# Tātai

## A Te Reo Maths-Leaning-Aid

Buster Major
Department of Software Engineering
University of Auckland
New Zealand
bmaj406@aucklanduni.ac.nz

***Abstract*—This report discusses the Tātai application developed over semester 2, 2017 at the University of Auckland. The application is a Māori context based math's learning aid, designed for 7-10 year old's who are first speakers of Te Reo, and want to improve their basic math's and practice arithmetic in the Māori language. The application contains voice recognition software which is able to recognize numbers pronounced by the user in response to mathematical equations shown on-screen, and returns feedback on their answer. The software development processes discussed include Extreme Programming (XP), modelling and version control. Also included are discussions surrounding the development of a suitable user interface for the target age bracket, what software frameworks and packages were used as well as a summary of some key features found in the application. Further discussion includes design constrains, methods of peer and client evaluation as well as how a high-quality software product was delivered through formal testing methods. Conclusions on the project and finished product are also made.**

***Keywords*—arithmetic; equarions; numbers; learning; Māori; Te Reo; software, user interface;**

## I. Introduction

For this project, I was assigned the task of creating a Māori language based math's learning aid for first language Te Reo language speakers. The application would contain voice recognitions software that would allow the user to pronounce Māori numbers 1 through 99 and receive feedback on their verbally communicated answer to arithmetic questions.

The project is significant because of the importance Te Reo holds for the cultural heritage of New Zealand. The language has been brought back from the brink of extinction over the past few decades, and recently huge importance has been placed on reviving the popularity and understanding of the culture. The Tātai application aims to create another way of supporting education through a Māori context, and allowing the next generation of Te Reo speakers to find interest in learning through their own language.

Tātai was to feature a game mode where users would be prompted with basic arithmetic questions of a pre-prescribed difficulty, and would speak the answer to the equation in Māori, where the software would then evaluate the users answer and display feedback. The target audience was chosen to be the 7-10 year old demographic, which meant designing a clean, simple and non-error prone graphical user interface, (refer to Figure 1).

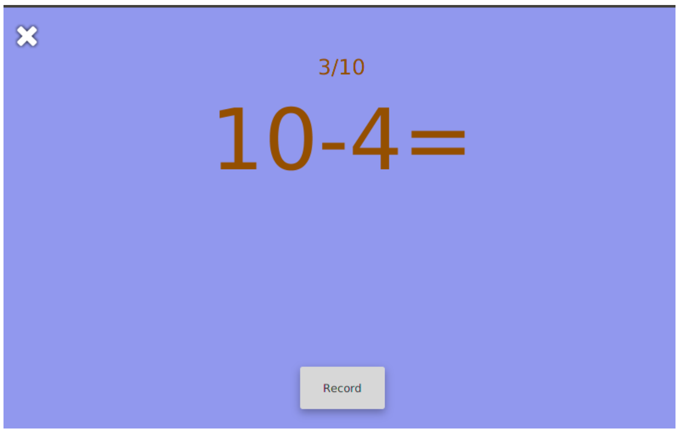


*Figure 1: Māori learning based Maths Aid Tātai – User Interface*

The application would also offer a range of additional features including the ability to input custom question lists, a tutorial and practice game mode, different levels of difficulty for questions and tracked statistics for individual users of the application.

The application was to be developed utilizing the Bash environment which is a UNIX based shell. As a result, the application is designed to work on a UNIX machine, and is optimized for Ubuntu, Linux. HTK software is also needed for speech recognition processes, and the libraries of JavaFX in conjunction with Scenebuilder, JFoenix, FontAwesome and Jackson JSON packages have also been used in the development of the application.

The way the project was laid out required several iterations to be released periodically while receiving feedback from peers and clients. Details and specifications were clarified as production went on, so short, efficient design iterations were required.

