BOSTON UNIVERSITY

Bi-lingual Video Conferencing

ECE601 Placement Exam

August 30, 2019

Kristi Perreault
College of Engineering
MS in Electrical & Computer Engineering

Table of Contents

Abstract

The bi-lingual video conferencing application is a cloud service developed to enable real-time language translation through video chat. Through extensive analysis of cloud providers and video conferencing applications, as well as user feedback in the form of personas, an acceptable minimal viable product was developed, leveraging basic video conferencing capabilities and AWS cloud service offerings. An expansive and thorough testing suite ensured a simplistic and cohesive experience for the user, and by addressing challenges such as real-time translation and pricing models, the customer was continuously remaining the center of attention. In addition, new product releases containing features such as support for multiple languages, audio playback, and user profile management were included in the product roadmap for future iterations. By utilizing an agile model, the developing engineer was effective, timely, and customer-centric throughout every stage of the application's development. This report will step through the engineering and product design process for the creation of a bi-lingual video conferencing application in an agile environment.

Introduction

Technology has and continues to impact every part of society. One area in particular that has transformed extensively over time is communication. People across the globe are more accessible than ever before, with just a quick phone call, text, or video chat. As people are expanding, exploring, and working with other areas of the world, one shortcoming that is still abundantly clear is the language barrier. Current text translators have begun to address this issue, but the world is still lacking in real-time language translation. Working remotely, long-distance relationships, and frequent traveling has become the norm, and although there is the luxury of text translation, there is no adequate substitute for face-to-face communication.

Bilingual video conferencing is a solution to this problem. This report proposes a software application capable of video conferencing with real-time audio and text translation, allowing users to communicate with anyone, regardless of what language they are speaking. The goal of this application is to support translation between any spoken language, with little to no latency. It supports basic video conferencing features, such as quality video and visual and audio control, with the added benefit of real-time audio and text translation through the cloud. Amazon S3 hosts text and audio files, Amazon Translate's neural network uses machine learning to train models and translate over sixty different languages, and Amazon Polly converts translated text to speech, with multiple different kinds of human-like readout. With this product, the world can be brought even closer together, one language at a time.

Agile Tools & Methodologies

Th agile concepts were modified slightly in this project due to the short timeframe and having only one engineer acting as Product Owner, Scrum Master, and an individual contributor. At the start of product development, a combined backlog grooming and sprint planning meeting was held to create and prioritize stories. The planning and tracking took place in a Kanban Board and Issue Log in GitHub. Stories, or “issues” in GitHub, were created from the provided guidelines and brought in to the project space. The Kanban Board was created with a “To Do”, an “In Progress”, and a “Done” column to keep the tracking simple with only one person and one week. All stories were brought in to the “To Do” column as they were created and were ordered based on priority – the top item being research, the bottom item being presentation practice. Stories were dragged across the board appropriately, as they were being worked.

The project was worked across multiple days, so a one-person “daily standup” was timeboxed for five minutes at the start of each working session, in order to be mindful of time. The Kanban Board was pulled up, and the work that had been completed previously was evaluated, the work in progress was reviewed, and the work planned for the day was outlined. Since this is agile, there were occasions where, in some cases, work priority was reevaluated, and the Kanban Board was adjusted as necessary. The demo phase of the agile process was handled in a 15-minute presentation to the stakeholders, in this case the professor, peers, and any relevant faculty members. As is common at the end of most sprints, a retrospective was held to determine what went well, what did not go well, and what could be done differently in the future. More information on this can be found in the Retrospective portion at the end of this report.

Option Analysis

The bi-lingual video conferencing service is a cloud service, meaning it will utilize services in the cloud to store files and data, run processes, log events, and so on. There are four big players in the cloud technology space today: Microsoft, Google, IBM, and Amazon. These four cloud providers all offer similar services for the application to use, so an option analysis was run to determine which provider would be the best fit. For this application, services to store files and data, transcribe text to audio, and translate languages were investigated and compared. It is important to note that, for the video conferencing piece of the application, each of these services also offer a video chat application. These video applications, along with a few others including Zoom and GoToMeeting, were also evaluated as competitors for this application.

Microsoft’s cloud platform, Azure, has a multitude of service offerings, and even guarantees a price match against Amazon [1]. Many of the service offerings for databases and storage are similar to most other cloud provider offerings, however, the machine translation with Microsoft Translator was one of the most impressive and was very well documented. Microsoft implements both Statistical Machine Translation (SMT) as well as Neural Machine Translation (NMT) for language translation and has been working with text and speech since 2007, meaning that the API data they have for machine learning is quite extensive. Unfortunately, Microsoft has been using their translation technology for live bi-lingual video conferencing in Skype since 2014 [2], and thus would make this application a direct competitor, so it was decided that Azure would not be used.

