Phonetic Evolution

12/14/2023

FINAL REPORT

Kenzie Norris

Krystian Bednarz

Sam Asher

Preston Lee

Version 1.0

Sponsor: Dr. Okim Kang

Mentor: Vahid F.

Organizer: Michael Leverington

Table of Contents

1. Introduction

Learning a new language is a goal that many people have over the course of their life, but not everyone is able to achieve it. It is a daunting challenge with a vast amount of information to take in that can get overwhelming very easily. Not only do you have to tackle new vocabulary, but also new grammar rules, patterns, and sentence structures. In the midst of all of this, it's very easy to lose track of other important aspects of a new language, like how it is spoken and used in everyday life. An important part of correctly learning a new language is receiving input in both written and spoken format, and in extended chunks so that patterns can be recognized and learned. Due to this, many people who stick to language learning sites don't properly pick up on the intricacies of diverse pronunciation and how contextualization can change the meaning of a word. These two areas are the main things neglected in the modern world of language learning and that's something we aim to fix with our application.

Our project is a web application that consists of various language learning tools and games that aim to teach contextual awareness, as well as provide avenues for self-guided mastery of pronunciation. A main feature of our site is the library page, which gives users the ability to search and explore the large NAU audio corpus that was provided to us by our project sponsors. It includes a context filter that makes it more convenient to specify the context size around a resulting word. Our clients noticed that many existing corpus search tools suffered from overly complex search functions, and were not freely accessible, so our project is our attempt to remedy that. Furthermore, many existing language corpora do not have an audio component, so their language learning utility is restricted to things like grammar and phraseology. The goal is to give users the ability to analyze speech in its natural form, and to pick up on patterns within language. Our language learning tool makes it easy to see the context that a word commonly appears in, as well as how it sounds.

To better facilitate this learning process, our website has multiple gaming elements specifically geared towards critical elements of language learning, such as through pronunciation and contextualization. Users can practice these skills and be rewarded for doing so. Persistence is key when trying to learn a second language, so gamified elements help to ensure that users can enjoy the process and encourage them to keep coming back to our tool. Learners can keep track of and reflect on their usage metrics, which include how many words they have practiced, their score in the various games, as well as unlockable customizations for their profile image. Being able to look back on your progress gives a sense of accomplishment, and can help to motivate users to

continue practicing. There are also leaderboards for the games so that users can compete with their peers.

Our main goal with this project was to create a tool that would allow users to easily explore the NAU audio corpus, while implementing gamified elements to help drive user engagement. We believe that we achieved this goal, and we are satisfied with the results. The following sections will dive deeper into the fine details of our project, and hopefully give a comprehensive guide to its inner workings.

2. Process Overview

After being placed into our team for the project, we began a process of creating documents and organizational procedures to help guide us into the future, and make collaboration easier. This included assigning members into roles geared towards specific areas of development. When working in a team, it is important to identify what standards each member should uphold before any real project work begins. Our first task was the creation of a Team Standards document, where we wrote out how to be productive team members, how to tackle problems that might occur while working in a team, and what areas each team member would have primary responsibility for.

A major factor in the success of any software development team is communication. Being able to assign tasks to your team members, and ask them for help in areas of project development promptly is crucial. It is expected that each team member can communicate where they are in terms of their assigned portion of project development. This is especially important when the team has strict deadlines to meet. Reaching a deadline and not knowing that a team member could not produce what they were assigned can be detrimental. The team members made a point of frequently updating the others whenever progress was made. In addition, we highlighted potential problems as soon as they arose, so that they weren't overlooked and the problem solving process could start as soon as possible. In our case, we created a team Discord server, a private messaging platform where each member can discuss project development while the project is being made. Adhering to our standards of timely updates, everyone knew how far along each member was in their work, and the development process went relatively smooth as a result. Our team held weekly voice calls to discuss work progression, with additional meetings planned based on need. This was a crucial component in making sure that everyone was kept up to date on the project's progress, as well as for planning upcoming work and assigning tasks. We also did our best to communicate with our client at least once a week, usually through a progress report email, but occasionally in-person when necessary.

Specialized roles can be useful when certain team members are more familiar with specific areas of programming compared to others. Specializations also help to make project development more orderly. Team members can consistently make adjustments to one component of the project that they are increasingly familiar with, and the others know who to talk to when inquiring about said component. In our team, for example, we had a dedicated Database Coordinator, who was responsible for supervising work that involved setting up, maintaining, and accessing the database. Another crucial role in our team was the Team Lead, tasked with the coordination of work for the rest of the team members. The lead figures out who needs to do what, and how to do that work efficiently. This role also comes with the responsibilities of keeping in contact with the client and ensuring that important deadlines are met. Everyone needs to be involved in programming, but organizing who should program what through team member specializations keeps workflow more organized.

When working in a team, problems are inevitable. What matters is how they are handled, and how the team continues to move forward. We planned for these eventual hurdles with our Team Standards document, which gave a detailed step-by-step process for settling team member disputes, in addition to laying out guidelines for how team members are expected to communicate and work together. Communicating when someone does something that makes another team member feel a negative way is necessary; never let bad feelings fester into something worse. Always be transparent with your team members about how you feel, and resolve conflict as soon as possible. For work-related problems, related to team members having trouble completing certain tasks, make the effort to help them solve those issues. You are a team, you need to work together. Consistent lack of performance from a team member needs to be addressed and handled with the utmost care. However, each team member needs to make an effort to support others who are struggling. Put your pride aside and focus on getting everyone to be productive, instead of blaming someone for their shortcomings, and opting to refuse to help them at all.