The project was to be completed in pairs. My partner was fellow software engineering student, Nathan Cairns, a partnership which worked well because of our similar ability, vision and work ethic. A significant focus of the project was on gaining experience in working in pairs and building good working relationships with our peers. Our working methodology was based off the Extreme Programming (XP) philosophy.

## II. Software Development Methods

### A. *Extreme Programming*

The extreme programming adaptive approach to code was deemed a suitable software development method for obtaining the high-quality software product that was required.

Due to the nature of XP it was essential that the project be centered around code design and client specifications. The user story in question had several fundamental operational demands, many of which needed to be expanded on before a clear picture of what the client wanted was obtained. This required asking the right questions to the client to develop a mutual understanding of the product, and get the client thinking about hard decisions needing to me made. Part of this process was also developing the clients vision for the end product, it is very rare a client completely knows what they want creatively, and it is part of a software engineers job to extract and clarify concrete information from abstract ideas and unclear intention.

Part of this process meant releasing several spikes weeks in advance, which would test the concepts of teaching based user interface, and voice recognition software. After the proof of concept models a first iteration was released mid-way through development marking the applications first iteration, which received feedback from peers in the class and allowed revision and reflection on the project direction, morals XP holds in high regard.

Extreme programming (or XP) is also widely known for its focus on team based culture. There are multiple XP values which our partnership implemented in building a strong, cohesive and productive working relationship.

#### 1) *Communication*

From an early stage it was emphasized that communication within the pair was to play a crucial role in the project. Much of the codebase was written in the same room as one another, which allowed concept, direction and issues to be discussed face to face. When not working together instant messaging platforms would be used to notify changes and additions to the code, as well as arrange meetings to discuss these issues further in person. In hindsight this likely prevented many communication errors and design misunderstandings, as decisions could be comprehensively discussed and developed on a personal level.

#### 2) *Feedback*

Reviewing one another's code was considered essential in the development process. It was common for programming methodology and code quality to be discussed in meetings relating to newly added code. Design and implementation decisions were always under scrutiny and carefully reflected upon. Design sketches and functionality notes were also heavily discussed before implementation, and the design of the user interface was constantly developed. Peer reviews further allowed us to analyze the quality, intuition and usability of the application.

#### 3) *Simplicity*

The code written for the project follows several clear and straight forward design patterns, including the singleton, model-view-controller, strategy, builder and adapter patterns. The patterns implemented are consistent across multiple class relationships in the codebase. Through planning and architectural analysis, the complexity of the program was able to be reduced before significant code writing began, with clear functionality of modules and relationships defined before implementation.

#### 4) *Courage*

I found I was encouraged to make changes boldly and promptly. Any changes, revisions or added features were tackled head on without shortcuts, and it was made clear to not be afraid to put in significant time and energy to the project.

#### 5) *Respect*

Respect for each other in the working relationship was an obvious merit both of us had to bring to the table. Any disagreements were carefully expanded upon and discussed, and solution was found through compromise and faith in each-others ability and vision.

