TATAI

A Māori language learning tool.

Nathan Cairns
Department of Electrical and Computer Engineering
University of Auckland
New Zealand
ncai762@aucklanduni.ac.nz

***Abstract*—This report outlines the development of Tātai, a Māori language learning tool. Tātai is primarily designed for a child user, aged 7-10, who is a first language speaker of Māori, and is looking to improve their math skills. This report covers the software development and design processes. Various design decisions and the reasoning behind them will be expanded upon. The various software packages used, and implementation design decisions will also be covered. The final design is provided and is accompanied by a feature overview. After such is an outline of the usability, covering GUI layout, colour considerations, user diversity and error handling. This will be followed by a discussion on design limitations. Evaluations on the project from both a self and peer review will then be discussed. This will include comments on what was changed / kept from them. Following this will be a discussion of possible future work. The report will be closed with conclusions on the project.**

Keywords— Māori; language; software development; useribility; evaluation;

I. Introduction

This project was assigned as the main part of the Software Engineering Design I course. For this project, *Buster Darragh-Major* and I were contracted to design an application to aid Māori language learners.

Tātai was to feature real-time feedback on being able to do basic math, with the answers given in spoken Māori. The application was also required to feature a well-designed graphical user interface (Please refer to ***Figure 1*** for an example of a GUI). As a design decision we decided to primarily target children aged 7-10 who are already Māori language speakers, and are looking to improve their math skills.

The application was required to meet certain functionality requirements. The main game mode of the application was to display randomized mathematical equations on the screen and have users speak their answer into the microphone. The answers of these equations were required to be an integer between 1 and 99. The application was then to use audio recognition software to test the pronunciation of the user's answer.

Furthermore, the application was required to meet certain other requirements. These included

- A way of tracking a user's statistics including a record of their best score.
- Help on how to use different features
- A practice module, where a user can practice saying Māori numbers
- Speech playback so users can hear their attempt
- Some form of reward to motivate the user.
- A scoring mechanism, including notifications of personal bests, and overall highest scores

The functionality above was to be constructed using the Java programing language with support from some Bash commands. The main tools used in the application were JavaFX for GUI construction and HTK for audio recognition. There were also some supporting libraries used. These included JFoenix for material design, FontAwesomeFX for iconography and Jackson for storing user's and their statistics.

Tātai was to be built incrementally, from four different assignments and a final project. This was intended to emulate the processes used by developers in industry software construction.

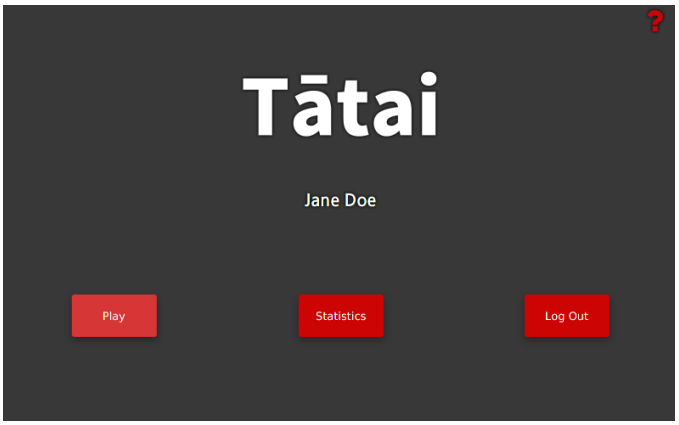

Figure 1: The homepage of Tātai

II. Software Development Method

A. General process

The main part of this project was to be executed in groups of two. A small team resulted in the use of a very light weight and agile development environment. Whilst we did not adhere to a specific agile methodology we did follow the general concept of iterative development. We did this to ensure we produced high quality software.

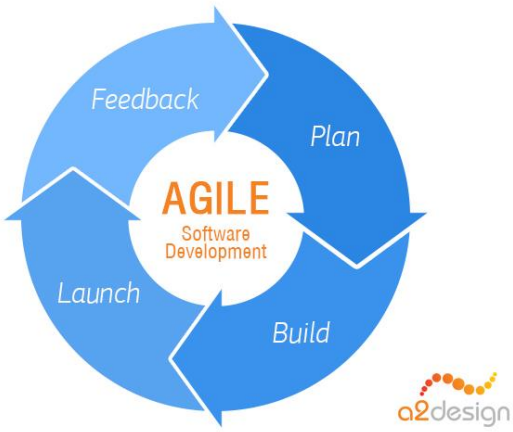

Figure 2: Agile software development process [7]

The general process we used followed the plan, build, launch, feedback methodology (Please refer to ***Figure 2***). For each stage we would have regular developer meetings where we discussed design and planned what we needed to implement. We would then implement these features and integrate them into the project. This was complimented by regular client meetings where we would get appropriate feedback. The process would then start again.

B. Client feedback

Client Feedback was an important part of our process. The first major feedback we got was at the end of Assignment 3. Here we had to present an early prototype to a course supervisor. After this presentation we had regular in-lecture design meetings. In these meetings lecturers acted as clients and we were given the opportunity to ask questions to clarify specifications. These meetings were very helpful and gave us something to discuss in our developer meetings. We also had our program tested and evaluated by our peers. These peer reviews were very detailed and provided us with very useful feedback to work with for the later part of this project.

The precise nature of this feedback and how we decided to handle it will be covered later in the **evaluation and testing** section of this report.

C. Version control

Version control was an essential component of our development process. For version control we used git & GitHub. We made sure to make frequent commits each one containing a single change. This was done to ensure we had a comprehensive and easy to navigate project history. It also helped us adhere to the principle of continuous integration.

We also made sure to have clear communication between developers. We made sure each of us knew what parts of the program the other was working on. This was done to avoid significant merge conflicts and breaking the build.

D. Documentation

Documentation was another important part of our process. Both developers made sure to keep design journals. These journals contained notes on meetings both between developers and clients.

