THE UNIVERSITY OF AUCKLAND
Department of Electrical and Computer Engineering

RESEARCH PROJECT (SOFTWARE ENGINEERING) REPORT

SEMESTER 1, 2016

MAORI PRONUNCIATION AID USING THE CLOUD

Written and Submitted by: Yuxiao Zhai
Student ID: 370790312

Supervised by: Dr. Catherine Watson

DECLARATION OF ORIGINALITY

This report is my own work and was not copied from or written in collaboration with any other person.

Yuxiao Zhai
Submitted on: 27/06/2016

Maori Pronunciation Aid Using the Cloud

Yuxiao Zhai

Department of Electrical and Computer Engineering

Faculty of Engineering

The University of Auckland

Auckland, New Zealand

yzha727@aucklanduni.ac.nz

***Abstract*—In order to help Māori language learners to improve their pronunciations, this project focuses on building a web application called MPAi which allows users to not only listen to canonical pronunciations of Māori words, but also record and analyze their own pronunciations. In the end, this application was successfully developed in a traditional software engineering approach called waterfall. Consequently, learners can use it to practice Māori language pronunciation on web browsers.**

***Index Terms*—MPAi, Maori Language, Language Aid, Speech Recognition, Cloud**

I. Introduction

Nowadays, the New Zealand government has been striving to increase the number of people able to speak Māori language. As R. a. Benton [1] mentioned, Māori language was recognized as an official language in 1987. A Māori Language Commission was also set up to advise the government to maintain and revive the language. Hundreds of Māori-medium pre-schools were built, and more and more primary schools and secondary schools have Māori language as a subject. Besides, information technologies, such as computer and internet, are also playing an important role in protecting and promoting the language.

In order to assist Māori language learners in improving their pronunciations, MPAi project, whose full name is 'Māori Pronunciation Aid', was set up. There are several prior projects about MPAi, producing different versions of MPAi software. One of the first versions was implemented by J. King and R. Harlow [2], where the users can record their own speech of some common Māori words using a microphone, and then they can receive the feedback of their efforts, such as 'VERY GOOD', 'GOOD', 'AVERAGE', 'POOR', and VERY POOR'. Another MPAi project produced a formant plot. The software allows users to record and analyze their vowel pronunciation by showing the real-time feedback on their vowels in a formant chart. The most recently MPAi project integrated the word recognizer and the formant plot into a complete PC-based MPAi application, which will be discussed in the section of related work. Now, the objective of this project is to develop a web application that allows users to not only listen to the canonical pronunciations of Māori words, but also record and analyze their own pronunciations through web browsers.

To this end, the project followed a standard software engineering process, and it has successfully fulfilled the major tasks within the process. Software requirements analysis is the first step, in which we confirmed that web-based MPAi is a tool, running on web browsers, whose users are Māori language learners. The main functions are to analyze the pronunciations of Māori words from users and provide canonical pronunciations for users to listen to. Then, we optimized some designs of the PC-based MPAi, and chose ASP.NET platform, with HTML, CSS, and JavaScript in front-end and C# in back end. We also chose JSON files to manage data instead of relational databases. Next up is developing and testing. In order to make sure the application delivers the right functionality, we made a demo first for each function, and then completed the whole implementation with testing. In this stage, we reused the core class called 'HTKEngine' from the PC-based MPAi project to implement the speech recognition function, and we also invoked a JavaScript API named 'MediaStreamRecorder' to record through browsers. After that, the application was deployed on AWS cloud platform so that it can be visited on the internet.

In the following sections we will first review the related work. Then, we will present the entire development process in detail, from planning, requirements analysis to deployment. Eventually, we will discuss the evaluation, followed by the conclusion and future work.

II. Related Work

The project started with researching the related work in the field of computer-aided pronunciation training (CAPT).

A. The PC-based MPAi

As referred to earlier, there are several projects about MPAi, one of which is the PC-based MPAi. The project was mainly completed in February 2016. The recognizer, 'HTKEngine' class, reused in our project is from this project.

The major goals of this project are as following:

1) Integrate the previous components of MPAi, like word recognizer, formant plot and user management.

2) Improve the accuracy of the previous word recognizer.

The application was implemented in C#, with eight Windows batch files to invoke the HTK toolkit efficiently, which is embedded in it.

The main functionality is three-fold, namely 'Video Player', 'Recognizer', and 'Pronunciation Aid', as in Figure 1.

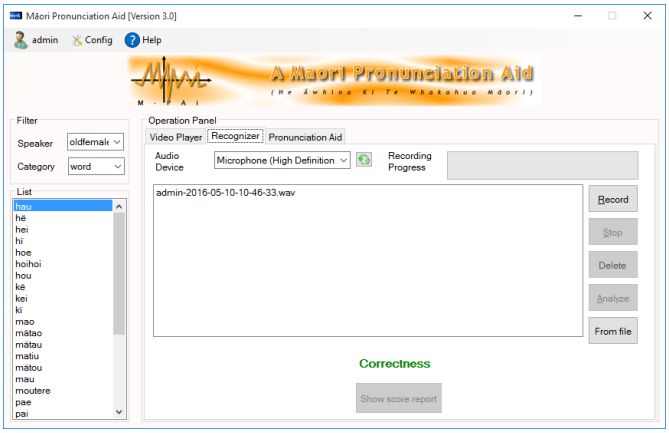


Fig. 1. The user interface of the PC-based MPAi.

The first feature is 'Video Player', which can play video recordings of the word if the word has already had any recordings uploaded to the database. The core feature is 'Recognizer', which allows users to record their pronunciations of Māori words and analyze them. This is the only feature that we need to transfer to online MPAi, so it will be discussed in detail later. The last feature is a formant plot, called 'Pronunciation Aid', which can provide a real-time feedback on their vowels through showing dots in the plot, as in Figure 2.