The second cloud provider option explored was the Google Cloud Platform. Google is widely known as one of the first to the language translation scene with Google Translate, which now features machine learning translation as well as a translation API. In addition, Google offers a Text-to-Speech application and DialogFlow to allow for “natural conversational experiences” [3]. This cloud provider also offers a video chat application, Google Hangouts, which integrates with some of their other cloud offerings. The storage offering, however, is not as robust as some of the other providers, and the developer documentation is not very intuitive [4]. For these reasons, the Google Cloud Platform was not selected for this application.

IBM offers another impressive cloud option known as Watson. Unlike the other cloud providers, Watson is almost solely focused on machine learning and AI, making its service offerings for the bi-lingual piece of the application very appealing. This cloud provider offers a language translator, a natural language classifier, and a natural language understanding, which will actually analyze text and identify key words, phrases, and relationships in sentence structure [5]. In addition, Watson offers text-to-speech as well as speech-to-text. Although tempting, the decision was made to not move forward with IBM’s offering, as it lacked in other areas such as file storage, cloud computing, and pricing.

The final option evaluated was Amazon Web Services (AWS). AWS was the first widely adopted cloud provider, and the one that was selected for this application for a number of reasons. The other cloud providers offer many of the same kinds of services, but not all, and definitely not as expansive as AWS’s offerings. Amazon Transcribe, Translate, and Polly provide language translation and text-to-speech/speech-to-text. In addition, AWS RDS is available for databases, S3 buckets are available for file storage, Lambdas and step functions allow for running event-driven code in the cloud, and CloudWatch, CloudTrail, and SNS are capable of logging and providing the engineers and users with notifications [6]. It is flexible and offers a “pay as you go” model, with a free tier, relatively low cost, and bundled options available.

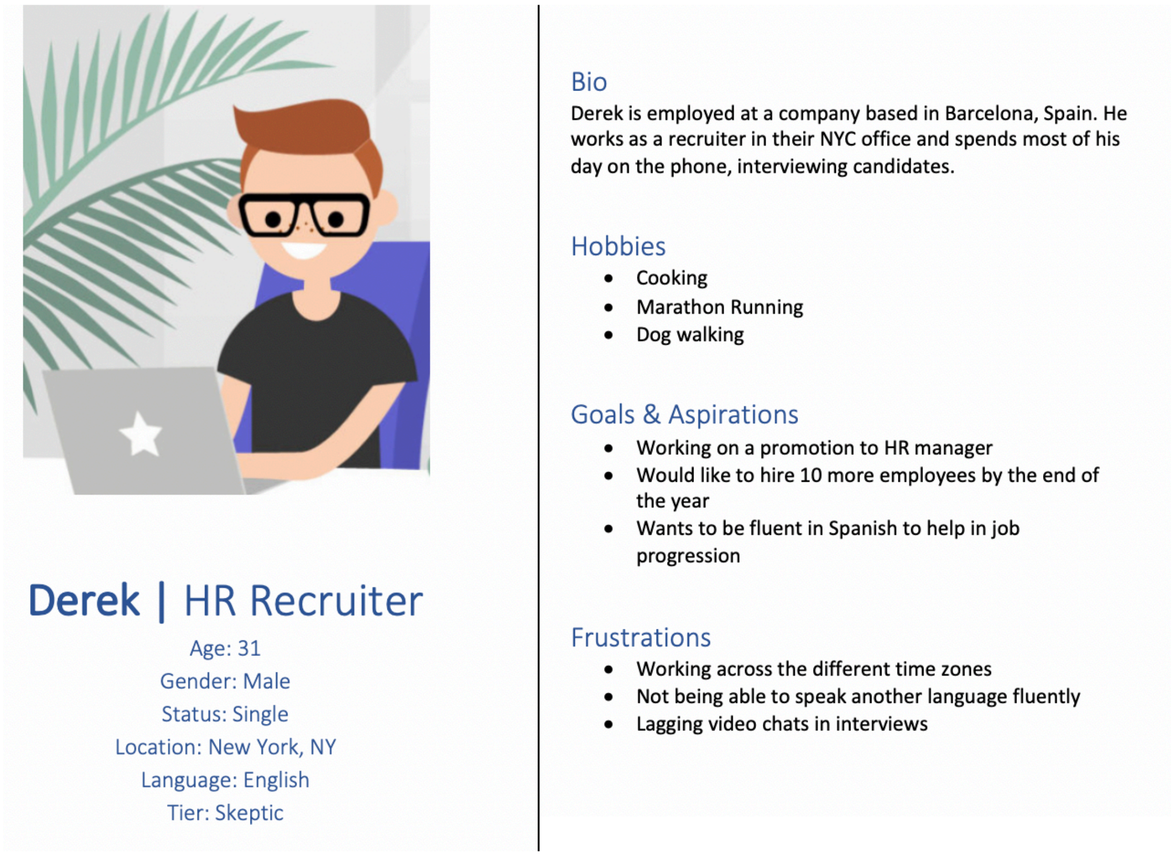
Each of the cloud provider options were impressive in their own way, and all were competitive. Some of the decision making came down to engineer familiarity and preference, as many of the providers offer variations of the same kinds of services at relatively the same price. The translation and depth of technology across other platforms was favored however, after looking at the application as a whole – completely end to end – it came down to what would provide the most cohesive experience, and the decision was made to go with AWS.