These journals were kept so that we both had written documentation containing requirements, design decisions and job allocations. This was very helpful as it meant we both had reference material to refer to during the implementation process. It also provided us with a project history to refer to for report writing.

III. Design decisions and product design

Requirements were given for Tātai which required several design decisions to be made. These design decisions were made to best achieve the required features, functionality and usability specified. These design decisions are outlined in the following paragraphs.

A. Target user

The target user we choose for Tātai was children between ages 7-10, who are first language speakers of Te Reo, and will be using Tātai to improve their math skills. This audience was chosen for many reasons.

Primarily, we choose this audience as we felt that they would be the most likely to use and engage with this product. i.e. we felt that the program as specified was either too simple or too irrelevant to the other two audiences we could have chosen. These were both older audiences more interested in pronunciation than having their math tested. An equation game mode was significantly emphasized in the specifications, which does indeed put more emphasis on math, rather than pronunciation. Tātai is intended to be used in a classroom environment.

This decision of a target audience had many implications on the design of Tātai, as detailed below.

The colour scheme we selected was heavily inspired by the colours seen on the Māori flag (Please refer to ***Figure 3*** for a picture of the Māori flag and ***Figure 1*** to see how the colours were used in Tātai). We choose these colours as they provide an established and familiar palette for our target user. We also went for light pastel colours and a material design look and feel. This was done to make the GUI have a modern look and feel, and to avoid user fatigue which can be brought about from using too intense colours. We also made sure the game modes were as colourful as possible, with a random new colour scheme being generated with each new question. This was done with the intention of making the game modes more engaging.

Furthermore, we made the interface as approachable as possible for children. We did this by making buttons responsive and obviously clickable. We also made a very clear path through the application. We did this by making buttons disappear when you're not meant to click them. We also made sure the application was resizable. The ability to go full screen is beneficial as it focusses the user's attention on the application. i.e. the child won't be distracted by the desktop or other processes running on the computer.

Furthermore, we added extra game modes to provide variability to the user. This included a reverse mode where the user gets numbers in Te Reo and must provide the English translation. This was done to allow a user to try another game mode if they get bored of the standard mode. It also allows Tātai to be used by people who want to increase their proficiency at reading and understanding Māori.

We also provided an interactive statistics window where a user can view how well they have been doing and what skill level they have. This was done to motivate the user to improve their skills in the various game modes.

These various features are covered in further depth in the **TATAi-final product and overview** section of this report.

Figure 3: The national Māori flag [8]

B. Software packages used

The software packages used for Tātai, were selected carefully, and with consideration of the client's requirements. The software packages used in the final product were ultimately chosen to ensure a high-quality product.

The primary programming language used for Tātai was Java. My partner and I selected Java as it was the programming language we were both most familiar, and proficient with. Java is also a very robust and object-oriented language. This provides a high-level abstraction of software design. It also allows for good extensibility when good object-oriented practices are followed. We made sure to take advantage of these language features in the creation of Tātai.

Furthermore, Java also provides support for easily calling some bash commands from your Java application. This was quite an important feature as the voice recognition software we were provided with along with some other functionality required bash shell calls to use.

For voice recognition we were provided with a software package created by our lecturer *Catherine Watson*. This package was created using the Hidden Markov Model Toolkit (HTK). This recognition software contained models for recognizing Māori numbers between 1 and 99. We used Java's process builder to call the relevant command from our application.

For GUI design and operation, we used Java's JavaFX framework. JavaFX is a powerful and flexible open source, GUI construction, Java-based framework. We decided to use JavaFX as we wanted to make an application using modern technologies and frameworks. The other Java-based option would be swing, however, we both agreed that this framework was very dated, and it would be much easier and practical to use JavaFX. JavaFX is good for many reasons. Firstly, it provides an option to use a markup language (FXML) to describe the layout of the GUI. This is good as it separates the logic for building the GUI from the business logic. It also allows you to view graphical changes without recompiling code. It also provides a more transparent view of the scene graph. Along with this, JavaFX allows for the use of Scene Builder, a GUI for constructing views which generates changes to corresponding FXML files. (Please refer to ***Figure 4***).

We also used several third-party libraries to help meet client requirements.

We used Jackson as a method of storing user data. Jackson is an efficient Java-based library which can be used to map or serialize Java objects to Java Script Object Notation (or JSON). This library was extremely helpful as it allowed us to easily convert users and their statistics to JSON and store them locally (object serialization). JSON is also a very popular way of sending data over the internet. This opens the program for possible internet connectivity in the future. It also provides a platform for implementing a web API with ease.

We used two external libraries to assist with GUI design. These were JFoenix and FontAwesomeFX. JFoenix is a material design library for JavaFX. We choose to use JFoenix as material design, if used right, provides very simple and modern looking GUI design. FontAwesoneFX is a Font Awesome JavaFX plugin. FontAwesomeFX provides an easy method for inserting scalable vector Font Awesome icons into FXML files. This was used for iconography throughout the project.

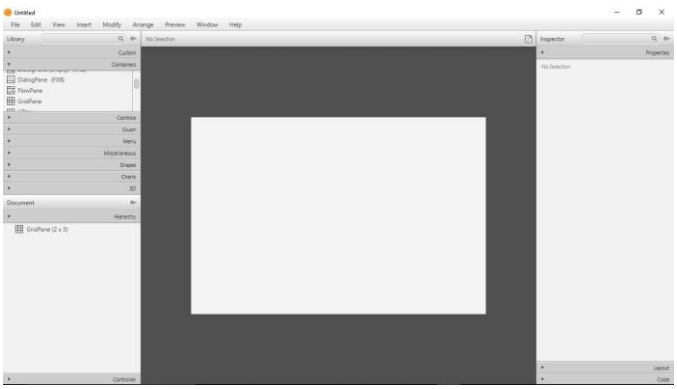

Figure 4: Scene Builder GUI

IV. TATAI-FINAL PRODUCT AND OVERVIEW

A. Overview

Tātai met the specifications set out by the project brief. Tātai has several main functionalities. These included an equation game mode, which generates random equations with answers between 1 and 99; the ability to create your own question lists; help on how to use various features; a practice module; speech playback and some form of scoring mechanism / reward system. These features along with some extra functionality that was implemented will be discussed below.