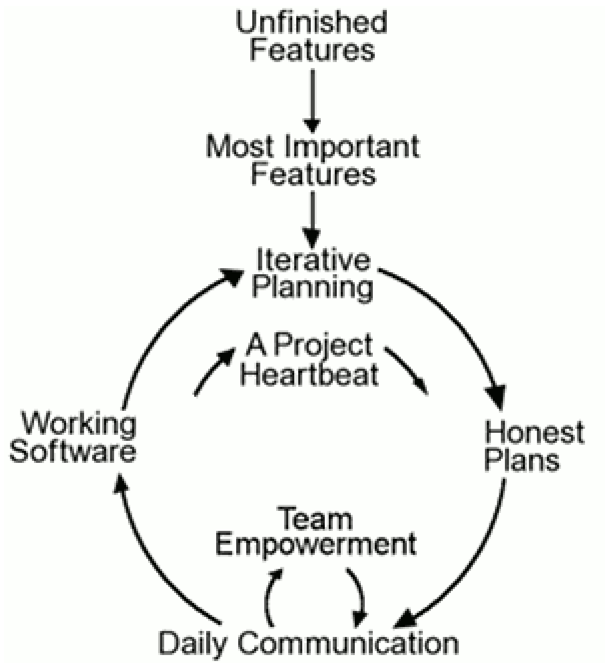


*Figure 2: Agile programming process flowchart*

A large part of XP also focuses on the relationship between team members. In my opinion Nathan and I worked excellently as a team because of our similar experience and skill level. The delegation of tasks was even throughout development. Initially, Nathan tackled the implementation of statistics data storage, as well as refactoring, adding CSS and the ability to resize the user interface. It was my job to focus on constructing game logic as well as much of the front end of the user interface. As the project progressed Nathan began focusing on implementing users and achievements, and I focused on adding the teachers custom list editing suite, as well as extra game modes. The communication we had in the pair reflects the values XP holds highly, as we were constantly discussing issues, direction and how to integrate changes. Working with Nathan again next time I feel it would be beneficial to do more pre-planning of the system on top of what we had already done. Although our high-level abstraction of the system was very good, more of it could have been done at an earlier stage, which would have increased the ease of programming later on. For example, adding multiple game modes with our current implementation served as a challenge due to some of our shortsightedness in implementing the first game type. Problems like these arose due to the iterative nature of the process, and requirements having a staggered release.

## B. Modelling

The importance of data modeling in the early stages of the project was made clear in our pair. UML diagrams in particular, were used to analyze the clarity and overall design quality of the program back end. This helped ensure the code architecture was robust, extensible and simple, which would allow further functionality to be added as an iterative process as specifications were clarified and feedback was received from the client.

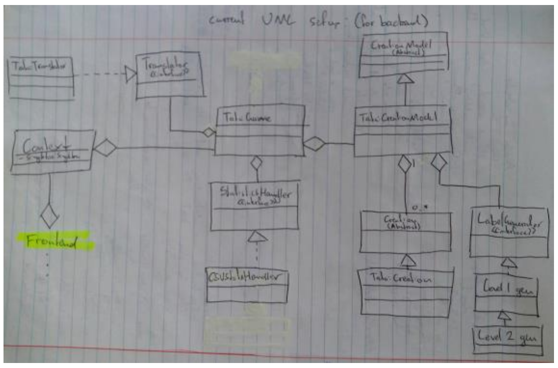


*Figure 3: Early UML diagram of Tātai backend*

User interface modelling was a further modelling tool used in preparation for coding the front end of the application. Screen diagrams were created for modelling menu navigation, ensuring logical control flow and an aesthetic look-and-feel would suit the target audience. As the application evolved and menu structure became more complex it was essential to plan transitions and states well before the design was implemented.

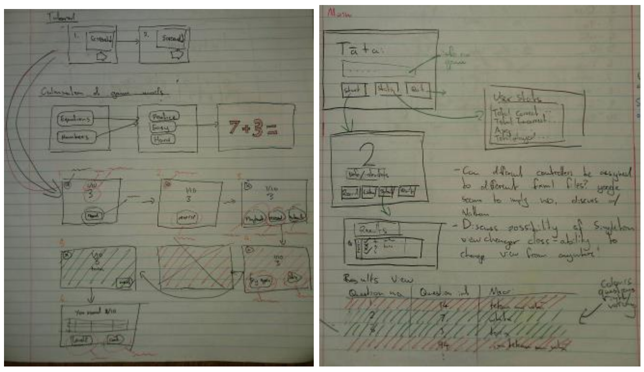


*Figure 4: Early screen diagrams showing intended tutorial window layout (left) and menu control flow (right)*

## C. Version Control

The form of version control used on the project was Git used with online repository server GitHub. GitHub is a distributed subversion control system used to track changes to files across versions. In the project it was utilized by committing changes to our local repository frequently, and often pushing the changes to our active working branch to avoid frequent merge conflicts. Merge conflicts were often avoided by effective communication and delegating tasks effectively, and those that occurred were often trivial. The few significant conflicts that arose were to do with merging changes in markup XML documents, changes that were often difficult to understand by looking purely at code. We worked around this issue by discussing what elements of the front-end interface we would work on and tried to ensure we would not work on the same XML file before pulling each other's changes.