With these concepts in mind, how does a team develop a project? Our experience involved determining what incoming deadlines needed to be accounted for, which project components needed to be completed, and by what time. Deadlines are more straightforward: complete this part by this time. Such deadlines mostly came from documentation completion, such as the final draft of the Requirements Specification document, or when the next Design Review presentation needed to be conducted, as examples. Start working on assignments with a designated deadline early, and review completed assignments multiple times before release. For actual programming

assignments, deadlines occur as demos of the project hosted on a certain day. Since what defines a project as demo-ready depends on the project being made, what exactly needs to be functional before a demo also varies. This means it is for the team to decide what would qualify as an acceptable prototype for presenting during those demos. As a team, you need to figure out what requirements need to be met, and how you would present those requirements. Our design process worked this way: separate project requirements into digestible components, assign components to be completed to each member, communicate progress about those components throughout, and set a fair amount of time for component integration into a single prototype.

3: Requirements

In the first section of our Capstone experience we created a Requirements Specification document. The goal of this document was to determine what our clients wanted out of the product, and what we needed to do in order to make that work. Our requirements acquisition process involved multiple client meetings, where we noted what exactly our project needed to achieve, how feasible certain requirements would be, and what restrictions we had for achieving certain requirements. After defining the features our project needed to have, and having our client agree to what we envisioned, these were the requirements we were able to outline:

3.1: Solution Vision

Our solution is a gamified language data explorer that helps with pronunciation through analysis of data, or PANDA for short. It is a browser-based web application that is connected to a large audio database provided to us by our sponsors. The following are the main features that we envisioned in our final product:

- **Library Search Function:** One of the key requirements of our project. Allows users to explore the accompanying database and interact with the results.
- **Administrator Page:** This will be a useful page for our sponsors, or whoever they designate as administrators to control and edit different aspects of the application. It will include functionalities that allow the administrators to interact with the database and user accounts, in addition to user group changes.
- **User Profiles:** Contains user-specific data such as learning statistics, game progress, and earned points from game performance. This is where the user will be able to access their personal notebook that displays all this information.

- **Practice Page:** Students will be able to select a word from the results page to reach the practice page or go to the practice page manually, where the game page will select words to work on automatically from the user's personal notebook. The practice page will contain the three following games to help users learn: The Pronunciation Game, the Context Game, and the Dictionary Game.
- **Leaderboard:** A good way for users to have friendly competition to drive their progress, either between all players or a select group. To help with user engagement, we believe that it's important to add features that will reward interaction and give feedback to help learners track their progress. Giving them a motivational percentile score to compare with others would provide a goal to strive towards, a reason to come back, and gratification to see their placement go up.

3.2: Functional Requirements

Functional Requirements are features with measurable metrics. Having them laid out in this form allows our team to know what must be fully functional by the end of the project. The following section details these steps to further guarantee requirements are met:

3.2.1: Library Search Functionality

As the website will be built for aspiring English learners, it's important to provide them with the ability to search through our database to find what they need. The contents should be displayed to their liking, as well as be interactable to provide full context for easier application.

- The system should be able to search for a singular word, or a pair of words, providing each entry that satisfies the query, without duplicates.
- The user should have the ability to hone in on select words to see a number of words in either direction of the query. This is to make the user able to hear the example sentence closer to the moment the word is said, rather than in its entirety every time.
- The system should have the ability to start and pause the audio segment at the convenience of the user's command.

3.2.2: User Accounts

Wanting the user to feel inclined to come back, the website must include an optional saving feature through accounts to allow for a more personal, educational, and enjoyable experience. The account must:

- Have a username and password. The password must be stored with some form of encryption to prevent damage from data leaks.
    - Password resetting will be done through administrators.
- Display the daily goal, and whether it is completed or not.
- Display statistics on game performance.
- Store which avatar is being displayed to enhance user customization and uniqueness.

3.2.3: Website Administration

With the need for adding content and managing what's available, an administrator role in some form is required for the project. The administrator should be able to add, remove, and edit anything using the UI, allowing them to make direct changes in the database.

- The administrator must have the ability to make changes in the database to better suit their needs.
- The administrator must have access to active user stats to both see how the site is being used, as well as how the community is enjoying it.

3.2.4: Practice Page

With the practice of words being one of the main attractions of the website, instances of that need to be numerous, and more involved than just flashcards. To do this the team came up with three simple games to start testing the user ability through pronunciation, filling in the blank, as well as context definitions.

- **The Pronunciation Game:** A game that allows users to practice their pronunciation of words by listening to and repeating words through speech. The user will be able to rate themselves to help track their progression. It must provide the user with a word in a given context, ask for a temporary recording with multiple retries of the user saying the prompt, and then provide the actual recording for comparison to their own audio. To aid the user in self-reflection, they are encouraged to rate themselves and are able to retry as many times as they'd like.