Please refer to the Tātai user manual for a more in-depth analysis of the various features and how to use them.

B. Game modes

We implemented 4 game modes for a user to play with. (Please refer to ***Figure 5*** & ***Figure 6*** to see how the game modes are shown to the user)

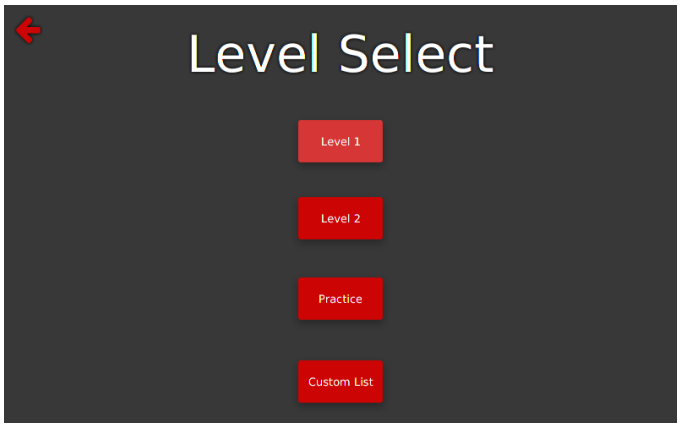

Figure 5: Tātai level selection window

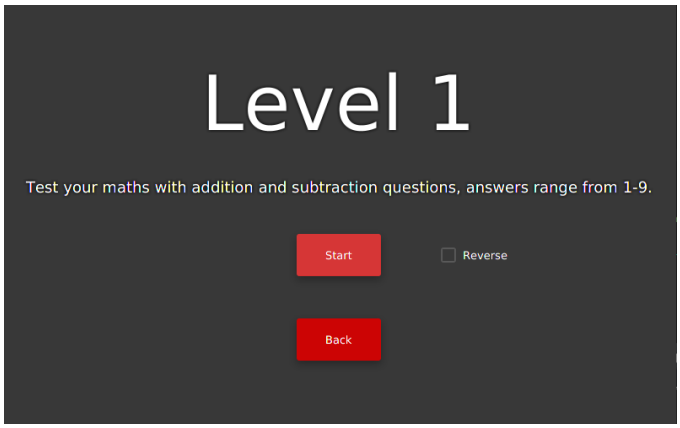

Figure 6: Tātai level selection confirmation window

1) Equation game: This game mode was one of the required functionalities of Tātai. In this game mode a user is presented with an equation and is asked to give the answer in Te Reo. Each game is 10 questions long. The user has the ability to listen to their attempt before submission. The user also gets two attempts at each question. There are two levels level 1 and level 2. Level 1 is the easy level with addition and subtraction equations with answers ranging between 1-9. Level 2 is the hard level and remains locked until the user scores 8 or more in level 1. Level 2 contains addition, subtraction, multiplication and division questions. Answers range between 1-99. (Please refer to **Figure 7**, **Figure 8**, **Figure 9**, **Figure 10** for an example of the various screens a user will experience in the equation game mode.