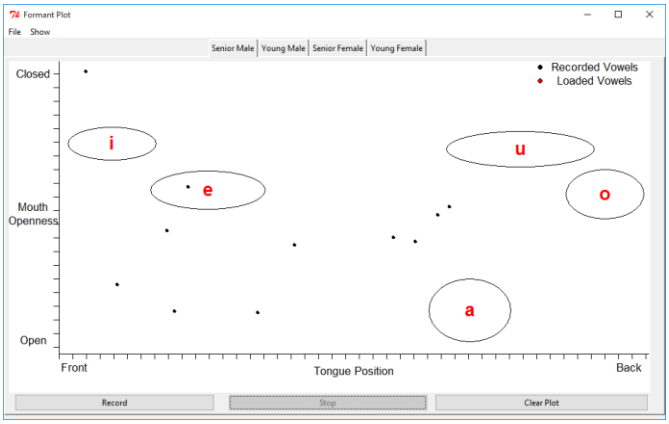


Fig. 2. The formant plot of the PC-based MPAi.

The PC-based MPAi has more powerful functions than online MPAi, while it also has some limitations as below:

1) Users can only select word from the 'List' on the left, which means that they cannot use it for all the Māori words.

2) The user role is ambiguous. For any user, if their pronunciation is recognized correctly, it will be added into the database of canonical pronunciations, which would decrease the accuracy of recognition.

3) As Figure 1 shows, users can double click words in the list to listen to their canonical pronunciations, but each time the pronunciation will be changed automatically, which means that users cannot listen to one pronunciation repeatedly even if they want.

4) As Figure 3 shows, the feedback is too complicated, and when it pops up, the recording is played automatically.

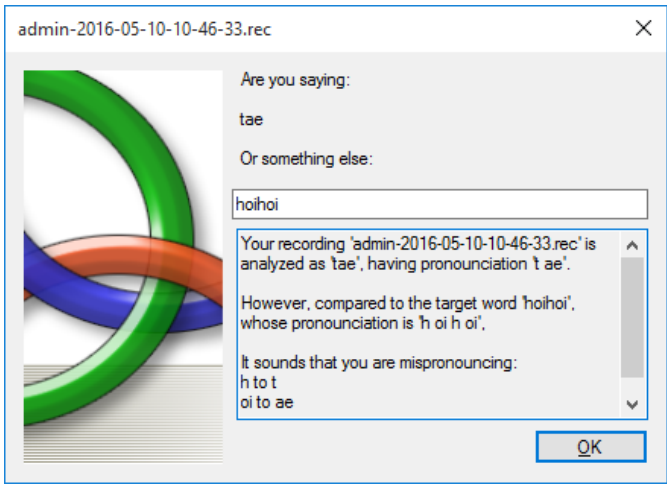


Fig. 3. The recognition feedback of the PC-based MPAi.

B. mENUNCIATE [4]

Unlike the PC-based MPAi, mENUNCIATE is a web-based CAPT system, which also has a prior PC-based version called 'ENUCLATE' [5]. There are some interesting similarities and differences between this project and online MPAi.

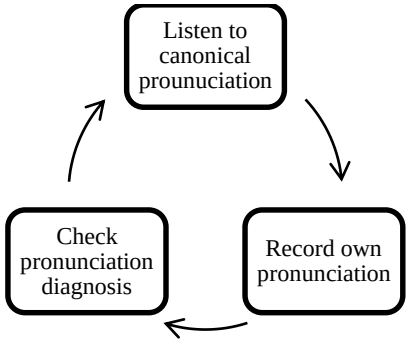


Fig. 4. The active pronunciation learning cycle.

The main similarities are in two terms, namely background and functionality. As for background, they share the same background [4] that mobile devices, including mobile phones and tablets, are more famous and influential than ever before. Therefore, there is a need to spread the PC-based applications to mobile platforms, and the best choice is web application. The second similarity is functionality. They both provide listening,

recording, and analyzing features to foreign language learners, which forms an active pronunciation learning cycle, as in Figure 4. For example, Figure 5 shows a practice of the analysing feature, where mENUNCIATE detected that the phone /r/ was missed and the phone /z/ was mispronounced as /s/ [4].

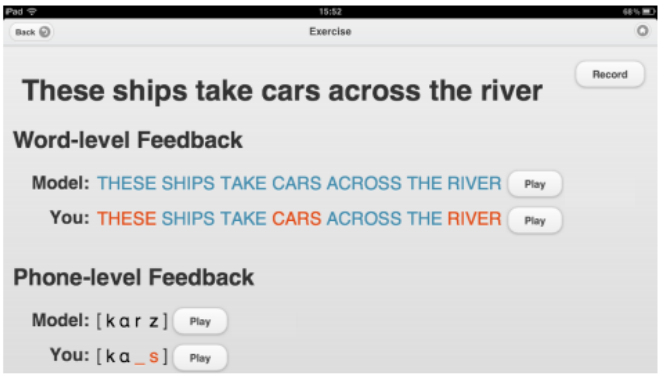


Fig. 5. The user interface of mENUNCIATE.

The main difference exists in the front-end architecture. Unlike the website-formed MPAi, P. Liu [4] developed different versions of mobile applications to support different mobile platforms. Mobile applications can do things websites can't, like pushing notifications and offline access, but they also have drawbacks. For example, users must download and install them on their own mobile devices, which is more time-consuming and storage-consuming than just opening the browsers mostly pre-installed. And obviously, the applications cannot run on Linux, Mac, or Windows 7 and earlier versions. Additionally, developers have set up the applications and submit them to the different application stores, which is much more complicated than hosting websites with a domain.

C. *Duolingo [6]*

Duolingo is a free language learning platform which contains a language-learning website (https://www.duolingo.com/) and mobile applications running on iOS, Android and Windows Phone mobile operating systems. This language learning platform has two main advantages. Firstly, it offers more than 50 different language courses across 23 languages, with over 120 million registered users all over the world. Then, this platform can make it as fun to learn languages, through speaking, listening, translation, and multiple choice challenges.