- **The Context Game:** A fill-in-the-blank game where users are tasked with figuring out what word best fits into a sentence. For each question the game must display a new sentence with a new word removed, have four options of words to choose from, and have one of those options be the correct answer. The user earns one point for answering the question correctly, and the correct answer is displayed if the user answers incorrectly. This process of asking and answering questions should be repeatable for the user to play as many times as they wish.
- **The Dictionary Game:** A game that works on a user's word definition abilities, as well as contextual reasoning skills. This game focuses on determining definitions for words based on their context in a sentence. The game selects a word in a sentence given the context. Providing the user with possible dictionary sources online encourages the user to come up with their own definition and compare it with the dictionary.

3.3: Performance Requirements

Performance requirements are set metrics that analyze how well the features in the project work. Such metrics include how quickly information is presented to the user, varying based on how optimized our code is. For each feature, here are the metrics we agreed upon:

- **Library Search Functionality**: The search can take a varying amount of time depending on how many times the word appears within the database, but our goal was to keep the initial results appearing in under 5 seconds, and the loading of the rest of the results shouldn't take any longer than 30 seconds.
    - This metric is based on the sample size of data we were provided by the clients. Any changes to the data set in the database will change site speed.
- **User Accounts**: The user account will have a maximum of 50 definitions and 50 starred sections.
- **Website Administration**: The changes to the database happen within 10 seconds of the request.
- **Practice Page**: The games feel fluid, minimizing page updates. The page only updates once to load each game.

3.4: Environmental Requirements

The key environmental requirement of our project is that we cannot use Gmail authentication for our password system, as the project needs to be accessible by students

from China. The requirements listed in this section have been made from meeting with the client directly, and have been made to show what is needed in the final product at the minimum. The website will be run on HTML with CSS and javascript, using PHP and MySQL in the back-end on an EC2 AWS instance.

4. Architecture and Implementation

Our project can be subdivided into two main components that work together to create the final solution. All of the games and user elements are implemented on the frontend website, hosted on an Apache web server and written mainly in PHP and HTML, with some Javascript scattered throughout to help make the games work properly. The back-end database is implemented using mySQL and hosted on an Amazon Web Service (AWS) EC2 instance. User account information is also stored in the back-end. User stats, and their related game scores and definitions are associated with their user ID so that the appropriate information can be shown on their profile page. *Figure 1* below gives a high level overview of the general system architecture.