The use of GitHub allowed separate modules to be developed independently, as well as integrated easily and consistently. This supported the XP manner in which the application was developed, as it became possible to integrate functionality as an iterative process. New features would only require a new branch to be created, features implemented then branches merged. This allowed my partner to work on independent features in the same or separate branches of a repository.

## D. Unit Testing

To ensure the quality of the software being produced it was made sure testing of complex classes was executed. The testing framework used in the project was JUnit, a testing framework focused on ease and repeatability of unit and module testing.

Some of the code in the project was written through test-driven development, where code would be written to pass the tests written by the software engineer. This ensured that the artifact being tested was indeed functionally correct before being integrated into the system, and supported the goal of producing a high-quality sofware system.

## III. PRODUCT DESIGN DECISIONS

The specification Tātai was based upon had several open-ended questions and requirements which needed further clarification and interpretation in our working pair. Requirements including the target age bracket, learning process and feedback systems, as well as ease of use were all decisions necessary for the software to be of worth. Some key decisions are discussed below.

### A. *Target Age Bracket*

The age bracket the Tātai prototype is designed for is a child user, aged 7-10. In this age bracket the functionality of the application was described as 'an aid that a child user, who is a first language speaker of Māori, will be using to improve their math's.' The functionality of the application was therefore focused on testing basic arithmetic skills in a Māori language integrated context. The program would have to be easy to use, and look engaging enough to keep a child's attention.

The look and feel of the menu was carefully chosen to reflect a clean and simple user interface design. The Simplicity and straight-forwardness of the layout and material design is intended to focus the user on the navigation of menus, and aims to ease the process of accessing program features. The color palette of the menu, it was decided, would be based off the colors found in the Māori flag among other common Māori designs. The combination of red, white and grey reflects the cultural relevance of the application while playing a role in keeping the complexity and intensity of the menus down. Refer to the figure below.

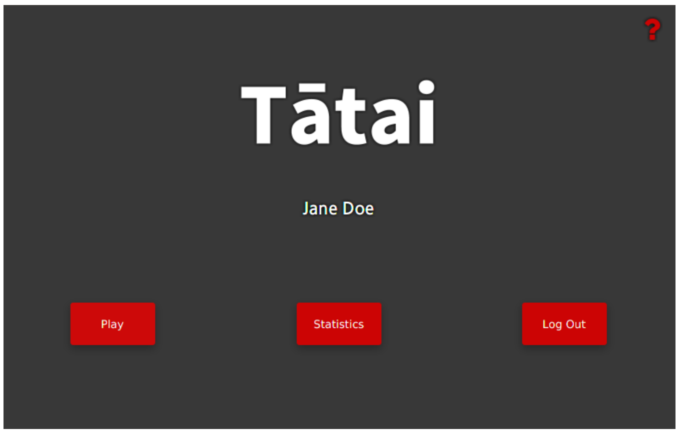


*Figure 5: Simple, clean look-and-feel of Tātai menu*

Further details were implemented in the look-and-feel to cater to a child user. Features like cursor-hand-hover (refer to Figure 6) over and dynamically changing buttons were considered an optimal way of communicating what is clickable on the screen. Importance was taken away from the use of the keyboard, and shifted to use of the mouse and on-screen buttons (refer to Figure 7). This is because children in this age bracket are typically mouse skilled in using a computer mouse rather than typing in a keyboard. It should be noted, however, that keyboard shortcuts do exist in the application and reasoning for this will be discussed shortly.