User Personas

User personas capture the character and interests of potential product users. The following three user personas illustrate users included in the target audience for the bi-lingual video conferencing service.

User Profile A

Derek is a U.S. citizen and native English speaker working in HR at a company based in Spain. His job is to hire working professionals for the company's U.S. and Barcelona offices, which often requires face-to-face interviews. Since Derek works in the U.S. and is not bilingual, he needs a way to meet with potential new hires. More information on Derek can be found in Figure 1 below.



Derek | HR Recruiter

Age: 31
Gender: Male
Status: Single
Location: New York, NY
Language: English
Tier: Skeptic

Bio

Derek is employed at a company based in Barcelona, Spain. He works as a recruiter in their NYC office and spends most of his day on the phone, interviewing candidates.

Hobbies

- Cooking
- Marathon Running
- Dog walking

Goals & Aspirations

- Working on a promotion to HR manager
- Would like to hire 10 more employees by the end of the year
- Wants to be fluent in Spanish to help in job progression

Frustrations

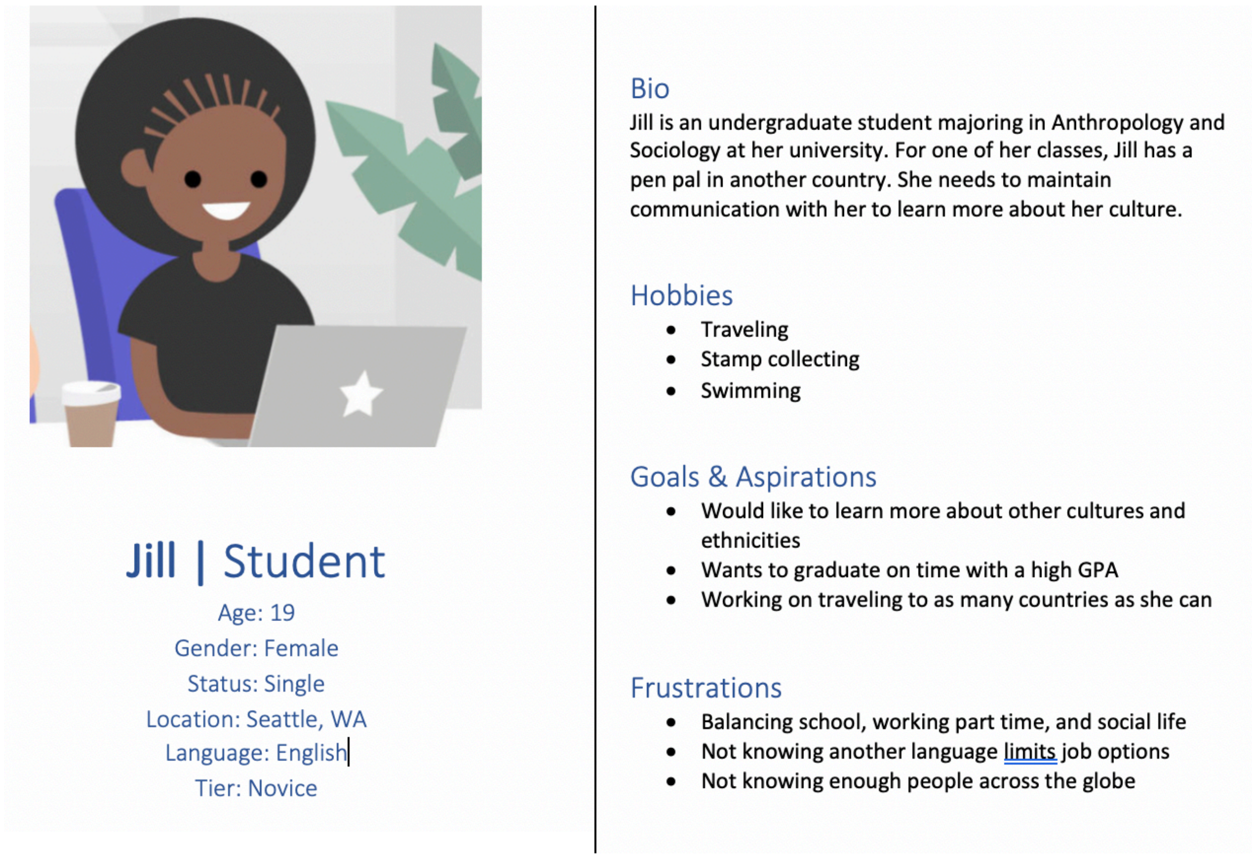
- Working across the different time zones
- Not being able to speak another language fluently
- Lagging video chats in interviews

Figure 1: User Persona A, Derek

User statement: As an industry professional, I want to meet with international, non-native speaking colleagues face-to-face so that I can get my work done quickly with them without needing to learn a new language or hire a translator.

User Profile B

Jill is an undergraduate student majoring in Anthropology and Sociology. As part of an assignment for one of her classes, she needs to meet with a native of another country. To really immerse herself in the culture, Jill wants to meet with someone face-to-face who may not speak the same language as her. More information on Jill can be found in Figure 2 below.



Jill | Student

Age: 19
Gender: Female
Status: Single
Location: Seattle, WA
Language: English
Tier: Novice

Bio

Jill is an undergraduate student majoring in Anthropology and Sociology at her university. For one of her classes, Jill has a pen pal in another country. She needs to maintain communication with her to learn more about her culture.

Hobbies

- Traveling
- Stamp collecting
- Swimming

Goals & Aspirations

- Would like to learn more about other cultures and ethnicities
- Wants to graduate on time with a high GPA
- Working on traveling to as many countries as she can

Frustrations

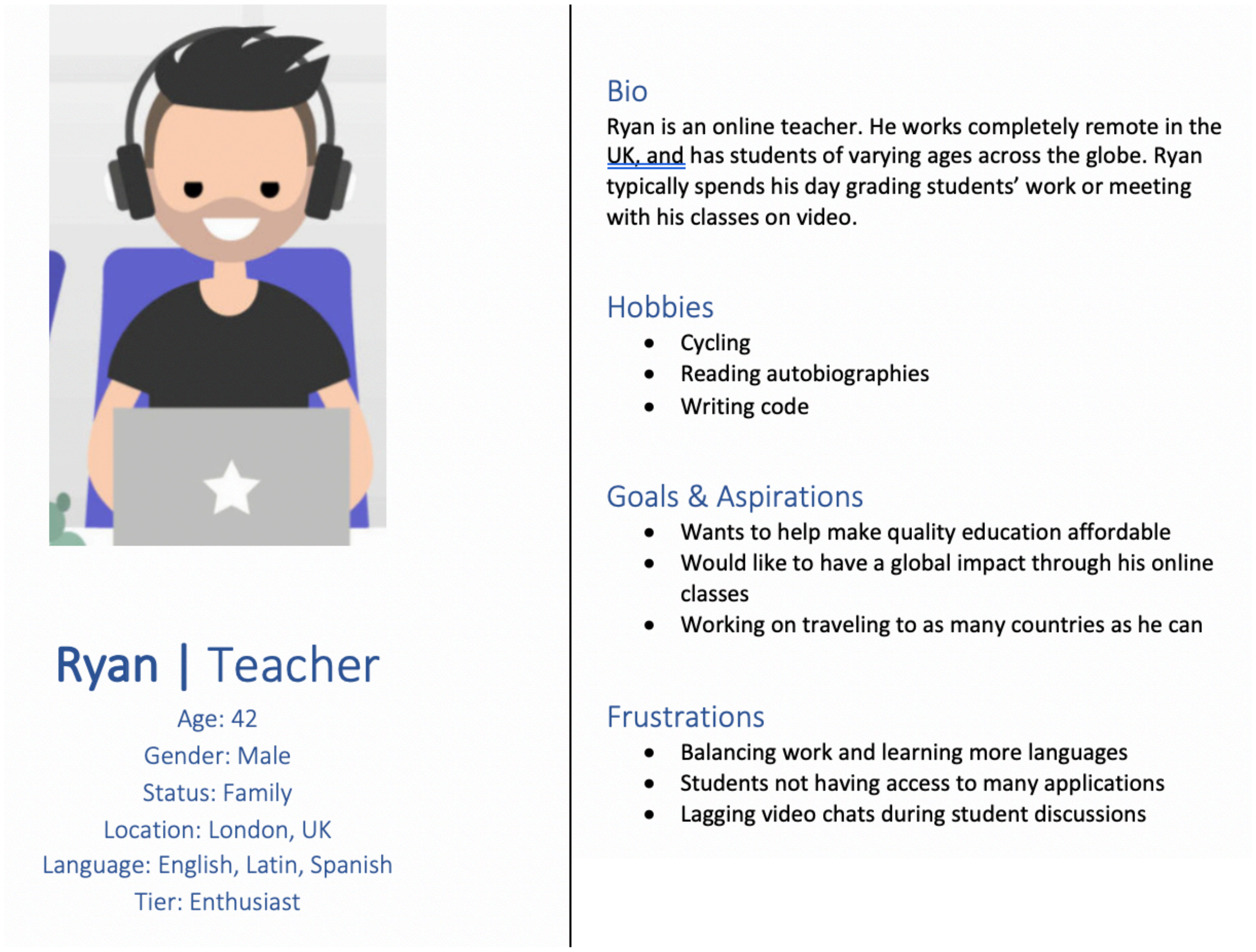
- Balancing school, working part time, and social life
- Not knowing another language limits job options
- Not knowing enough people across the globe

Figure 2: User Persona B, Jill

User statement: As a student, I want to communicate face-to-face with remote individuals who speak different languages so that I can learn more about their culture and identity without either of us needing to be bilingual.

User Profile C

Ryan is an online educator, specializing middle school education. As part of his teaching philosophy, Ryan wants to be able to provide anyone with quality education. Although he can speak multiple different languages, his students come from all over, and usually only speak one language. Ryan prefers to engage with her students face-to-face and can find difficulty doing this with the language barrier. More information on Ryan can be found in Figure 3 below.



Ryan | Teacher

Age: 42

Gender: Male

Status: Family

Location: London, UK

Language: English, Latin, Spanish

Tier: Enthusiast

Bio

Ryan is an online teacher. He works completely remote in the UK, and has students of varying ages across the globe. Ryan typically spends his day grading students' work or meeting with his classes on video.

Hobbies

- Cycling
- Reading autobiographies
- Writing code

Goals & Aspirations

- Wants to help make quality education affordable
- Would like to have a global impact through his online classes
- Working on traveling to as many countries as he can

Frustrations

- Balancing work and learning more languages
- Students not having access to many applications
- Lagging video chats during student discussions

Figure 3: User Persona C, Ryan

User statement: As an online teacher, I want the ability to meet with the students in my diverse class across the globe at any time so that I can provide anyone anywhere with a quality education, regardless of background and language.

These three user personas were used as examples of users that could be part of a target audience. These personas do not account for all use cases of this product and are not representative of the entire target audience.

Minimal Viable Product

The Minimal Viable Product (MVP) is defined as the first release of a product with the basic allotment of features necessary to please users. The point of the MVP is to deliver services and features quickly, always iterating on the initial design and getting continuous feedback from the users. Normally, to help shape the MVP, formal user interviews with a target audience are conducted to gather preliminary ideas and shape the start of the application. For this assignment, a few probable users were created, and a few interested individuals were informally asked for input. From this, there were three high-level features that were defined as absolutely necessary to test the basic concept and functionality.

The first of these features was to have rudimentary video conferencing capabilities. User A would have the ability to dial a number, User B would have the ability to answer or decline the call, and video and audio feeds are enabled. This feature was intended to be very basic – no user profiles, and options limited to call, accept, decline/end, with no preferences to set aside from volume and source language.

This led in to the second feature, a basic user interface. In order for the first feature to be a reality, there needed to be some actionable space available to the user. A keypad or even simply a text field to dial a phone number, a button to place the call, a button to accept the call, and a button to decline or end the call would be included in this interface. Even though the cloud services used have automatic language detection, the users will have to set their spoken language to inform the translator service of the language they wish to set. Aside from a window to show the video, this is the bare minimum functionality.

The last feature included was intended to prove out the bi-lingual piece to the application. According to numerous online publications [7], the top three spoken languages in the world include English, Spanish, and Chinese. Since this product is to be developed in the United States, where English and Spanish are the top two languages, an English to Spanish translation was the most appropriate for the MVP. Translation between two languages was the most minimal feature needed to prove out the concept of the bi-lingual video conferencing application. To start, this was proved out in translated subtitles displayed to the users. Although this may seem like a basic feature to the user, there was a lot of ground work and technical expertise needed to make this a reality.

System Diagram

The bi-lingual video conferencing application had the potential to be immensely complex. There was the user interface and API to be concerned with, which could easily be broken out into separate services as new features are added. The cloud services, which are responsible for storing data and handling the entire translation process, were a large part of running the application and making bi-lingual capabilities a reality. Security (authentication, tokens, IAM roles, policies, permissions, passwords, etc.) and events and logging for visibility and error handling are two other major considerations. The following diagram depicts a very high-level overview of the system architecture – slightly beyond that of MVP - with attempts to call out each of these features in some respect.

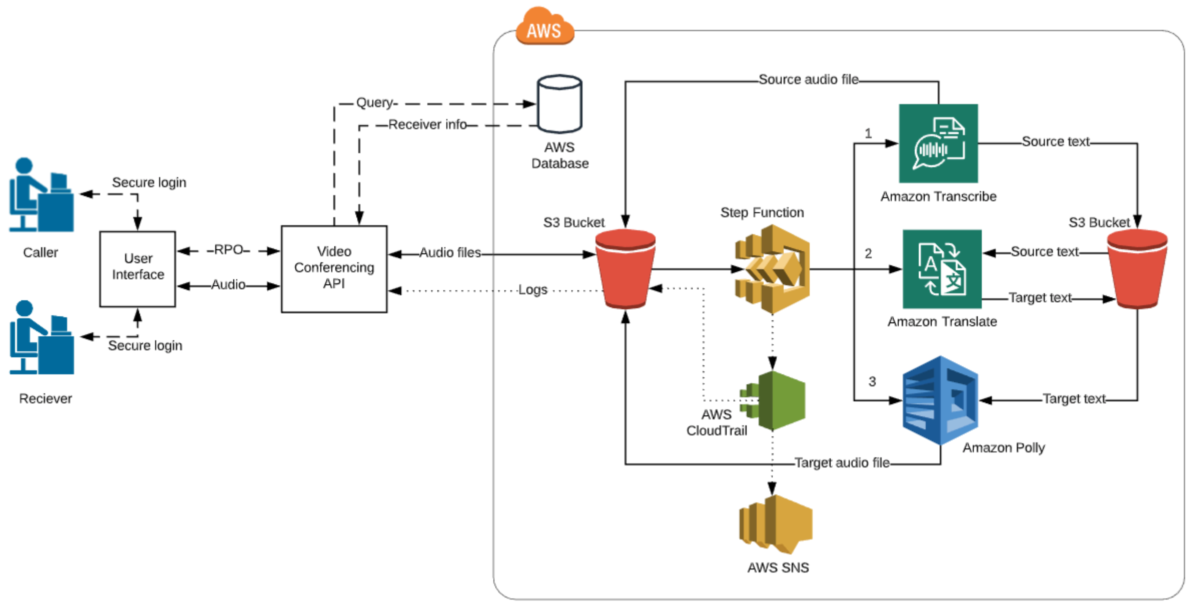


Figure 4: System diagram of the bi-lingual video conferencing application

There are three distinct flows that need to be addressed in this overview, as indicated by the different line types. The first is the dashed line, which illustrates video conferencing user experience. There is a Caller, who inputs a phone number or, in more advanced features, a name or username of the person they wish to call. The diagram calls out that this action will require a secure login but is not something that will be enabled for MVP. The user interface sends this data to the API as a request processing object, which will be used in the API to query an RDS database running out in AWS. This is most likely a MySQL or PostgreSQL query, as a lower-cost relational database would more than suffice. The database will return the requested information (username or phone number) to the API, where a call will be placed to the Receiver and reflected in the interface. Although not called out in the diagram, the interactions with the database will need to make use of security with elements such as dynamic secrets and credentials.

The second flow illustrates the audio and language translation and is represented by the solid lines. When one of the users speak, the audio file is collected and handed back to the API via a request, where it is sent to an S3 bucket along with the user-set language. The bucket will trigger a step function, which is similar to a Lambda function but allows for multiple triggers and actions in a distinct order. When the step function kicks off, it calls Amazon Transcribe, which pulls the audio file from the bucket, converts it to a text file, and places it the same S3 bucket (the image above shows two buckets for diagram clarity but this can in fact be one bucket, as long as each operation places their object in with a distinct prefix to avoid setting off other triggers and confusing the step function). With the source language text file in place, the step function then calls Amazon Translate to grab the file and, using deep learning and neural networks, translates it to the target language. This target language text file is placed back in the S3 bucket. For MVP, the process would end here, and the API pulls the translated text file from the bucket and displays the translated conversation to the user with subtitles. For the complete application, the file placement triggers the third service, Amazon Polly, to convert the text file into an audio file. The audio file is saved to the S3 bucket once more, where the API pulls and passes it to the user interface to play for the Receiver.

The final dotted line represents the logging that will take place. The step function has the capability to write logs to AWS CloudTrail as events occur, such as when a service starts or stops, when a file is saved or pulled from a bucket, or if any errors occur during this process. CloudTrail writes these logs to the S3 bucket, where the API can retrieve and print them to the user or developer. In addition, these logs can be viewed directly in the services in the AWS Console, or can be sent to AWS SNS, where developers can set up email or messaging notifications for successes and failures. All of this logging provides visibility into what is happening in the cloud and is especially useful when testing the application.

Testing Strategy

The testing strategy is comprised of a number of different elements, each meant to test specific parts of the application. The first kind of testing is manual testing, which involves placing a phone call, having a conversation, and ending the phone call. This test is to be done when the application is complete and can involve a number of different test cases such as answering or declining a call, making sure both users can make and accept or decline calls, testing bi-lingual conversations and video chatting when both users speak the same language, and so on.

End to end automation and integration tests are effective tests to write to test the design and functionality of the user interface and its interactions with other services. The front end is written in a scripting language like JavaScript, where a testing framework such as Jasmine is used to test for components existing, object equality, ajax calls, and much more. These types of tests ensure that the user interface is designed properly, components are accounted for, and expected behavior is observed. In this application, e2e testing would ensure that requests for user data and phone numbers were formatted correctly, and responses back from the API with this data are in the correct format. It is also helpful to make sure all components and containers for the video window and phone number input were available and appearing as they should be.

Similarly, for the API, unit tests can be written to mock input and async/sync service calls, validate request objects, and test different kinds of input for expected output. Unit tests are also helpful to make sure errors are handled with appropriate documentation, and exceptions are caught where they should be. For something like a Java SpringBoot video conferencing API, Junit tests make sure correct errors are being thrown when a user enters a bad phone number, or an incorrect call is made to the database. Unit tests in this application also validate that phone numbers are in the proper format and can test different input cases.

Utilizing logging and the AWS Console, as mentioned in the System Diagram, is another way to test and debug. Most processes, events, and errors that can occur are logged when the AWS services are running, and are written to CloudTrail, CloudWatch, and other cloud services. These logs are accessed by APIs and displayed to the engineer, sent to SNS to notify the engineer via email or a messaging platform, or are viewed live directly in the AWS Console or through the AWS CLI. The logs are descriptive and provide the engineer with visibility into the services running, so they are easily able to locate errors and bugs. For instance, if the API returns an error message that it cannot obtain the translated audio file, the logs in the AWS Console might show that the process never even reached Amazon Polly because the translated text file got uploaded with the wrong prefix to the S3 bucket.

As a stretch goal, another testing strategy that would be beneficial for this application would be an external validation service. For example, one of the features post-MVP would be to have users fill out a form with valid email, username, password, birthday, and so on. This form could be validated right in the user interface, or it could be passed to a validation service that would verify that the email is in a proper format, the username is not being used, or that the password follows the correct convention. This could even extend into the API, where calls and SQL statements could be validated for correct format and punctuation.

Challenges

The challenges present in this application revolve around product optimization. They can be broken down into two categories, technical challenges and logistical challenges. The technical challenges represent highly complex technical difficulties, where engineers need to apply creative and unique solutions. The logistical challenges revolve around the product from a business perspective and require plenty of thought and design, but less technical changes.

One of the biggest hurdles in a video conferencing application is the video quality. Poor internet speeds, processing power, outdated hardware, and large distances can all impact the quality of a video chat. Video may be pixelated, blurry, or freeze altogether, and audio may cut out or not line up with the video. Adding real-time language translation into the mix will only add more stress to the service and could potentially slow down the application or make the video quality even worse.

Real-time translation is another technical challenge all on its own. Language translation is possible through services like Amazon Translate, and translation can be done with audio, as the system diagram and video playback depict. This process, however, is highly complex, with multiple services being called, large files and data being transferred and store, machine learning occurring with neural networks, and so on. The cloud services work relatively quickly, but they may not be fast enough to avoid latency issues in a live video chat.

As mentioned above, storage may cause another potential problem. A normal audio file can be quite large (and take a while to process); the audio files for this application will be full conversations that have the potential to continue on for hours. Amazon S3 provides a great deal of storage with their buckets, and, if need be, Amazon Glacier can hold even larger amounts in “longer term” storage. As with any service, S3 does have its limits, and the constant file transferring and storing may reach bucket capacity.

Lastly, perhaps one of the biggest concerns for our user is price. Amazon practices a “pay as you go” model, meaning the user only needs to pay for what they use. The down side to this is that this application is using multiple AWS services, each with their own cost that can add up quickly when they are constantly in use. The higher the cost, the more the engineers need to budget for, which could mean higher prices (or a price) for the users.

Future Releases

In agile, it is common to hold larger planning events to look ahead to future feature releases. They do not go as in-depth as the MVP or current release planning but they do give an overview of what is on the horizon. Since agile revolves around iteration and change, planning any more than two releases ahead is counterproductive and in poor practice. Typically, these features would be dictated by the users and feedback from MVP, with priority set by the product owner. For this application, a high-level outline of some probable features for the next two releases was proposed.

After working through MVP, release two will begin to examine the capability of offering translation between three languages, Chinese being the most probable to add considering its popularity. This may introduce some complexity; however, with the user still setting their language and the support with Amazon Translation, this should not be overly complicated. A second feature would be the ability to playback the translated audio in Amazon Polly. This could be useful for those that may prefer the spoken conversation instead of reading the screen for the conversation but would definitely introduce some challenges with adding in another cloud service. The third feature in this release would be to create a user profile. This feature is more challenging, and will require setting up user settings, security, and new forms in the user interface. This feature would be crucial going forward for users to begin setting their preferences and protecting their data.

The third release is further out and will not be defined as clearly. To progress the bi-lingual piece of the application, it is natural to predict a fourth language will be added in in release three. Amazon Polly offers the ability to select which kind of “human-like” voice to play, so this could be a second feature to add to release three for the user to begin setting their preferences. Finally, as is a popular feature in most video chat applications, release three will include the ability to exchange text in a chat window as an alternative and additional way to communicate. The features outlined in this release may be moved up or pushed back based on user feedback.

Retrospective

A sprint retrospective was held at the end of the week to review how the sprint went. There was only one individual, so it was a solo reflection on what went well, what did not go well, and what could have been done differently. The results of the sprint retrospective are illustrated below.

What went well?	What didn't go well?	What to do differently?
▪ Defining stories first	▪ Kanban Board	▪ Story pointing
▪ Use of GitHub	▪ System diagram	▪ More research
▪ Division of work	▪ Including more technical information	▪ Compare/contrast of cloud providers
▪ Working on research, report, presentation at the same time		▪ Complete product roadmap
▪ Utilizing wiki pages to organize thoughts		

Figure 5: Sprint retrospective board

The "what went well" column illustrates some of the noteworthy processes that went well. Writing out the stories first based off of the requirements worked well because it gave a good overview of what needed to be done and how long it might take. Utilizing GitHub to keep everything in one place was a great design decision and using the wiki pages in there to jot down notes, organize ideas, and break out the stories helped keep everything consistent and organized. By constantly keeping the research and wikis, the report, and the presentation up and working on them simultaneously, it ensured consistency across all three and minimized context switching between stories.

Although the Kanban board was a great way to provide visibility into the progress and track the work being done, it does not include story pointing. In the future, using an agile board that allows for story pointing would be preferred, to accurately estimate story complexity and therefore provide a better understanding as to how much work has been done and how much is left. The system diagram provided a good high-level look at the application architecture, however it ended up turning into a hybrid of MVP and the overall application, so it may be difficult to understand without more context. Diving into the technical aspects of the application a bit more, such as the neural networks and specifics of video chat, would have been more beneficial for the report and the diagram as well. It would have also been nice to include more technical systems diagrams to illustrate security, logging, and what is happening more specifically inside the services.

The last column is similar to the middle one but focuses more on what actionable items can be taken into account for next time. The idea of story pointing, as previously explained, would be something to take advantage of in the next sprint. More research into the mechanics behind video conferencing and language translation would be helpful for adding to the diagram and report. A more effective and formal way of comparing and contrasting the different cloud providers – such as a table with attributes – would work better than listing everything out in a wiki and organizing it later. Finally, having more time to create a full product roadmap, illustrating big key features and releases, would be a good addition to this sprint.

Conclusion

The bi-lingual video conferencing application provides a solution to communicating face-to-face with people speaking different languages. This cloud service has the ability to enhance jobs in industry and academia, improve long distance relationships, and revolutionize communication when traveling. By utilizing the AWS services to transcribe and translate, the MVP includes the capability to host a basic video chat between two languages. Thorough and adequate testing ensures all elements of the application work as expected and will provide visibility into the more challenging technical implementations such as optimizing video quality and real-time translation. Storage and price remain in perspective as the application grows to multiple releases with features such as multiple languages and audio translation support. By using the agile process and taking advantage of agile ceremonies, the application code can be delivered quickly and often, with constant iterations and user feedback loops on new features. The bi-lingual video conferencing application will be a revolutionary product and can be materialized through this process.

References

[1] Azure Pricing. (n.d.). Retrieved August 25, 2019, from https://azure.microsoft.com/en-us/pricing/#transparent-pricing

[2] Machine Translation. (n.d.). Retrieved August 25, 2019, from https://www.microsoft.com/en-us/translator/business/machine-translation/

[3] DialogFlow. (n.d.). Retrieved August 25, 2019, from https://cloud.google.com/dialogflow/

[4] Google Cloud Documentation. (n.d.). Retrieved August 25, 2019, from https://cloud.google.com/docs/

[5] Natual Language Understanding. (n.d.). Retrieved August 25, 2019, from https://www.ibm.com/watson/services/natural-language-understanding/

[6] AWS. (n.d.). Retrieved August 25, 2019, from https://aws.amazon.com/?nc2=h_lg

[7] The Most Spoken Languages in America. (n.d.). Retrieved August 25, 2019, from https://www.worldatlas.com/articles/the-most-spoken-languages-in-america.html