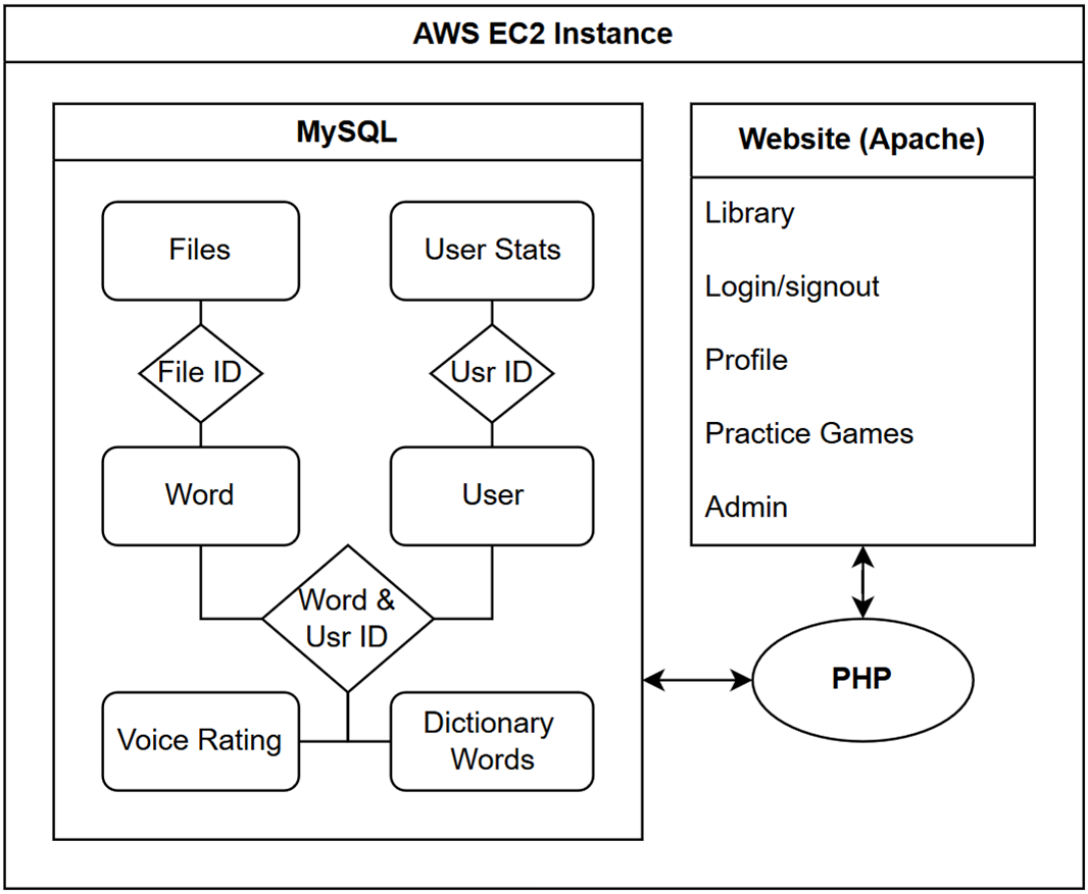

Figure 1: High-Level Architecture Diagram

First, let's do a walkthrough of each of the front-end components, and how they work with one another. All of the files used for the front-facing website portion are stored in one folder, as well as their associated stylesheets and any graphics that they use. The landing page of the website is the library page. As stated earlier, the key responsibility of this page is to allow the user to search and explore the database. It works by performing a SQL query to the database through PHP code that is written alongside the HTML of the page, and then sequentially displaying the results. A header.php file is used to keep a consistent header throughout every page of the website, and is imported at the start of each file. From the library page, you can access any of the other pages except for the admin page, which is only accessible by user accounts that have been placed in the administrator group.

Navigating to the login/sign up pages, users are able to log in to an existing account, create a new one, or log out if they are already signed in. Users' login information is manually hashed and salted before being stored, and must be decrypted anytime that it is accessed to ensure that it is kept secure. Again, PHP is used to communicate between these pages and the database, to both store and retrieve information.

Also accessible through the library page are the practice game pages. Clicking on practice in the navigation drop down menu will bring you to a menu page that will allow you to select which game you want to play. All three games work in a similar way, their main differences lie in how they ask the user to interact with them. Unless the user has selected a specific word from the library page to practice, then the initialization is the same for each game: a random word is selected from the audio corpus, and is displayed to the user along with a few words surrounding it on each side for context. Each time a word is practiced in any one of these pages, the user's "words practiced" statistic is updated within their profile page. All practice statistics displayed to the user are both stored in and taken from the database.

The pronunciation game allows the learner to listen to the randomly selected word in its given context. They are then prompted to record themselves repeating the word or sentence and to listen to the playback. Afterwards, they can rate themselves using the slider at the bottom of the page, although it is not required. Ratings are anywhere from 1-5 based on how they felt they did pronouncing the word or sentence. These ratings are stored in the database and associated with the user who is signed in. The ratings will also appear on the users profile page. Recordings created on this page are not stored, in order to protect user privacy.

The context game selects the original random word and context, as well as three other words and their associated context. There are two modes for the context game: word matching and sentence matching. Each mode is the inverse of the other. In the word matching mode, the user is given the randomly selected word, and must choose the sentence that it fits best into out of 4 multiple choice options presented beneath it. Those options are other viable sentences, with a word removed. The sentence matching mode presents the user with the original words context, with a blank where a word should go. The user then must select the word that best fits that specific sentence out of 4 multiple choice options presented beneath.

The dictionary game presents the user with the randomly selected word and its associated context. This time the goal is to try to use the context around the word to write a definition for it. As usual, these definitions can be found on the users profile page, as they are also stored in the database and associated with the userID of whoever is signed in and playing the game.

The profile page is accessed by logging in or creating an account and clicking on the profile icon in the upper right corner of the screen. Attempts to reach this page without signing in first will simply display a message prompting the user to do so. The profile page lets the user view their progression by displaying statistics such as their score in the context game, their saved definitions from the dictionary game, their pronunciation scores from the pronunciation game, game leaderboard rankings, how many words they have practiced in total, and their day streak. Also included are daily challenges that further incentive users to play certain games or practice more words. All of these metrics are stored in the back-end and are associated with a userID.

The back-end database is separated into two main sections that are occasionally linked together. The meat of the database are the actual audio and transcription files. The audio files are in .flac format, which is like a high definition MP3 file. Accompanying each audio file is a .TextGrid file and a .txt file, that are named to match the audio file. The .txt file begins with the name of the audio file that it correlates with, and then provides a transcription with periodic timestamps throughout. The corresponding .TextGrid file is a more detailed synopsis of each word in the file and the exact time that it is spoken. This is what allows the library page to function with contextualization like it does. All of these files are tied together and given a File ID in the database, which allows them to be fetched and indexed through. Each word in the database is connected with the File ID's that contain it.

These words are then connected to user accounts using their unique userID. The combination of the word and user identifiers allows for population of the profile page for

each user. We check which words their userID is associated with, then fetch and display the relevant data from there.

The admin page is the main link between the back-end and the front-end, and is meant as a user-friendly tool for anyone with sufficient privileges to easily make changes to either end. As such, the website first checks with the database to ensure that the connecting user is a part of the admin group before allowing access to this page. Non-admin users are redirected away from the page. Admins are also required to re-enter their password before changes can be made as an extra layer of security. Admins have the power to change user passwords if necessary, change what group a user is placed into, and to add and remove files from the database easily. User account information is changed using the modify_user.php file, which is called from within admin.php. To modify a user's group, a combination of fetch_user_names.php and fetch_user_groups.php are used.

We followed our development plan very closely, and ended up with a product that implemented essentially every feature that we wanted. We had to make some changes to our original structure for the database to make some features work, and to have it interact with the games correctly, but for the most part we stuck to our original design. We initially planned to code the gaming portions of our website in python, but received input from our mentor that PHP would be a better option, so we switched to that.