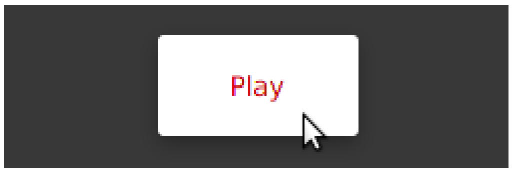


*Figure 6: Button color changing dynamically on hover*

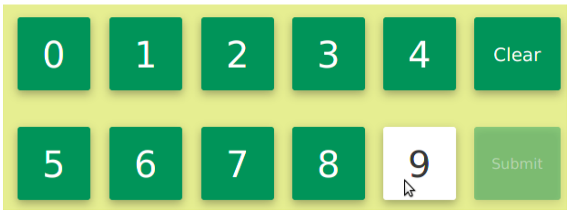
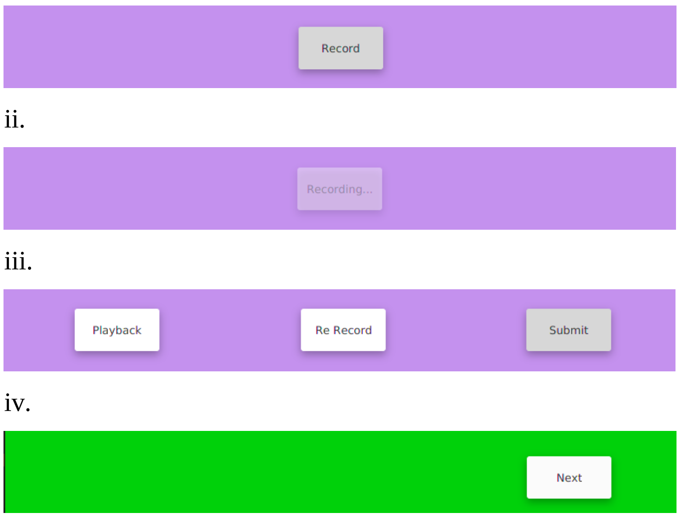


*Figure 7: Example of on-screen keyboard*

The design of the game mode itself contrasts that of the menu layout in terms of color and button changeability. Background and text colors are randomly generated on creation of a new game, and buttons appear and disappear about the screen depending on the state of the game. For example, when entering a question, only the record button is visible, with all others (re-listen re-record etc.), hidden from sight. This was a conscious design decision for the target audience, as for the age bracket of younger children it is desirable to make the actions required as obvious as possible, and remove the existence of actions which are not possible. This means that at any point in the game the user is faced with buttons pertaining only to what they are allowed to do at that time (refer to the figure below).

i.



ii.

iii.

iv.

*Figure 8: Dynamically changing buttons along bottom of game screen*

They greying out of buttons in many menu options where that action is not allowed follows a similar philosophy. For example, if level 2 has not yet been unlocked by the user the button for it is greyed out in the level select menu. Information can be obtained by hovering over it with the mouse. Decisions like these were aimed at making the user interface as logical and intuitive as possible, and remove most possibilities for user error.

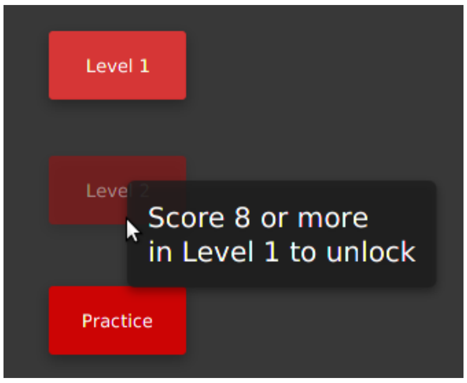


*Figure 9: Level 2 button disabled when not unlocked, mouse hover reveals further information on unlocking*

Window resizing was a further point considered to be of importance in the development of the application. For the target audience it was decided the ability to enter the application into full screen mode would optimize engagement. In context, the application would have its intended use in a school or home classroom, so in effect 'blocking off' the rest of the monitor screen and only have the application taking up the full-size of the screen ensures attention is paid to the learning aid program, and it is made more difficult for a child user to access other areas of the computer. Tātai can enter full screen mode on all views.