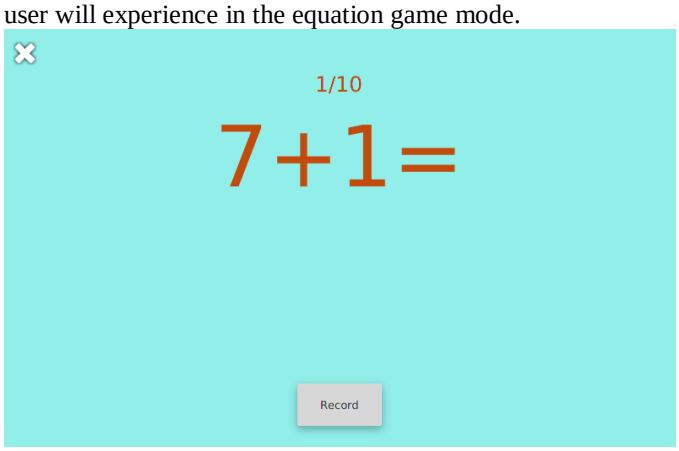

Figure 7: Equation game mode pre-recording

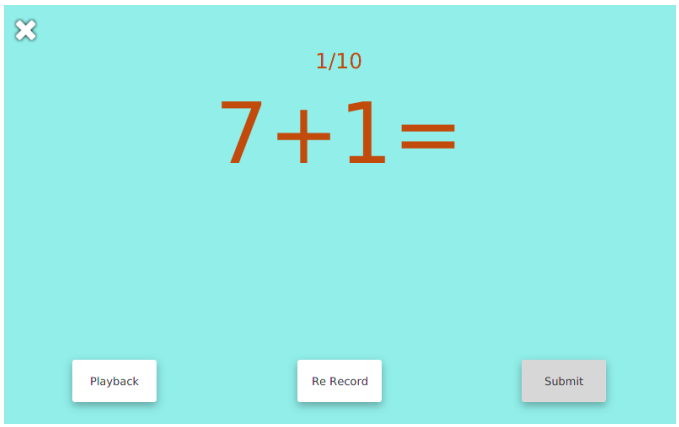

Figure 8: Equation mode pre-submission

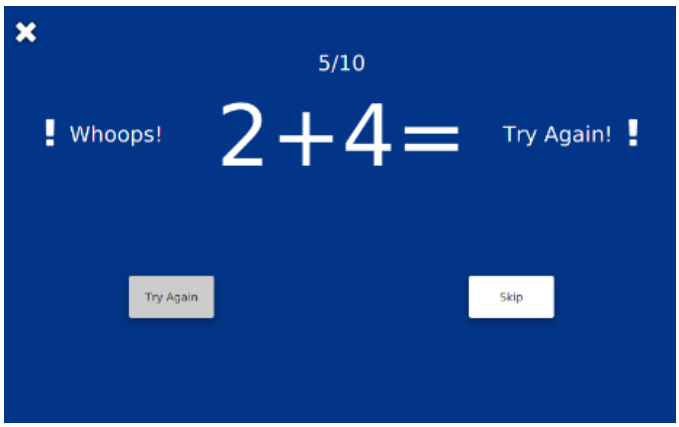

Figure 9: Level 1 Equation mode try again (first attempt incorrect)

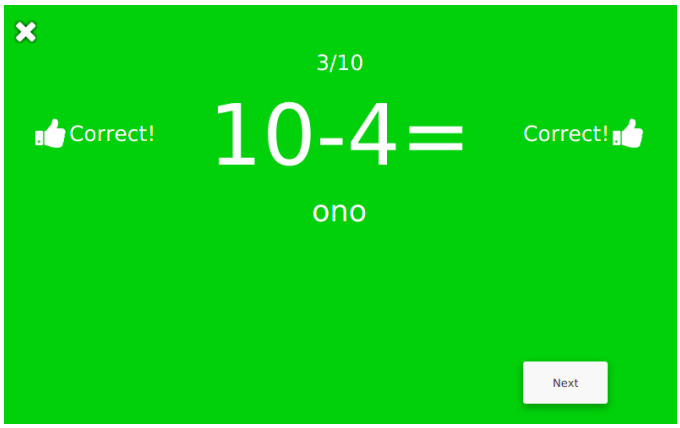

Figure 10: Equation mode correct

2) Reverse game: This game mode was one of the pieces of extra functionality we provided. In this game mode a user is presented with a Māori number, and has to input the translation into the provided keyboard. The option to enter the reverse game mode is seen in the level confirmation window (see **Figure 6**). Just like the equation game mode there are two levels. Level 1 has a range of numbers 1-9 and level 2 has a range of 1-99. To unlock level 2 the user has to get a score of 8 or more in level 2 of the equation mode. This game mode was included to provide variability to the user and appeal to a wider audience. i.e. someone who wanted to learn to read Māori numbers could use this mode. See **Figure 11** for a demonstration of the reverse game mode.

Figure 11: Reverse game mode

3) Custom list: This game mode allows users to use custom lists defined by Teachers. There are no special requirements to use this game mode. This was part of the required functionality. Custom lists can only contain equations (addition, subtraction, multiplication and/or division) with answers between 1-99.

4) Practice module: This game mode allows users to test their pronunciation of Māori numbers. There is no score kept. A user can choose what numbers they want to practice (1-99). This mode goes on for as long as user wants it to. See **Figure 12** for a demonstration of the practices module.

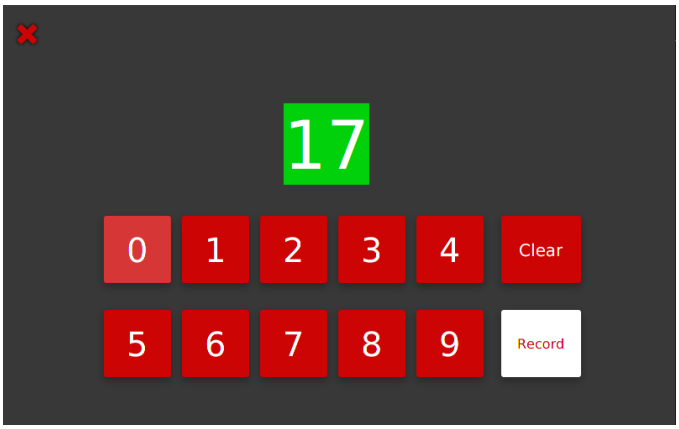

Figure 12: Practice Module

C. *Statistics & skill levels*

1) Statistics: Statistics were a major required functionality of Tātai. We decided to implement a comprehensive statistic tracking system for each individual user. For each individual user their *Personal Best, Total Played, Total Correct, Total Incorrect and Average* for both levels of equation mode and reverse mode are stored. These statistics provide a simple but effective analysis of how the user is doing in each level. Our goal here was to keep it simple and easy to understand for our target audience. We also made sure to supply these statistics in an interesting and interactive way. See the Tātai user manual for a further description of the stats menu and its interactive components. See **Figure 13** for a demonstration of the statistics window.

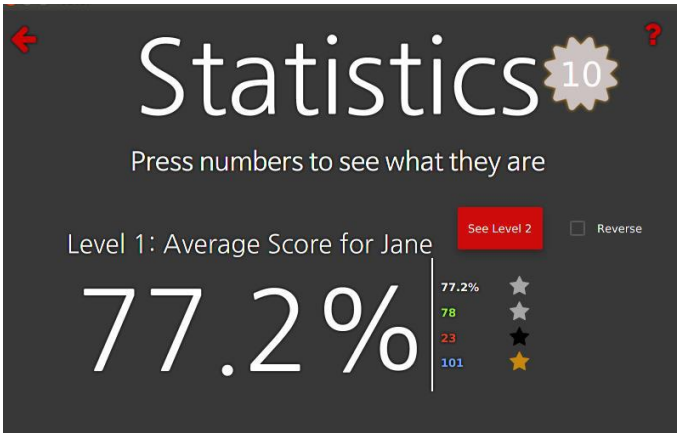

Figure 13: Statistics Window

2) Skill levels: To accompany our statistic tracking system we provided skill levels. Depending on a user's score for each statistic they are rewarded a skill level. The skill levels are the following: None, Bronze, Silver, Gold and Platinum. The better a user's score is, the higher the skill level. This is the reward system we chose to implement as per the specifications. Skill levels are intended to motivate the user to play more games and increase their skill levels. We included this system as it is important to provide some form of motivation to keep a younger audience interested. The stars

next to each statistic in **Figure 13** represent the skill levels for each statistic. The colours of the stars change to correspond with the user's skill level. Please refer to the Tātai user manual for more on skill levels and what conditions users must meet to earn specific levels.

D. Users

Users are another essential part of Tātai. We chose to focus a lot on this feature. Each user has their statistics tracked individually and are saved between sessions in JSON format in txt files. There are two types of user, teachers and students. ***Figure 14*** shows the user selection screen where the user can choose who to login as. The user can also delete and create new users. We choose this system as it reflects one of our goals which was for Tātai to be used in a classroom.

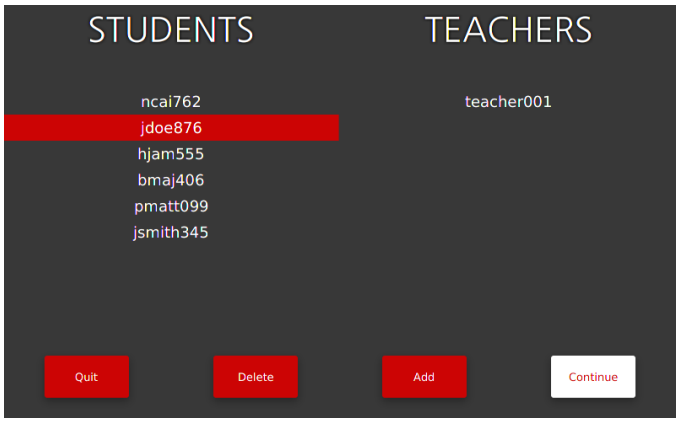

Figure 14: User Selection

1) Students: These are the main users who the application is designed for. These users experience the most "game-like" experience. Levels start off locked for students and they have skill levels assigned to each one of their statistics.

2) Teachers: These users are suppose to manage the students using the application. Teachers have access to special functionality. For example, a teacher can create custom lists for other users to use. See **Figure 15** for an example of custom list selection. Teacher's also have access to all game modes straight away. Furthermore, teachers do not get skill levels. This is because teachers are expected to already know the content and be administrators rather than players.

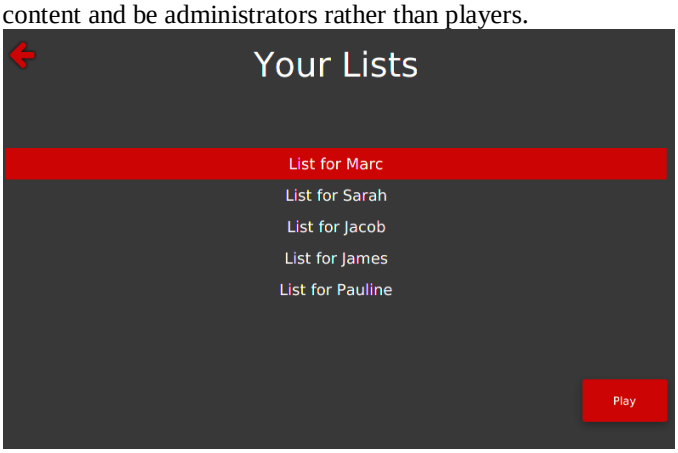

Figure 15: Custom list selection

V. USABILITY AND USER INTEFACE DESIGN

The user interface for Tātai was designed with user interaction in mind. We tried to make an extremely coherent and natural user experience. To achieve this, we followed a well-planned design process. This started off with high level abstractions such as screen diagrams. These abstractions were then used to develop cohesive GUI components.

The decision to make Tātai as user friendly as possible came from our choice of target audience. A software application for children must be both robust and easy to use.

A. User workflow

Computer-human interaction is vital to any application targeted towards a younger audience. Children like to click all over the screen. This can be problematic as it can ruin the flow of the program. To avoid this issue, we made sure to restrict user input as much as possible.

For example, instead of having the user type in input from the keyboard for the reverse game mode. We provided an on-screen keyboard with key bindings set to numeric keys. This made sure that a user could not input erroneous data as an answer. See ***Figure 11*** for an example of the on-screen keyboard.

Another example of where we did this was in the normal equation mode. Buttons that we do not wish the user to press are hidden when they are not needed. For example, a user does not have the ability to submit answer until they make a recording.

Another way we have aided user workflow is by providing various key bindings and auto focusing. This provides a much faster way of navigating the program for someone who is not interested in using the mouse.

A good example of where we used this is the equation game mode. At each stage of the game the button focus is redirected to the button we would most like the user the press. This means a user can play the equation game mode using just the mouse bar.

For a full list of key binds and other convenience features please refer to the Tātai user's manual.

B. Layout and GUI planning

We did a lot of careful planning before implementing our GUI. This started off with a meeting involving a white board and numerous screen diagrams. When we had chosen a design we both created copies in our design journals and set about created the GUI. The GUI was one of the first things we did using stubs and dummy data to emulate an actual session.

This process was important as it allowed us to have a clear design which we could use. It also meant that we could ensure that our design was well suited to our target audience.

C. Colour

The colour palette was carefully selected to properly match our target audience whilst remaining simple and approachable to others.

We chose light pastel backgrounds to avoid overwhelming the user. On top of this we layered strong contrasting colours. We chose to make our program colourful to make it more engaging to our target audience. This included having varying colours during the primary game modes. See ***Figure 16*** for an example of this.

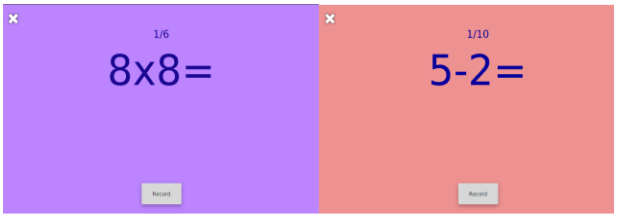

Figure 16: Changing colours in equation game mode

D. User diversity

Considering user diversity, we decided to implement various convenience methods for more familiar users to use when navigating the GUI. This was mainly in the form of key bindings and auto focusing, outlined in the **User workflow** section above.

We also tried to capitalize on user familiarity with the use of easy to recognize icons. For example, in the reverse game mode (***Figure 11***) we used a speaker icon to hear playback of the Māori numbers. We also used a question mark to indicate places where a user could click for help. These icons already have well known and intuitive meanings. providing the user with a familiar experience.

We also accounted for potential users with disabilities, namely colour blindness. When the user receives feedback upon getting a question wrong / right we made sure that a label indicating the correctness of the answer was also displayed. i.e. the screen didn't just change colour to blue / green as this would not be easy to understand for someone who is colour blind. See ***Figure 9*** & ***Figure 10*** for examples of this accessibility feature.

E. User feedback

We took great care in implementing user feedback such that it provided a clear and informative look at how the user was doing.

In the main game modes, we chose to make the screen go either green or blue depending on whether a user got a question correct or incorrect. This was accompanied by a label clearly stating the correctness of the user's answer. We chose these colours as green and blue are clear feedback indicators. We chose to accompany this with text to make it clearer, we also did this for accessibility reasons (see the previous sub section, **User diversity**). We used blue rather than red, as we found red to be too harsh and discouraging for our target audience.

Furthermore, at the end of each game mode Tātai provides a comprehensive breakdown of the game just played. This is presented as a table which illustrates the questions that were answered. For each question the question number, the question itself, the correct answer, and whether the user got it correct or incorrect is displayed. This view is included as it gives the user a good look at how they did in the game they had just played. It also allows them to see where they went wrong and therefore where they need to improve. An example of post-game feedback can be seen in ***Figure 17***.

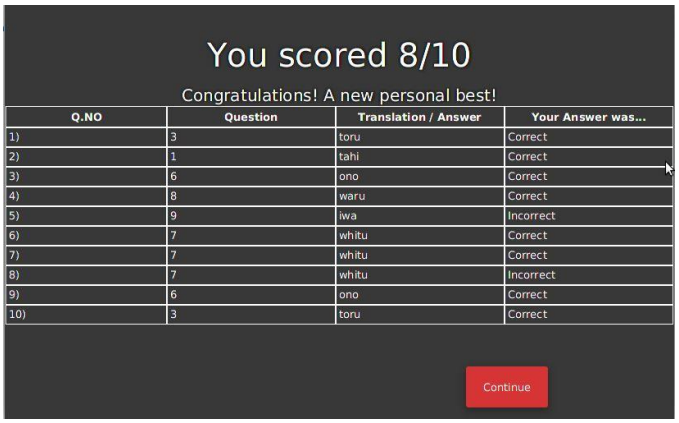

Figure 17: Game results table

We also choose to implement a simple but clear statistic tracking system. This statistic tracking system included the use of a "Personal Best" statistic. This statistic tracks the user's personal best over each game mode. On completion of a game a user is notified whether they have scored higher than their previous personal best. This is illustrated in ***Figure 17***. This is presented to a user only if they complete a game. This is intended to encourage a user to finish a game and see how they did rather than quitting mid-game. Please refer to the **TATAi-final product and overview** section or the Tātai user manual for further information on statistics.

F. Presentation of information and error handling

The presentation of Tātai was very important to us. We choose to display information such that it was as consistent as possible with our GUIs look and feel. We applied this methodology whenever possible. This included how information was presented to users and how functions were presented to the user.

For information presentation, we implemented various user help windows which were used to give the user vital information about the application. This included both a statistic skill level help window and a game tutorial window. Refer to ***Figure 18*** & ***Figure 19*** to see how these windows were designed to fit in with the rest of the GUI

We also implemented tooltips for various stages of the GUI. This mainly appears in the level selection windows where one can hover over a locked game mode to see how to unlock it. This was also implemented in the statistics window so that a user can clearly see what the stars next to their statistics mean. See ***Figure 20*** for a demonstration of this. Tooltips were chosen as an effort to avoid bloating the GUI with too much information. Tooltips also tie into the idea of creating an accessible and user diverse application as discussed in the previous sub section, **User diversity**. This is because it provides an alternative way for interpreting one's skill level if they suffer from colour blindness.

Figure 18: Skill level help screen

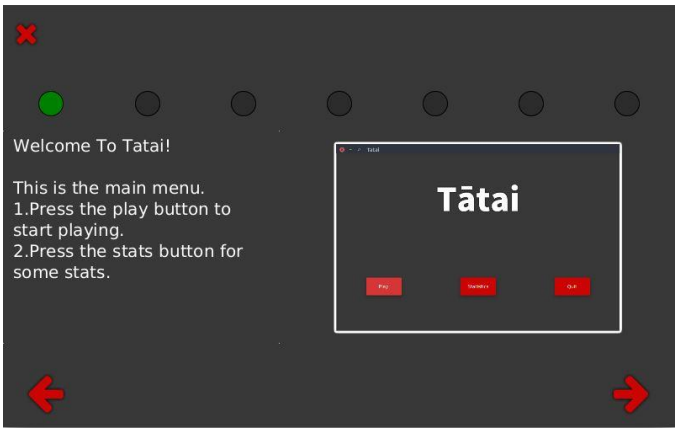

Figure 19: Game tutorial

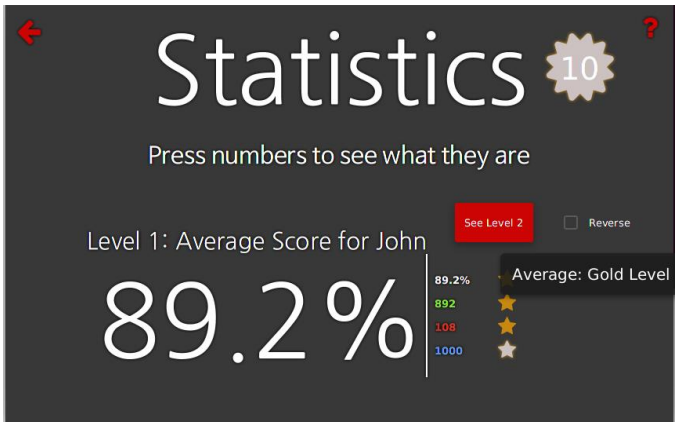

Figure 20: Tooltip demo

With error messages, we decided to make them as clear as possible. For example, if a user tries to create a new teacher/user with an already existing username an error message is displayed. This error message informs the user that a user with that username already exists. This clearly illustrates that the user must pick another non-existing username to create a valid teacher/student. Please refer to ***Figure 21*** for an example of this message.

Figure 21: Already existing user error message

For Tātai, we tried to implement the GUI such that errors could be avoided from happening whenever possible. This was done as a way of discouraging users from causing unexpected errors. Resultingly this, reduced the chance of user's having to fix and deal with errors.

This strategy included not allowing users to press certain buttons until the program was in a valid state. An example of this can be seen in ***Figure 11*** where the submit button is greyed out and disabled until the user inputs a valid answer to the question. Referring to ***Figure 11*** again, it is clear that a user has a finite range of valid inputs they can select from. i.e. letters, spaces and null characters are not allowed to be inputted as answers.

Furthermore, we decided to limit the amount of characters a user could input whenever user input was taken. For example, in the reverse game window (***Figure 11***) all questions have an answer in the range of 1-99. For this reason, we only allowed users to input answers of a maximum length of two. Another example of this is in the user creation form (***Figure 21***) where each text field has a character limit of 20. This is to ensure that no ridiculously long names are entered. This could affect the aesthetic and therefore user experience in the user selection view.

Lastly, we made sure that all function was labelled simply and clearly with appropriate icons and labels used for the various buttons. This was done to make sure that a user could clearly see how to complete the action they desired. This also allowed us to avoid cluttering the GUI with excessive descriptions of each function.

VI. DESIGN LIMITATIONS AND INTERFACE ISSUES

One large restriction on our program was the requirement that it must be written for a Linux platform. One of the key benefits of using Java is the Java Virtual Machine (JVM). The JVM allows programs to easily be written and run on different operating systems. This meant that writing an application specific for a Linux system was in direct conflict with one of the key benefits of using Java. Furthermore, a Linux distribution is not a common operating system for schools to

use. i.e. it is far more common to see Windows or Mac OS used in schools.

One other significant limitation of this project was the dictionary provided by Catherine's HTK software package. This dictionary was limited to recognizing Māori numbers between 1 – 99. The limited range of recognizable numbers meant that we were constrained to implementing features exclusively using numbers within that range. If we were given a more extensive dictionary or the ability to extend it ourselves, we could create a much better and more variable application. For example, we could have created game modes with more numbers or game modes which test non-numbers. However, the modularity & extensibility of our project does allow for these features to be easily implemented, with the addition of a larger dictionary.

VII. EVALUATION AND TESTING

A. *Self testing*

As a pair we decided to thoroughly test each aspect of our code. Through each iteration of the development process we implemented unit tests for various new features. This included unit tests for translators, label generation and custom question lists.

We also made sure to do significant testing using Catherine's Virtual Machine. This virtual machine is what Tātai was to be assessed on. We did this by playing through each feature and trying to break aspects of the program. This helped us ensure that Tātai was robust and ran smoothly on its required platform.

B. *Peer evaluation*

A beta submission was submitted to be reviewed by our peers. These peer reviews used the following criteria to assess and give feedback on the Tātai application.

1) *Code quality and software architecture*
2) *The quality of user feedback*
3) *What functionality was implemented*
4) *How well the program handled error*
5) *How well the GUI was designed, specifically for the target user.*

C. *Results of peer evaluation*

We received high-quality feedback from our peers. This aided us in adding several missing features and fixing some issues with the GUI and code.

This included:

1. Improving our user feedback system for visually impaired users. A common comment among our peer reviews was that our user feedback system was not clear enough. This was because we were previously just turning the screen green / red to tell the user whether they got an answer correct or incorrect. The issue our peers had with this was that a colour blind user would not be able to easily distinguish whether they got an answer right or wrong. To fix this we added a label to the feedback screen. This label informed the user whether they got an answer correct or incorrect. See ***Figure 9*** & ***Figure 10*** for an example of this.
2. Other GUI criticisms included issues with buttons and text sizes. Where text sizes were cited as an issue we made sure to increase the size and therefore the readability. For the size of the exit button in the statistics window we decided to replace it (and various other buttons) with Font Awesome icons. This improved the GUIs aesthetic and improved upon the clearness of the functionality of various buttons. We decided to not make buttons resizable with the window as it would require a decent amount of work for not much improvement. We believed this time could better be spent implementing important functionality, such as another game mode. Content such as the extra game mode was more important to implement before the deadline.
3. We also got criticized over our project structure and lack of separation between the view and controller components of our Model View Controller (MVC) design. To fix this we changed our project structure to match the maven standard directory layout [9]. See ***Figure 22*** for how we restructured our project to match the maven standard directory layout.

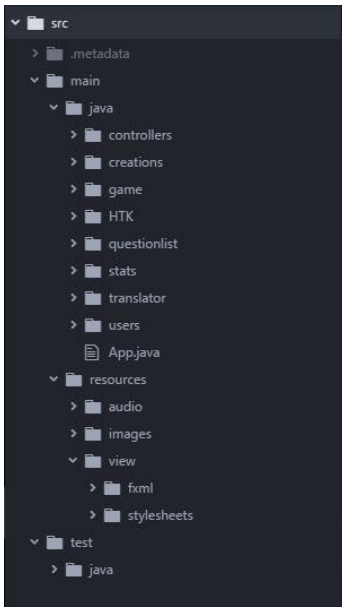

Figure 22: Tātai project structure

4. Another criticism was a lack of feedback to encourage user learning. To fix this we implemented the skill level feature as describe in the **TATAi-final product and overview** section above.

5. Our beta submission ended up having a bug on the Virtual Machine. This manifested in our buttons randomly jumping around the screen. It turned out this was due to us having dynamically resizing buttons when hovering over them. One peer review suggested we change the colours on hover rather than resizing. This issue only happened on the virtual machine, however, since this is the platform Tātai would be finally assessed on we decided to implement the suggested changes. This fixed the issue.
6. It was suggested that we have more in-depth statistics. To fix this we implemented statistics for each game mode, user-specific statistics, skill levels and a personal best statistic. Please refer to the **TATAi-final product and overview** section for further detail on these features.
7. One of our peer reviews claimed that turning the screen red was too harsh for our target audience. We discussed this and agreed so we decided to change the colour used to a dark blue. We thought that this still clearly conveyed that the answer was incorrect whilst being far less harsh. Refer to ***Figure 23*** to see how the feedback colours were changed.

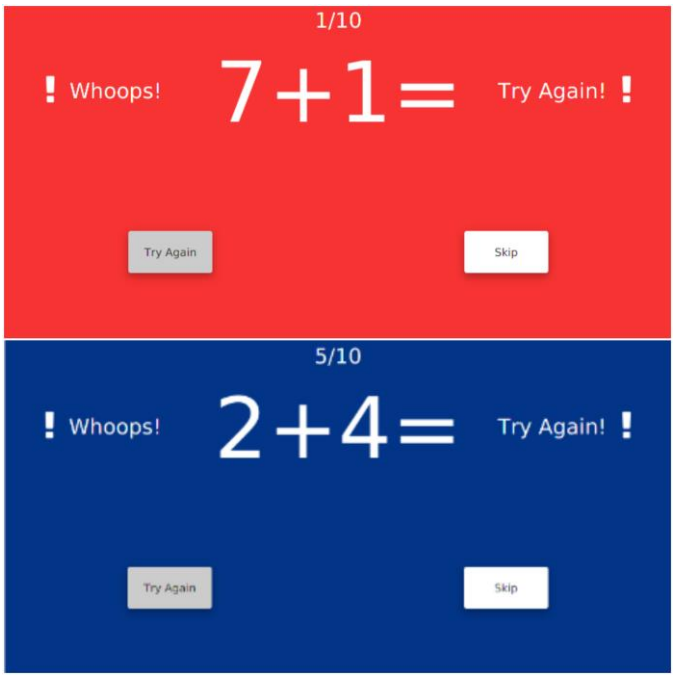

Figure 23: Updated feedback colours

8. Our peer reviews almost universally agreed that the most urgent fix to our program was the development of the Teacher's window. When we handed in the beta this feature was only partially completed. We prioritized finishing implementing this feature upon receiving our peer reviews.

VIII. FUTURE WORK

As mentioned previously in this report, Tātai was designed to be modular and extendable. This means that Tātai is ready and able to be expanded on in the future. It is important to note that the current implementation is a prototype.

Since we serialize our user objects in JSON it would be easy to send and receive user data between a server and the application. This would open the program to be able to connect multiple userbases. These userbases that be within schools or between schools. This would allow to further develop the game aspect of the application. It would also not be very difficult to build a leaderboard system on top of the software we have provided.

Furthermore, it is simple to extend Tātai with additional game modes or statistics for future releases.

Future work would include the implementation of features described above. It would also include further increasing the robustness of the platform. For example, making teacher users the only users with the privilege to delete and create other users.

IX. CONCLUSION

In conclusion, Tātai was an extremely educational, rewarding and overall successful project. Throughout the process I learnt and further developed many skills relating to software design and construction.

I learnt how to use multiple tools and design methods. This included the JavaFX java framework, scene builder, MVC, version control systems and various other java-based frameworks and libraries.

This project also aided me in further developing essential soft skills. This was mainly due to undergoing most of the project in a group. This helped develop my communication, planning and design skills. It also gave me good practice working in a small team and in an agile-like development environment.

Furthermore, I also developed client-communication and customer consideration skills. This was due to the lecturer's role as "clients" for the project and the requirement of designing the application for a specific target audience.

ACKNOWLEDGMENT

First and foremost, I would like to acknowledge my partner Buster Darragh-Major for the great teamwork, solo-work and cooperation he brought to our Tātai development team.

I would also like to thank the developers of JFoenix [10], Jackson [11] and FontAwesomeFX [12] for the invaluable contribution their respective libraries made to Tātai.

Lastly, I would like to thank the lectures / clients for tasking me to create Tātai. It was a wonderful and highly educational experience. I hope to be assigned projects of a similar scope in the future.

REFERENCES

[1] Why Should You Use Agile Software Development Methodology?. Retrived from http://www.a2design.biz/blog/why-should-you-use-agile-software-development-methodology/

[2] Ministry form Culture and Heritage. (2015). The National Māori flag. Retrieved from https://nzhistory.govt.nz/media/photo/national-maori-flag

[3] (2017). Introduction to the Standard Directory Layout. https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

[4] JFoenix. https://github.com/jfoenixadmin/JFoenix

[5] Jackson. https://github.com/FasterXML/jackson

[6] FontAwesomeFX. https://bitbucket.org/Jerady/fontawesomefx