5. Testing

To ensure our final product is fully functional and works the way our team intended, our team exercised a rigorous software testing plan once all project components were fully integrated. For our project, our main focus for software testing comes from user data, and how that data is interpreted by all facets of our site. Back-end information comes from both stored word data (text-based / audio-based,) and information gathered from the user (account information / words searched.) Dynamically changing back-end information is a major component for software testing, as the static data stored on the site would only need to be searched for and remain unchanged. Areas of our project where users input specific forms of information needed to be tested much more. For example, a user’s input that is typed into a search bar varies much more drastically than comparing if a user pressed one button over another. Regardless of which facets of our project require more testing over others, it was still necessary to check if everything in the project works as intended, so every part of our project needed to be tested.

5.1: Unit Testing

For each component in our project, we determined what areas in that component required inputs of some kind, we accounted for as many erroneous kinds of inputs as possible, and we modified how that component analyzed those inputs. Components requiring inputs contained at least one of the following types of inputs: numerical text, character text (words and sentences,) audio file changes, user group changes, and database changes. We first determined what equivalence partitions and boundary variables each type of input had, whether that be an exceedingly long string of characters in a text box, or inputting an empty file into the database, as examples. We then programmed how the component handled strange inputs, retested that code, and repeated this process until we deemed that the component as error proof as possible.

Outside of user based input entry, we checked to see if the data being sent from component to component is functioning properly. Components such as the Library page, for example, require the usage of the database to gather the sentences an inputted word is contained in. The result of this testing process produced a more robust project with a lesser likelihood of something breaking as a result of malicious inputs into our project.

5.2: Integration Testing

The integration testing process for our project involved determining what data was being sent, which component that data was sent from, and how the receiving component handled that data. The Library page is the first place the user encounters when using the site, so we started there. Although we tested that the data is pulled properly in Unit testing, our main goal here was to ensure that the audio playback is functioning, as well as how the library functions with the account system. The library shouldn't allow the user to access any of the other pages until the user signs up or logs into their account. The easiest way we tested this was to ensure as the user is not logged in, they cannot access any other links related to the site.

Once the user is logged in they will have full access to the rest of the site. From the Library page, a logged in user is able to connect to any one of the different practice pages. The user can select a game, and if they have a word selected, they have the option to practice it. The easiest way to test this was with games such as the Pronunciation game and the Dictionary game pages. A user should be able to enter the word "test" and then select either of these practice pages. Once they select either of the games, it will prompt them with the word "test" rather than any other random option. This will allow the library and the practice pages to be completely connected

Once the user begins to practice, their words practiced, the definitions they create in the dictionary game page, and any ratings they give themselves on word pronunciations will be stored to the userID. This means when they access the user account page, they will see an overview of their progress. The best way to test this was to have a user practice words over the course of a few days. We set up a few test users and ran through each day as a regular user of the site would. We ensured that we had relevant variables taken into account, done so to view the results at the end of our testing period.

Admins of the website are able to see all of this data on the administration page. Once the testing of the User Account page was finished, we then tested the integration of the Administration page. After creating users and practicing a few days with them, they appear as editable users inside the Administration page. One of the final tests to ensure that all of the pages are integrated fully was the administrator's ability to add new data or remove old data from the library. As an admin we simply added a new file with our own text-grid and audio files, and did the same unit test to ensure that these files have been properly integrated to the library.

The results of the integration testing process highlighted major breakpoints in our project, where each possible navigation that could be problematic was resolved on a case by case basis. We encountered integration testing problems before software testing even began, going back to the Alpha Demo prototype we created midway through the semester. Beyond resolving those issues then, and resolving the new ones encountered conducting the actual testing plan, we made the site more reliable to use.

5.3: Usability Testing

Our usability testing process was based on the requirements we acquired early on in project development. While acquiring requirements allowed us to outline what exactly our project needed to have, usability testing instead involved how *well* those requirements were implemented into the project itself. To figure that out, we invited our colleagues to explore our site to determine the efficacy of our requirements as features in action.

Moderated usability testing allowed us to observe a user’s interactions with the software and gave us insights on what parts of the user experience can be improved upon. By watching what they did while guiding them through a series of tasks, we were able to determine how quickly they learn to use core features, as well as how intuitive the design is. The users we invited tested all aspects of the site, ranging from their interactions with the practice games to their perusal of word searching in our library page. For each component they explored, we ran through the requirements previously outlined, took

notes on how well the participants thought we did on each requirement, and noted criticisms on how to improve each requirement. After retrieving the data they provided, we tweaked the site accordingly, ensuring our site is both more intuitive to use and more effective as a project as possible.