The statistics mechanics and view were designed with the function of showing user progress and development in their learning. Large, bold and colored numbers were chosen to support the colorful and engaging nature of gameplay, and the statistics menu was designed with interactivity as a core focus. The intention was to intrigue a user of the target audience by making many features of the menu cursor-hand-hover clickable, and information to 'pop out' when hovering over many of the icons.

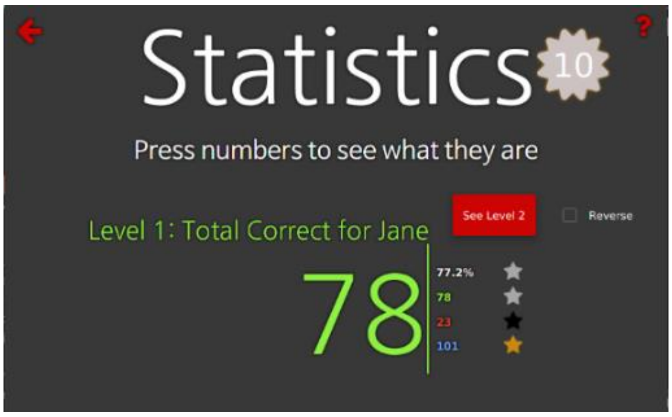

Statistics
10
Press numbers to see what they are
See Level 2
Reverse
Level 1: Total Correct for Jane
78
77.2%
78
23
101

*Figure 10: User statistics view*

Finally, the inclusion of keyboard shortcuts along menus and gameplay was a decision made with older users (teachers) in mind. The decision to have teachers as a potential end user introduced the necessity to have features increasing the ease of navigation for those with greater keyboard skill. On-screen keyboards, gameplay and menus all include keyboard shortcuts increasing the ease and speed of navigation/playthrough, and as we felt there was no need to limit the ability of any user, these key bindings are available to the student user also.

## *B. Software Packages and Frameworks*

Several code resources were utilized in the building of Tātai. The choice of whether to include a package was dependent on whether it would help ensure a high-quality piece of software was delivered to the client.

The majority of the program was written in the Java programming language, an excellent choice because of its object-oriented nature, making code written robust, maintainable and extendable. This fit well with the character of the project, with client requirements being released in stages, not all the information being known and an iterative approach to development. Java is also a highly portable language, often able to run on multiple machines with minor changes. It is also low level enough to justify use as a user interface language because of its resulting high computational speed.

The program uses an underlying statistical framework for its voice recognition functionality. Called the HTK speech recognition toolkit, it uses a Hidden Markov Modelling process to analyze speech patterns, and (for the project) came with pre-loaded statistical models for recognizing Māori numbers 1-99. The commands used to interact with the toolkit run in the Bash (Unix) shell. Running of the program therefore requires a Unix based machine. Development was done in the Ubuntu operating system for Linux.

The user interface was built using the JavaFX framework in conjunction with the JFoenix material design library. Many of the buttons and lists found on the application are from the JFoenix library, and the underlying structure of the front end is built off the JavaFX Model-View-Controller design pattern.

Further design resources include the Font Awesome font and icon toolkit. All icons (crosses, arrows, pencil icons, etc.) are from the extensive icon library the toolkit offers, and helped create an intuitive, logical and aesthetic design.