Duolingo is far more sophisticated than our project, and points out a valuable direction for our future work, especially its speaking feature. As Figure 6 shows, when the web page is completely loaded, Duolingo will play the recording of a sentence automatically. Users need to speak this sentence while pressing the microphone icon, and then they can check their own recording or skip. The result shown in Figure 7 contains two parts. The first part is on its sentence itself showing in different colours, with green for the correct and red for the wrong. In Figure 7, the word 'is' was wrongly pronounced while the others were all done correctly. The second part is at the bottom of the window, where there are a summary of the recording and its translation in users' mother languages. In this case, the summary illustrated as mainly correct, with the translation of this sentence in Chinese. This web application combines listening, speaking and translation together, which can offer users more information to practice their speaking and master a sentence.

Fig. 6. The speaking user interface of Duolingo.

Fig. 7. The result of users' recordings in Duolingo.

III. PLANNING

In this project, a traditional software engineering approach, 'waterfall' [7, 8], was employed, with the rationales as following:

1) This is a small and short project where the requirements are very well understood and fixed.

2) The product definition (a Web-based tool) is stable.

3) The technology (mainly HTML, CSS, JavaScript, C#, ASP.NET) we optimised is understood.

The waterfall model is simple and easy to use, with which the project became not so hard to manage, and each phase had precise deliverables and did not overlap with other phase.

Before we start the first phase, requirements analysis, we made a project plan, adopting five valuable software project management techniques to ensure the success of this project.

A. Project Stakeholders

In order to get a big picture of this project, we analysed the project stakeholders and their interests in the project first, as below:

1) Māori Language learners. They are end users who ultimately use online MPAi, and can improve their Māori words pronunciations through this web application.

2) Catherine Watson. She is my supervisor, the project client and sponsor, who owns and directs the project, which can help her to achieve the strategic objective in speech related research.

3) The PC-based MPAi developers. They can help us to understand the requirement and provide some technical support, including the recognizer, 'HTKEngine' class. They can also get some feedback from this project to improve the quality of the PC-based MPAi.

4) API developers. In our project we need to invoke an API so as to make audio recordings through browsers. They could also benefit from our feedback.

5) Online testers. They contain other MPAi related researchers and students. They offer to test the web application so that it can be refined in usability.

B. Project Critical Success Criteria and Factors

The project critical success criteria and factors play a decisive role in software projects.

As to the project critical success criteria, because this is a research project, we did not consider the business benefits, but we did consider the budget. The criteria are as below:

1) Complete on time, within budget.

2) Implement all the software requirements specification with high quality.

3) Ensure the satisfaction of the stakeholders strategically so that it can be involved in the entire speech related research project.

Accordingly, the project critical success factors are as following:

1) Communication amongst all stakeholders.

2) Clear requirement definition.

3) Good understanding of user's needs.

4) Adequate project manager competency.

C. Communication Plan

As recognized before, communication would be the most important project success factor.

In order to ensure that all the stakeholders are kept involved actively, we made the communication plan, both verbal and written, based on the analysis of stakeholders. To be specific, we were going to have weekly regular face-to-face meetings on Wednesdays with Catherine Watson and two other students supervised by her. We should also have some temporary meetings if necessary. Besides, we should send an email report to our supervisor weekly to account the progress, problems encountered and the support or resources needed. In addition, we should contact the API author directly if need be.

D. Work Breakdown Structure (WBS)

According to the principles of project management and traditional software life cycle [8], the project is divided into six major stages: requirements analysis, design, implementation, testing, deployment and evaluation.

The WBS is shown in Figure 8.

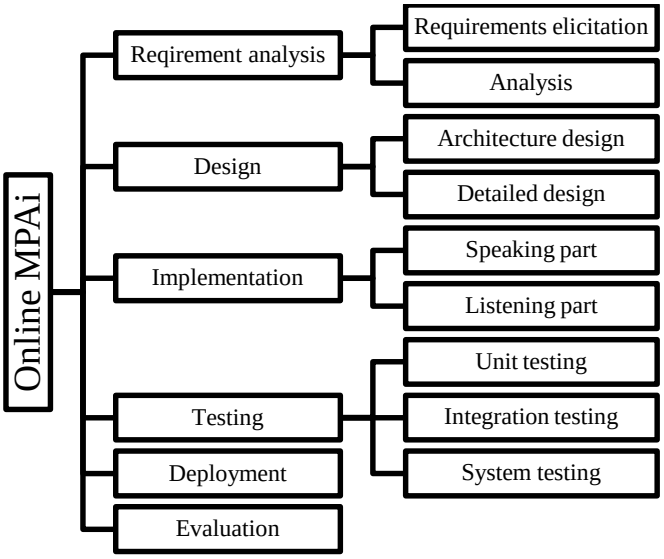


Fig. 8. The WBS of online MPAi.

E. Gantt Chart

Gantt Chart [10] is a tool to illustrate and manage a project schedule. In order to make a Gantt Chart, we analyzed every task in WBS and estimated the respective duration basing on its difficulty and the resource which can be allocated.

Just like the name of 'waterfall' model, the Gantt Chart of this project is simple, without any tasks overlaped, shown as Appendix.

IV. Requirements Analysis

From this section, we start to present the six major developing phases in detail. This section focuses on the first phase, requirements analysis, describing how we did requirements elicitation, conducted the analysis, and arrived at the specified requirements

There are three major channels to collect the requirements, Catherine Watson, the PC-based MPAi application and its developers. Firstly, we talked to our supervisor, because she has done the Māori language related researches for quite a long time, and knows the requirements very well. After that, we got the scope of online MPAi. What we need to implement is just a web-based tool, with only one user role, Māori language learners, and it does not account users' data. Then, we had a discussion about this with the author of the PC-based MPAi, achieving some detailed understanding of each feature. In addition, I also installed and tried the application of the PC-based MPAi, trying to consider the requirements at Māori language learners' point of view. During this process, we also prototyped the user interface to explore and confirm the

requirements. We did not do a large-scale survey or interview with end users, because this is a continuation project of the PC-based MPAi, whose requirements have been clear and fixed, and there is not enough time to do it.

After being collected, analyzed, and communicated with stakeholders, the requirements were specified, and separated into two groups, functional requirements and non-functional requirements.

A. *Functional Requirements*

As the user case diagram Figure 9 shows, this application contains two main features, listening and speaking, which are also two critical aspects of learning a language.

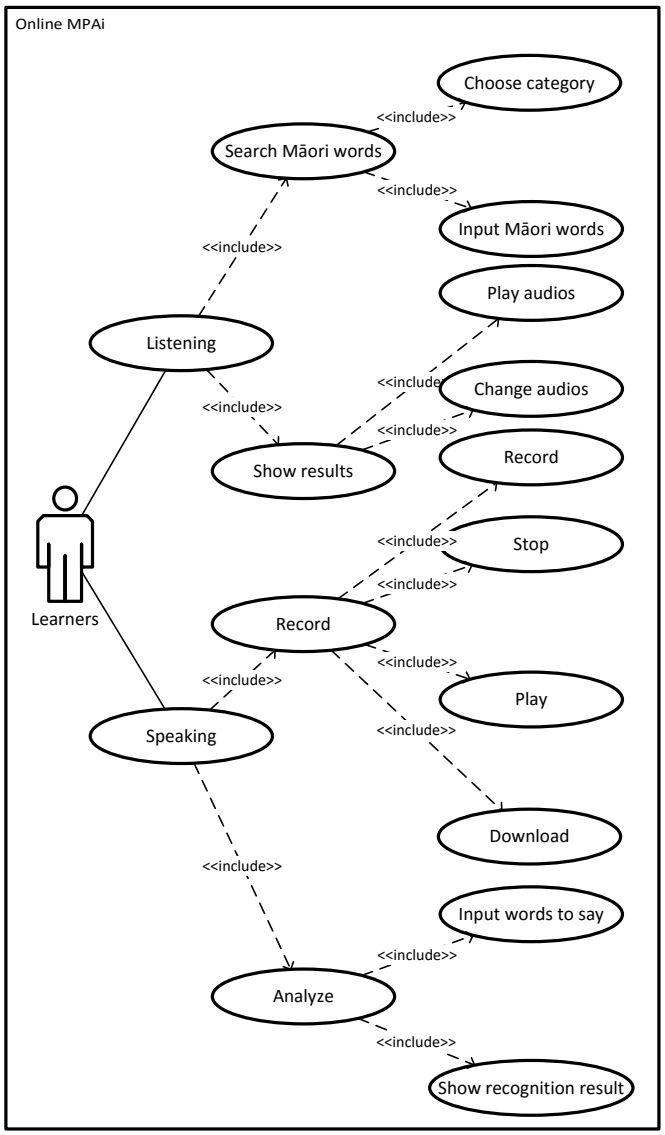


Fig. 9. The user case diagram of online MPAi.

In listening, users can search a word after choosing a category and inputting the word itself, then all the pronunciations of this word in this category will be presented as a result set. Users can play them repeatedly or change a different pronunciation of the result set.

In speaking, users can record their own pronunciation of word and analyze the accuracy, then the application will make a speech recognition and show the result. Users can contrast the word they said and the word which the application recognized to find out their pronunciation issues. And, they can also download their own pronunciation locally to their machines.

B. *Non-functional Requirements*

The non-functional requirements mainly contain the hardware and software environment that the application server should support.

1) Hardware Requirements (minimum)

- Processor: 64-bit 2 cores required, 2.0 GHz
- RAM: 2G required
- Hard drive space: 35 GB available

2) Software Requirements

- Operating System: Windows Server 2012 R2 64-bit Standard or greater.
- Microsoft .NET Framework Version: 4.5
- Web: IIS 8.0 or greater with ASP.NET

V. Design

After the confirmation of requirements specification from our supervisor, we moved into the design phase. In this phase, different diagrams were employed to model the system, in both static and dynamic views, which would show the high level architecture design and detailed design. During this process, the improvements to the PC-based MPAi were made, and the related alternatives were also discussed.

A. *Design Models*

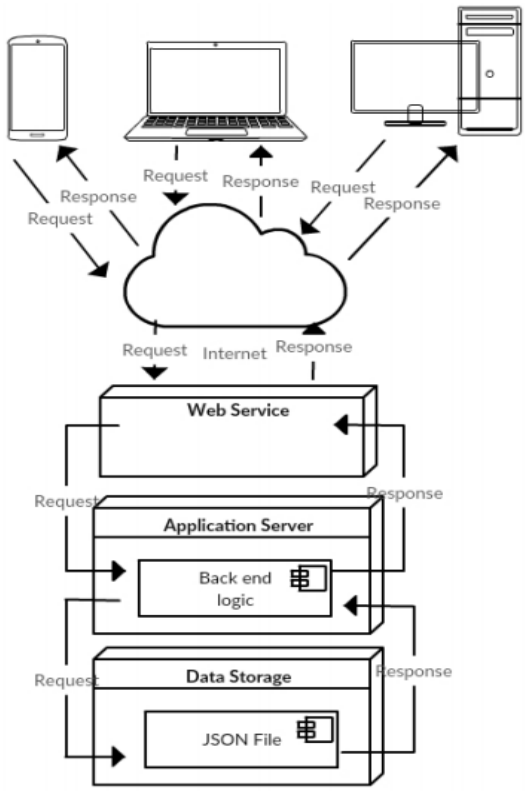


Fig. 10. The architecture of online MPAi.

This web application does not contain too many complicated components, and most of features can be realized by browser event handlers in JavaScript, and consequently we choose

functional programming approach [10], which means the main features were realized by composing functions. As a result, the architecture (shown in Figure 10) also documents the distribution of web application components.

In Figure 10, users can visit online MPAi through browsers no matter whether it is on mobile phones, laptops or computers. When they make an operation on browsers, the JavaScript event handler will be invoked on their local devices, and if necessary, an HTTP will be then sent through internet into the server side, where the back end logic will be executed. After that, the result will be sent back to users' local devices through an HTTP response, and presented on their browsers. If there is a need to read data from storage, for example in listen feature, the back end logic will collaborate with the JSON file rather than a relational database. In this version of online MPAi, there is no operations writing data into storage.

This is the high level architecture design, and then we used UML activity diagrams to illustrate the detailed operation in each feature dynamically, with Figure 11 for the Speak feature, and Figure 13 for the Listen feature.

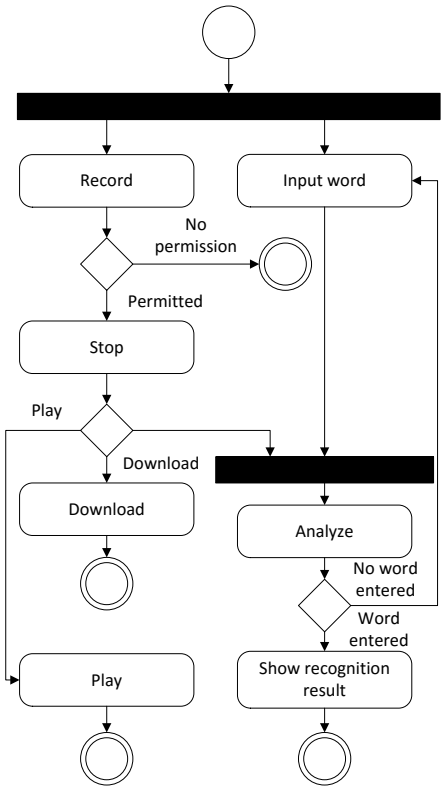


Fig. 11. The UML activity diagram of the Speak feature in online MPAi.

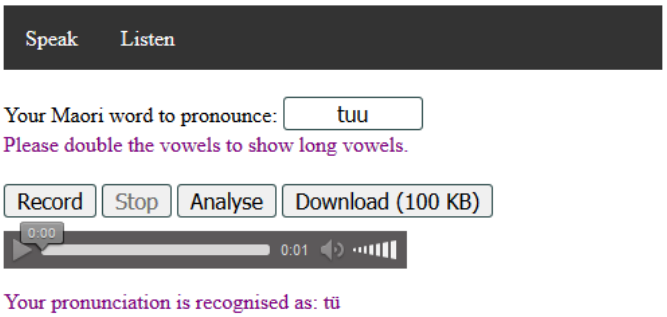


Fig. 12. The user interface of the Speak feature in online MPAi.

In Figure 11, when users choose the speak feature, they need to input the word they are going to say and make a recording of their own pronunciation. When they record their pronunciations, they must give the audio permission to the website, otherwise they cannot make a recording. Then, they can play their recording through a standard HTML5 audio bar, or download it to their local devices. If they choose to analyze their pronunciation, online MPAi will check whether their word has been entered first. When the check is passed, the recognition result will be shown, so that users can make a contrast between the word they said and the word recognized by the application. Figure 12 is the user interface, corresponding to this feature.

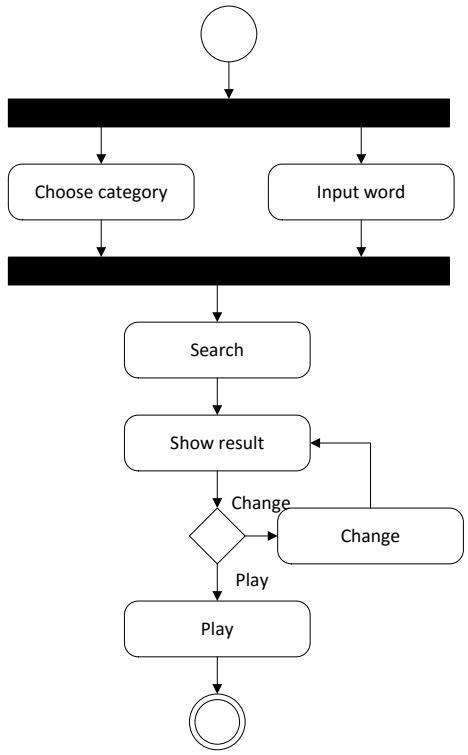


Fig. 13. The UML activity diagram of the Listen feature in online MPAi.

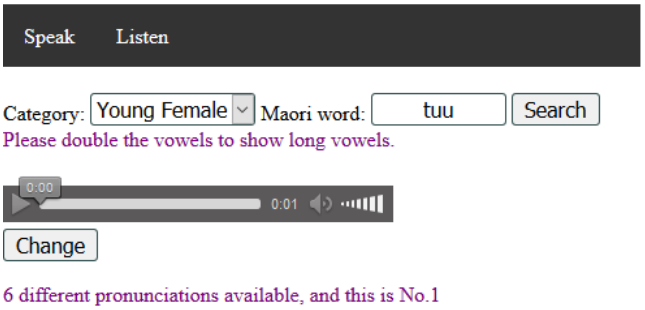


Fig. 14. The user interface of the Listen feature in online MPAi.

Figure 13 demonstrates the UML activity diagram of the Listen feature. At first, users need to choose the pronunciation category (including 'Young Female', 'Old Female', 'Young Male', and 'Old Male'). For example, if users choose 'Young Female' in cateogry, the searched result will be the recordings of young female speakers. Next, users need to input the word they want to listen to, and then they can search it. After the result is shown, they can play it, or change to another recording

of the same word but a different speaker. The corresponding user interface is shown in Figure 14.

B. Design Improvement to the PC-based MPAi

As we mentioned in Section II, there are some limitations in the PC-based MPAi. In this project, all of them were overcome to a certain extent, as following:

1) In the PC-based MPAi, users can only select word from the 'List' on the left, which means that when there are a lot of Māori words pre-saved in the system it would become very hard for users to find a precise word. That may lead to bad usability. In online MPAi, we use searching instead of choosing. The benefit of this design is that users can find every Māori word easily as long as it is in this application, which improved the usability significantly.

2) In the PC-based MPAi, for any user, if their pronunciation is recognized correctly, it will be added into the database of canonical pronunciations, which would decrease the accuracy of recognition. In this application, users' pronunciations are not inserted into the canonical set, and it does not provide any pronunciation collecting feature.

3) In the PC-based MPAi, users can double click words in the list to listen to their canonical pronunciations, but each time the pronunciation will be changed automatically, which means that users cannot listen to one pronunciation repeatedly even if they want. In online MPAi, we added a button call 'change' so that users have right to choose to change a new pronunciation or listen to the current one repeatedly, which is beneficial to imitation.

4) The feedback was simplified as well. As seen in Figure 12, the application only shows the recognized word out, and users can contrast them by themselves. Besides, the recording can also be played manually rather than automatically.

C. Discussion of Alternatives

In design phase, we made two important decisions in choosing the technical platforms:

- Which recording API should we invoke?
- Which kind of Windows Server platform should we use?

As to the first question, we deselected flash and any techniques using plug-ins first, because of their poor compatibility with mobile platform. Then, we narrowed our choices to two APIs, Web Audio API [11] and MediaStreamRecorder [12]. By contrasting them in terms of functionality, compatibility and usability, we finally chose the latter with the rationale as below:

1) Both are developed in JavaScript and can make audio recordings on web browsers.

2) Both are compatible with major browsers, Chrome, Firefox, Opera and Microsoft Edge, and even work on Android browsers.

3) Although the former may be more powerful, the latter is more usable for new users. As M. F. Zibran [13] mentioned, clear and up-to-date documentation and examples of usage increase the usability of APIs. Firstly, MediaStreamRecorder provides an audio recording demo with source code [11], while the Web Audio API does not. Secondly, the functions exposed in MediaStreamRecorder is just enough, with good quantity and encapsulation. In contrast, Web Audio API provides too many properties and methods, which is not easy for new users to learn and use.

This is about the API selection, and there were also some alternatives while we were choosing Windows Server platform. Because the PC-based MPAi is implemented in Microsoft .NET platform, we did not consider other server-side operating system. And then, we finally chose Windows Server on Amazon AWS [14] over on-premise and Microsoft Azure [15]. The rationale was based on the non-functional requirements, in three aspects:

1) Security. This web application need secure firewalls against hackers. The three all support firewalls, with 'Security Groups' [16] on AWS and 'Network Security Group' [17] on Azure.

2) Cost. For on-premise, the total cost of ownership (TCO) is paid by the department, however it would cause a big unnecessary waste with only online MPAi itself running on one physical server. Fortunately, both AWS and Azure provide free trials, for 12 months (limited but enough services) and one month respectively.

3) Deployment. They all support the traditional deployment, while AWS and Azure also support continuous deployment, which would accelerate testing and deployment.

VI. Implementation

The web application was developed in ASP.NET platform, with HTML, CSS, and JavaScript programming languages in front end and C# programming language in back end. The communication between front end and back end is built through ASP.NET web services. In this section, we will introduce the programming languages, platform, and data storage used in this project, and in addition, the strategies for reuse of existing products and components will also be discussed.

A. Front End Programming Languages

HTML, CSS, and JavaScript are the most popular programming languages in front end. They work together to make dynamic web pages which can be used in most browsers without any extensions or plug-ins installed.

HTML [18], with full name Hypertext Markup Language [19], specify the content of web pages using HTML tags, for example paragraph tags, table tags, etc. CSS, or Cascading Style Sheet [19] specify how the content of HTML is presented, the layout of the web page and the details such as colors, fonts, margins, and so on.

HTML and CSS can make static web pages. If users expect dynamic, interactive web pages, JavaScript is essential. JavaScript provides a way to add programming to web pages so that web pages get behaviors, for example giving users an alert, updating data without communicating with back end, etc.

When web applications are executed, the process is comprised of two steps, parsing web pages and event-driven invoking [18]. During the web application loading, the browser parses the HTML tags and set up a document object model (DOM). Then the DOM is presented on the screen based on CSS. Then the browser parses JavaScript codes mostly to do the initialization and register functions, some of which are event

handlers. This is the first step, parsing. After that, once some events occur (such as mouse clicks or window resize), the event handlers will be triggered, manipulating the DOM elements based on the events, and then the web pages will be updated accordingly.

B. Back End Programming Language and Implementation Platform

In the back end, we chose C# programming language with ASP.NET platform.

For the programming language, the best choice is C#, not only because it is an expressive and simple programming language but also because the PC-based MPAi was developed in C#, which means that if we choose C#, we can easily reuse the components from the PC-based MPAi.

As for ASP.NET, the most significant advantage is that it is easy to create and access a web service without writing infrastructure code to handle details such as communication protocols [20]. With web services, we can exchange messages between front end and back end.

C. Data Storage

In this project, the only interaction with data storage is reading file path of canonical pronunciations from data storage so that they can be played on web browsers. In this situation, JSON files were chosen as data storage, with the reasons as following:

1) For this project, it is sufficient to use JSON files instead of relational databases, because there are not complicated queries.

2) After benchmarking, the search speed is acceptable, although it is not sure whether searching in JSON files is faster than in relational databases.

3) Using relational databases may cost for the software license, or cloud services, while using JSON files is for free.

4) It is more complicated to install, set up and maintain relational databases than JSON files [21].

D. Strategies for Reuse

Strategies for reuse played great role in the implementation phase. There are two important reuse, audio recording API, 'MediaStreamRecorder' and speech recognition class, 'HTKEngine'. As we introduced before, 'MediaStreamRecorder' API is written in JavaScript, through which we can realize features of recording, uploading, and downloading an 'WAV' audio file on web browsers. Besides, 'HTKEngine' is the main function of this class is to get an 'WAV' file and return the recognition result by invoking the tools in 'HTK' (the Hidden Markov Model Toolkit [22]).

When reusing the components, we paid attention to the three aspects, as below:

1) Functionality. First, we ran the examples to verify that the target components to reuse can satisfy corresponding functional specifications.

2) Documentation. Before we invoked the components, we had read the documentation carefully. For 'HTKEngine' class, there is no documentation, and we read its source code.

3) Communication. When we encountered problems, we tried to communicate with the authors, which was also expected in the analysis of stakeholders. Figure 15 demonstrates the communication record with Muaz Khan [23], the author of 'MediaStreamRecorder' API.

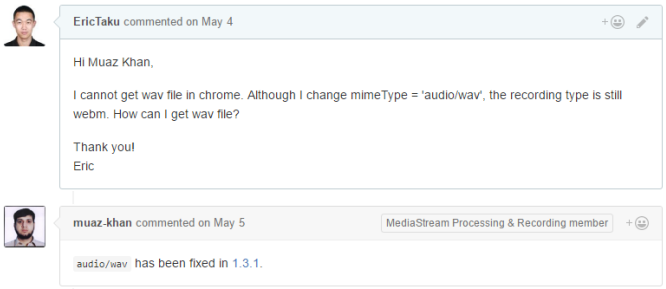


Fig. 15. The communication record with Muaz Khan, the author of 'MediaStreamRecorder' API, in Github.

VII. Testing and Deployment

This section will present how we did the testing and deployment.

A. Testing

In testing phase, we utilized unit testing, integration testing and system testing. We did unit testing and integration testing while we were developing the features by ourselves. Therefore, there was an integration phase at the end of implementation phase or at the beginning of testing phase. With this integration phase, we proved that most of the test cases can be executed correctly. And then, we delivered the system testing to online testers after we completed the implementation phase.

B. Deployment

After testing, we deployed this web application on Amazon Elastic Compute Cloud (EC2) [24]. The application is accessible via the URL http://ec2-54-206-89-39.ap-southeast-2.compute.amazonaws.com/. (Please access it on Firefox or Microsoft Edge, because Google Chrome needs an SSL certificate, which is pending approval from department.)

There is more than one way to deploy a Windows Server-based application in AWS. EC2 was chosen, because it can be used to create a virtual server, called an EC2 instance, to run the web application. Just like running in a remote server, administrators can access the EC2 instance over Remote Desktop Protocol (RDP). This made the deployment easy to master.

As we discussed before, AWS has a security component called security groups [25], which acts as a firewall. The security groups control the traffic permitted to arrive at EC2 instances. When an instance is launched, one or more security groups will be associated to the instance, and then security rules will be added into each security group to allow traffic related to its associated instances. In this project, one security group was created with following rules:

1) Allow inbound HTTP access from anywhere.

2) Allow inbound RDP traffic from the developing computer's public IP address.

VIII. Evaluation

From the software engineering's point of view, this project is successful completed. Firstly, it completed on time, within

budget. Secondly, it implemented all the requirements specification including listening to canonical pronunciations and recording and analyzing users' own pronunciations. Thirdly, it was thought acceptable by major stakeholders.

However, because the time and resource are both limited, this web application also has some limitations, as following:

1) Users cannot make recordings through any browser on iOS mobile platforms, because the audio permission is restricted by Apple, and only native applications can get the permission.

2) Users can only get hundreds of canonical pronunciations to listen to. That's because the dataset of canonical pronunciations is not complete.

3) On Microsoft Edge, if users want to play their own recordings, they have to download them first, because the audio bar on Edge cannot play the 'WAV' recordings recorded by itself. As a result, we replaced the audio bar with a label confirmation, as in Figure 16.

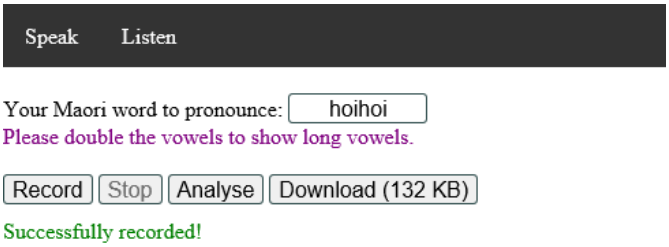


Fig. 16. The user interface of Speak feature on Microsoft Edge.

IX. Conclusion and Future Work

In conclusion, this project achieved the object of making an online MPAi, with listening, recording and analyzing features. This web application was developed in a traditional software life cycle, with good project management. Finally, it was deployed on AWS cloud platform.

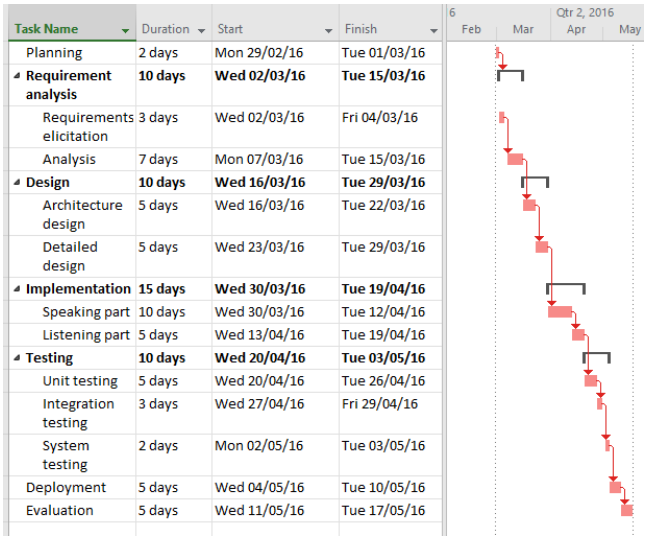
Based on this project, there are some aspects about online MPAi can be improved in the future:

1) More canonical pronunciations need to be added so that users can search more words. Besides, these canonical pronunciations can also be used to refine the accuracy of recognition.

2) The user roles can be expanded from learners to teachers and pronunciation contributors, with identity management built. Learners can manage their own recording and listening record, and teachers can see the statistics of a group of learners. In addition, MPAi can also start to collect canonical pronunciations from contributors, after both being verified.

3) For learners, the user interface can be redesigned, combining Speak and Listen features together, so that learners can complete the active pronunciation learning cycle (in Figure 4) in the same page. To be specific, learners can first listen to the canonical pronunciation and then practice and analyze their own pronunciation, or they can record and analyze first, and then listen to the canonical pronunciation if necessary.

Appendix



Task Name	Duration	Start	Finish
Planning	2 days	Mon 29/02/16	Tue 01/03/16
Requirement analysis	**10 days**	**Wed 02/03/16**	**Tue 15/03/16**
Requirements elicitation	3 days	Wed 02/03/16	Fri 04/03/16
Analysis	7 days	Mon 07/03/16	Tue 15/03/16
Design	**10 days**	**Wed 16/03/16**	**Tue 29/03/16**
Architecture design	5 days	Wed 16/03/16	Tue 22/03/16
Detailed design	5 days	Wed 23/03/16	Tue 29/03/16
Implementation	**15 days**	**Wed 30/03/16**	**Tue 19/04/16**
Speaking part	10 days	Wed 30/03/16	Tue 12/04/16
Listening part	5 days	Wed 13/04/16	Tue 19/04/16
Testing	**10 days**	**Wed 20/04/16**	**Tue 03/05/16**
Unit testing	5 days	Wed 20/04/16	Tue 26/04/16
Integration testing	3 days	Wed 27/04/16	Fri 29/04/16
System testing	2 days	Mon 02/05/16	Tue 03/05/16
Deployment	5 days	Wed 04/05/16	Tue 10/05/16
Evaluation	5 days	Wed 11/05/16	Tue 17/05/16

Acknowledgment

We are deeply grateful to Catherine Watson, Shawn, Ahmed, Muaz Khan, online testers, and all stakeholders who are involved in and support this project. Special thanks to our supervisor, Catherine Watson, whose guidance and kindnesses made this project an enjoyable journey.

References

[1] R. a. Benton, "Perfecting the partnership: revitalising the Māori language in New Zealand education and society 1987–2014," Lang. Cult. Curric., vol. 28, no. 2, pp. 99–112, 2015. http://www.tandfonline.com/doi/full/10.1080/07908318.2015.1025001

[2] J. King, R. Harlow, C. Watson, P. Keegan, and M. Maclagan, "Changing Pronunciation of the Māori Language Implications for Revitalization," Indigenous Language Revitalization: Encouragement, Guidance & Lessons Learned. pp. 85–96, 2009. http://jan.ucc.nau.edu/~jar/ILR/ILR-7.pdf

[3] YsqEvilmax/MPAid: Maori Pronunciation Aid from University of Auckland. Retrieved from https://github.com/YsqEvilmax/MPAid

[4] P. Liu, K. W. Yuen, W. K. Leung, and H. Meng, "mENUNCIATE: Development of a computer-aided pronunciation training system on a cross-platform framework for mobile, speech-enabled application development," 2012 8th Int. Symp. Chinese Spok. Lang. Process. ISCSLP 2012, pp. 170–173, 2012. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6423507

[5] K. W. Yuen, W. K. Leung, P. F. Liu, K. H. Wong, X. J. Qian, W. K. Lo, and H. Meng, "Enunciate: An internet-accessible computer-aided pronunciation training system and related user evaluations," *2011 Int. Conf. Speech Database Assessments, Orient. COCOSDA 2011 - Proc.*, pp. 85–90, 2011. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6085985&tag=1

[6] Duolingo: Learn Spanish, French and other languages for free. Retrieved from https://www.duolingo.com/

[7] A. Aitken and V. Ilango, "A Comparative Analysis of Traditional Software Engineering and Agile Software Development," *2013 46th Hawaii Int. Conf. Syst. Sci.*, pp. 4751–4760, 2013. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6480417

[8] T. Stober and U. Hansmann, *Agile Software Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. pp. 15-33 http://link.springer.com/10.1007/978-3-540-70832-2

[9] J. M. Wilson, "Gantt charts: A centenary appreciation," *Eur. J. Oper. Res.*, vol. 149, no. 2, pp. 430–437, 2003. http://www.sciencedirect.com/science/article/pii/S0377221702007695

[10] K. Hinsen, "The promises of functional programming," *Comput. Sci. Eng.*, vol. 11, no. 4, pp. 86–90, Jul. 2009.

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5076325
[11] Web Audio API: Web Audio API – Web APIs | MDN. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
[12] Streamproc/MediaStreamRecorder. Retrieved from https://github.com/streamproc/MediaStreamRecorder
[13] M. F. Zibran, F. Z. Eishita, and C. K. Roy, "Useful, But Usable? Factors Affecting the Usability of APIs," in *2011 18th Working Conference on Reverse Engineering*, 2011, no. Table II, pp. 151–155. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6079520
[14] Amazon Web Services (AWS) – Cloud Computing Services. Retrieved from https://aws.amazon.com/
[15] Microsoft Azure: Cloud Computing Platform and Services. Retrieved from https://azure.microsoft.com/en-gb/
[16] Security Groups for Your VPC – Amazon Virtual Private Cloud. Retrieved from http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html#VPCSecurityGroups
[17] What is a Network Security Group (NSG) – Microsoft Azure. Retrieved from https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/
[18] H. Park, W. Jung, and S. Moon, "Javascript ahead-of-time compilation for embedded web platform," in *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, 2015, pp. 1–9.http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7351768
[19] A. Butowsky, K. Gai, M. Coakley, M. Qiu, and C. C. Tappert, "City of White Plains Parking App: Case Study of a Smart City Web Application," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, 2015, pp. 278–282. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7371494
[20] Introduction to Programming Web Services in Managed Code. Retrieved from https://msdn.microsoft.com/en-us/library/yzbxwf53(v=vs.140).aspx
[21] A. Adamanskiy and A. Denisov, "EJDB - Embedded JSON Database Engine," in *2013 Fourth World Congress on Software Engineering*, 2013, pp. 161–164. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6754279
[22] HTK Speech Recognition Toolkit. Retrieved from http://htk.eng.cam.ac.uk/
[23] Record audio/wav in chrome #81 - streamproc/MediaStreamRecorder. Retrieved from https://github.com/streamproc/MediaStreamRecorder/issues/81
[24] Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS. Retrieved from https://aws.amazon.com/ec2/
[25] Amazon EC2 Security Groups for Windows Instances - Amazon Elastic Compute Cloud. Retrieved from http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/using-network-security.html