6. Timeline

From the very beginning, our team has been keeping track of the work we have been producing. For document deliverables to completion of essential project components, we have cataloged all of our submissions throughout project development. Here we will overview our condensed timelines that outline when we completed each part of the project, starting off with our first semester of the capstone experience in *Figure 2*:

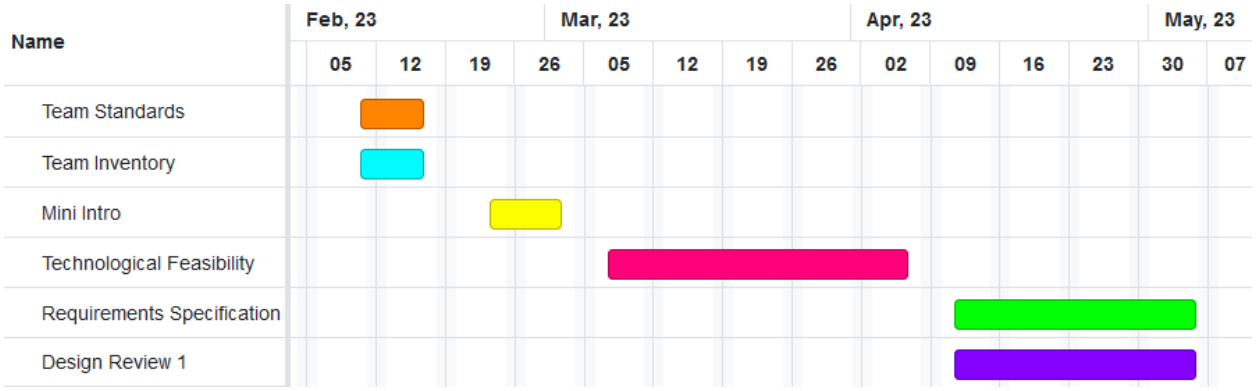

Figure 2: First Semester Gantt Chart

We started off first writing our Team Standards document, where we agreed upon a means to work together and determine how to solve team problems. The Team Inventory document listed our various skills, overviewing what properties we all had that would prove beneficial towards project development. After introducing ourselves in our Mini Intro, we began development of the two major documents that we would reference for the rest of the project's development: the Technological Feasibility and Requirements Specification documents. These two documents explained how feasible it would be to develop components of what our clients had envisioned, and what features our project needed to have, respectively.

We concluded the first semester with our first Design Review presentation, overviewing all the work we had done that semester, and what our future plans were for the next one. In that presentation, we determined the five major milestones we would

complete in the future. These milestones are shown here in *Figure 3*, displaying the major components necessary for project completion:

Figure 3: Major Milestones in Project Development

Work in the first semester consisted mostly of document writing and planning for software development in the next semester. That being said, we continue the timeline to the next semester in *Figure 4*.

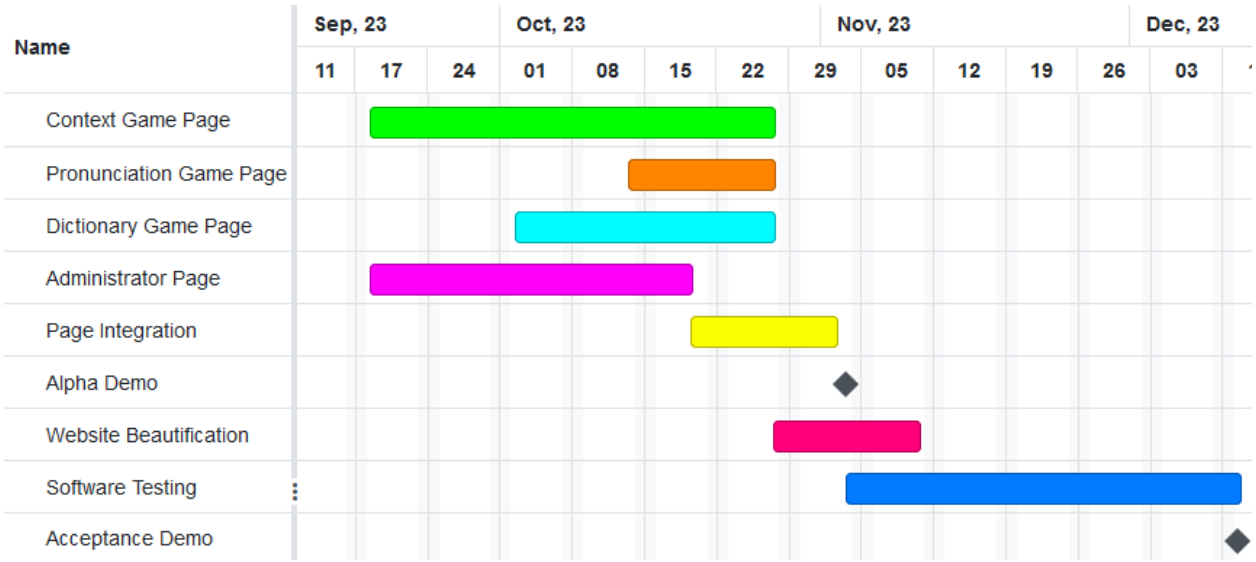

Figure 4: Second Semester Gantt Chart

Work in the second semester started with the development of the three game pages: the Context, Pronunciation, and Dictionary games. User and Admin pages were also started at this time. The development of these pages needed to be completed and integrated before our Alpha Demo presentation, where we presented project functionality

with our mentor. After the Alpha demo was conducted, we moved on to software testing, bug fixing and usability testing. Throughout this semester, we also conducted various design review presentations and wrote supporting documents, but only the major areas of project completion are displayed in *Figure 4* for clarity.

Each team member was assigned a game page to work on: Krystian made the Context Game, Sam made the Dictionary Game, and Preston made the Pronunciation Game, in addition to working on the Administrator Page. Kenzie worked on the various presentations we needed to conduct, such as the Project Info Mini Video that she animated and choreographed herself. We all met as a team to finish up the pages and integrate them into the Alpha Demo prototype, and we all fixed the various bugs that came up as implementation concluded. Documents in this semester were handled by Krystian, Sam, and Kenzie as well. Kenzie also guided the Website Beautification process by aiding everyone to align with a set page format, so all pages on the site looked the same for consistency. Preston also handled backend integration for all pages, login and signup functionality for the account systems, and spearheaded general bug fixing throughout project development.

Development of the game pages took much longer than expected to complete. Although we submitted everything within a reasonable timeframe, we needed to alot more time to the pages than we anticipated. Though everything was submitted within a reasonable timeframe, in hindsight getting started on those pages even sooner would have made project development a much more smooth process. We had multiple extra meetings conducted before the Alpha Demo prototype was presented to make sure everything was fully functional. Also, our schedule did not have the time for us to work on the stretch goals we had planned, but we still submitted a final product our team was proud to deliver.

7. Future Work

We believe that with this application, we have provided a strong foundation for future expansion. Our clients envisioned this project as a small portion of a larger learning platform. In meetings with our clients they mentioned technology that they have been working on in parallel that would allow for better analysis of the recordings that users create of their own speech. At the moment, many of the games we implemented put an emphasis on self-reflection. This is because we believe that it is an important aspect of learning, and also because we did not have the time necessary to implement more advanced evaluation systems. Our clients have been working on leveraging AI neural networks in order to learn and rate pronunciation patterns. This would be a great feature

to add to the pronunciation game, as it would allow users to get more technical feedback on their speaking. We have tried to set up the pronunciation page specifically to be flexible enough to add this additional logic in at some point.

Another stretch goal that we identified throughout the development process that unfortunately didn't make it into the final application is a more sophisticated word selection algorithm for each of the games. At the moment the pronunciation, context, and dictionary game all rely on random word selection from the database to work. This is sufficient for their purposes, but sometimes can produce suboptimal results. For example, in the context game there is always the possibility that the game could select two words that both fit the prompt, but technically only one is correct. While it doesn't happen very often, when it does it can be very frustrating for the user. The dictionary game also suffers from a similar problem, where sometimes the randomly selected word is difficult or uninteresting to define ("uh," "a," "um," etc.)

We discussed this issue with our clients and they had a reasonable solution that could be implemented in the future. There is technology which would allow us to add tags to the word files inside of the database so that we could identify them based on characteristics such as "noun," "verb," "adjective," etc. With this we would be able to eliminate much of the problem that we are currently facing with the games.

Finally, there is a lot of potential for expansion of the user reward and challenge systems. Currently users can unlock a total of 6 different user icons for completing challenges and playing practice games. It would be very easy to add more customization unlocks, as well as more complex and complicated daily challenges.

8. Conclusion

The goal of our project was to create a gamified language data explorer that is user friendly and encourages learners to repeatedly practice skills related to learning a new language. This tool allows users to easily explore the extensive NAU audio corpus to help them with their language learning journey, in addition to giving them fun ways to practice their word contextualization and pronunciation skills. Our tool was created as a means to improve upon corpus based language learning techniques. We wanted to fix the problems of over complicated search features, restricted accessibility, as well as leverage the higher utility of a corpus that contains not just text, but a matching audio component.

Our project will be an important addition to a larger set of language learning tools that our clients have been developing, and will allow them to better test their language learning theories. It gives them insights into the effectiveness of gamified corpus

language learning by providing important usage metrics. Outside of the scope of our clients, this project has the potential to change our perspective on what is important in the language learning process. We may be able to highlight different areas and techniques that, if given a heavier emphasis, can enhance the way that new languages are taught.

Our capstone experience gave us a lot of practice with the software development process. We learned the importance of careful and detailed planning, as well as how to effectively work as a team to efficiently accomplish tasks and reach predetermined goals. We got the chance to dip our toes into the water of the professional world, work with a real client, and experience the ups and downs that come with any type of large scale team effort. We not only built up our programming prowess, but also developed more strong interpersonal skills which will help us to advance our careers outside of a university setting. Multiple presentations throughout the year gave us the opportunity to refine an elevator pitch for our product, an important part of getting any project into development. In the end, we overcame the obstacles that we knew were coming and that we planned ahead for, and delivered a product that we are all very satisfied with. We hope that our tool helps not only our clients, but anyone who wants to learn a new language.

Glossary

AWS - Amazon Web Service - an on demand cloud computing platform
Corpus - a collection or body of knowledge or evidence.
Discord - an instant messaging social platform which allows communication through voice calls, video calls, text messaging, and media and files.
UI - User Interface - a means for the user to interact with the site through intuitive means, such as by using buttons or text boxes.
AI - Artificial Intelligence - a program that actively learns and refactors itself to better perform the task it was made to do
SSH - Secure Shell Protocol - allows a user to remotely and securely access another device over a network.
PHP - A general-purpose scripting language geared towards web development. PHP is a recursive acronym for PHP: Hypertext Preprocessor.
HTTP - HyperText Transfer Protocol - an application layer network protocol used for transferring information between networked devices.