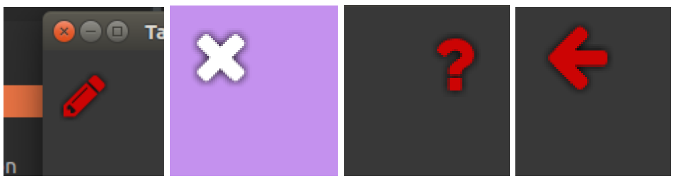


*Figure 11: Several FontAwesome icons integrated into Tātai*

Further resources needed for the project were Jackson JSON packages, which allowed the Java language to read and write to JSON file objects when working with user data. Packages used are the following: jackson-databind-2.1.4, jackson-core-2.1.4 and jackson-annotations-2.1.4.

## IV. TĀTAI – FINAL PRODUCT DESIGN AND FEATURE OVERVIEW

The final Tātai prototype meets all project specifications and as I am aware satisfies the client requirements. The Tātai application provides a game feature where users are presented with a mathematical expression, and the user speaks the answer as a Māori number between 1 and 99. Scores for a game of 10 questions are recorded for each individual user on each level, as well as their high score. There are two difficulties available to play in. The higher level (level 2) is locked for users until they prove their ability in level 1 by answering 8 or more questions correct in a single round. There is also a feature for teacher users able to write their own question lists for students to play. Other base features include; a well deigned graphical user interface, help on use of the application (tutorial), practice mode (refer to Figure 12), speech playback, a reward/achievement system and being able to run on a Linux-based operating system.

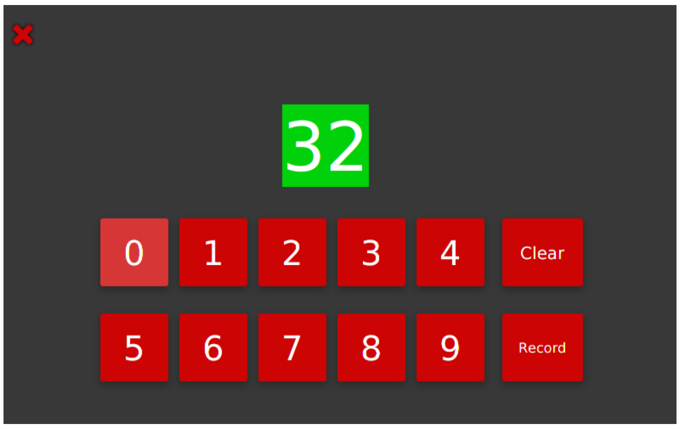


*Figure 12: Tātai practice mode giving feedback on correctly pronounced '32'*

On top of the base features included there is some added functionality in the application. Users have the ability to play in ‘reverse mode’, where a number written in Māori instead of a mathematical equation is shown on screen. The ability to sign in as individual users and have personal statistics tracked was a further feature added to the program. This includes having a special user known as a ‘teacher’, who has access to the custom list creation suite, and a standard ‘student’ user, who can collect and track personal achievements.

For a comprehensive discussion of features in Tātai consult the user manual.

## A. *Reverse Game Mode*

In the specification it was stated our target audience (7-10 years old) would be using Tātai to practice arithmetic rather than speaking Māori numbers correctly, as it was assumed the user would be a first speaker of the language. The reverse game mode for these users is rather trivial, but shows a step towards opening up the application towards a more diverse audience, including those who are learning the language for a first time. The reverse game mode offers the opportunity for users to learn numbers from scratch, they are shown a Māori number in word form, and have the option to listen to a pre-recorded voice pronouncing the number, (at least for numbers 1 through 9). There is an on-screen keyboard for users to type in what they think the number is as an integer, and further feedback is given in the same style as the traditional game modes (refer to Figure 13). This shows intent for expanding the application in future iterations to be a platform for children of diverse backgrounds to learn basic math’s skills in a Te Reo based environment.

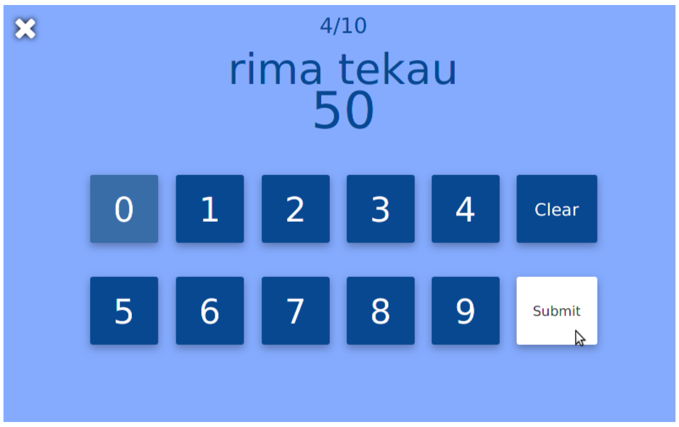


*Figure 13: Reverse game mode with answer '50' typed in from on-screen keyboard*

## B. *Teacher vs. Student Users*

The Tātai prototype we have introduced holds a feature of being able to distinguish between teacher and student users. Because the application is intended for use in the age bracket of 7 to 10 year old’s, it would make sense for the application to be used in the context of a home or public classroom. A teacher or parent (we will simply refer to as ‘teachers’) should therefore be able to sign in with admin-like privileges and be able to observe/interact with the student user base. Teachers can create, edit and test custom equation lists then submit them to a file, where all students then have access to playing them. Teachers also have the ability to play game modes, and achievements are not unnecessarily tracked on their user account, as it was assumed a teacher has no need for this.

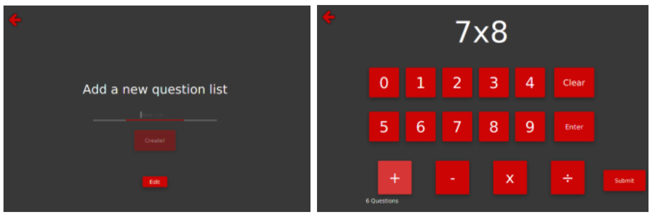
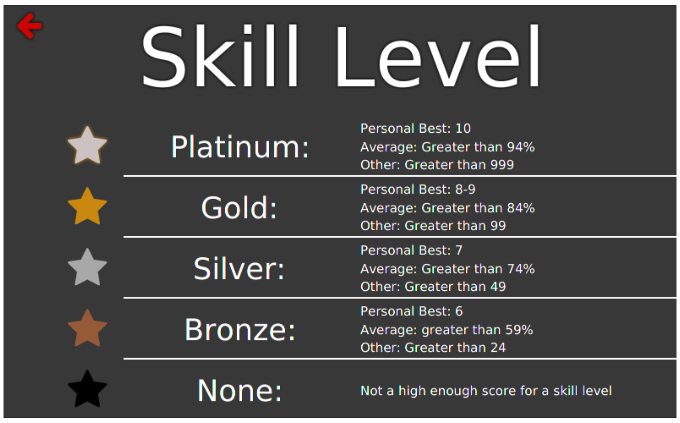


*Figure 14: Custom list editing suite accessed through teacher user portal*

## C. *Rewards*

One of the rewards systems in place in the application is the ability to unlock/gain achievements from the total amount of questions answered. There are levels of achievement associated with a certain volume of gameplay. Users can gain platinum/gold/silver/bronze achievements for their efforts in answering enough questions, as well as total amount answered correct or incorrect for each level. Status for answering a sheer number of questions, whether they be correct or incorrect is intended to engage the user in the application and entice further gameplay. The decision to reward a user for a high total incorrect score was made with the intention of removing the sense of penalty for answering a question wrong. The more the user challenges themselves, the more questions they will answer incorrectly, and it was not intended for user to feel discouraged from challenging themselves because of the possibility of increasing their total incorrect score. A high total incorrect score, if anything, shows engagement and progression in the users learning.



Skill Level

| | | |
|---|---|---|
| ☆ | