080742.pdf .

Tech, Alibaba. "Candy for You: Design of A Flutter Interaction Engine." *Medium*, Medium, 28 Feb. 2020, alibabatech.medium.com/candy-for-you-design-of-a-flutter-interaction-engine-1d7f1196130b.

Top 4 Gamification Techniques , Gamify, 19 Apr. 2020, www.youtube.com/watch?v=iX3zQo_TCM0 .