Appendix: Development Environment

This section will be a guide to the development environments and toolchains that were used in this project, and should be a handy tool for quickly getting things set up and ready to go so that future development can be done.

Hardware:

The team developed on a mix of Linux and Windows machines. That being said, the backend database was created on and functions best with an Ubuntu distribution of Linux. Setting it up in this way is a lot less hassle than on windows, and so that is what is recommended. Team members who didn't have access to a linux machine were able to use a virtual machine to achieve the desired results. Oracle's VM Virtual Box worked great, and it is recommended to use at least the default settings when setting up the machine, but any extra memory that you can give to the machine is helpful. Ensure that you have admin/sudo rights.

Toolchain:

You can use whichever IDE you like best to work with the PHP code. Our team mainly used VScode. The initial script used to populate the database is written in Python, so if you don't have the database setup yet then you'll need to ensure that you have a version of Python installed on your machine. Additionally, if you are setting the database and server up on a Linux distribution it is important to have terminal access and administrative (sudo) rights. MySQL is necessary for running and querying the database. We used an Apache HTTP server to host our PHP files. Many team members made use of XAMPP to easily test PHP files without quick and easy access to the server.

Setup:

Since we only used Linux for deploying the website and database these instructions will be based around a Ubuntu-based Linux distribution. Setup should be possible on windows machines, but may require some extra steps and toolchains that will not be covered here. As stated earlier, make sure that you have terminal access and administrative (sudo) rights on the linux machine that you are using.

First, open a terminal and perform a general update of your linux repositories by typing **sudo apt update** and pressing enter. Any command featuring "sudo" may ask you to enter your linux user password. This is the same password that you would use to log in to your linux machine. Text may not appear in the terminal while you are typing, that is

normal, just type your password in and press enter. Once these updates have finished install Apache by typing **sudo apt install apache2**. After Apache has finished installing, do the same for PHP with **sudo apt install php libapache2-mod-php php-mysql**. Next, Apache needs to be restarted to enable PHP to work with it. Type **sudo systemctl restart apache2** and press enter.

Now that you have Apache and PHP set up then you can setup mySQL for the database. In the terminal, navigate to the website folder that can be downloaded from the PANDA github repository. The command **sudo apt install mysql-server** and **sudo mysql_secure_installation** will install the MySQL Server software necessary for hosting the database. Now open mysql with **sudo mysql -p**. You can omit the -p if you set up your linux root user without a password. Now create the database that we will upload the files to with **CREATE DATABASE Evolution_DB;**. To start using it type **use Evolution_DB;**. The command **source mysql;** will pull from the premade MySQL schema within the website folder and automatically configure the database to store the data how we need. Next use **CREATE USER 'your_username'@'localhost' IDENTIFIED BY 'your_password';** , filling in the your_username and your_password sections with the administrator login information that you would like to use. This is making the profile that PHP can use for modifying the backend. Now **GRANT ALL PRIVILEGES ON Evolution_DB * TO 'your_username'@'localhost';**, which allows the profile to make changes. Finally **FLUSH PRIVILEGES;** to finalize it. Ctrl + d will get you out of MySQL.

Now you're ready to populate the database, which is easily done using the provided data_transfer.py file that is also included within the github repository files. If you don't already have python installed then enter **sudo apt install python3** in your terminal and let it install. Once that is done you'll need to also install the mysql python connector using **pip install mysql-connector-python**. Place the data_transfer.py file in the same directory as the folder containing the flac and textgrid folders. Open the data_transfer.py file with your favorite text editor and make sure that the host, user, and password areas are set to the same as the ones that you created above. Save and exit the file and run it with **python3 data_transfer.py**. This script will populate the database with the textgrid and flac files.

To set up your admin account, first go to your host's IP and make a profile using the signup page. SSH into the host device and open the MySQL, then **use Evolution_DB;** like before to access the database. **SELECT * FROM User;** to find your userId and then **UPDATE User SET groupRole = 'admin' WHERE userID = 1** , make sure to change "1" to whatever your userID is. Ctrl + d will exit MySQL and your user account will have admin privileges.

Production Cycle:

Our production cycle involved a few main steps. The first step is always requirements gathering. After identifying the changes that we want to make, we made sure to understand what functional and non-functional requirements the modification would require. The goal is to get as clear of a picture as possible of what the end result would look like before we even touched the code. From there we would create a more in depth design implementation. This would delve more into the details of how the change would actually be made. For example, which files will be affected and do other files depend on that one? It's important to understand the ripple effect that sometimes small changes may have on the project as a whole. Once we had a solid plan we would begin development. Usually at this point whoever was taking the lead on that specific feature would fork the repo and begin working in their own branch on the implementation.

Once the implementation is complete, testing would be done on the branch. Depending on the feature different tests can be performed. Please see the testing section of this document for a more in-depth look at the different testing methods that we used. Finally, if all testing went smoothly, then the branch will be merged into the main repository. We do our best to write documentation during the entire development process because we feel that it is easier, and the most accurate way to get a descriptive result.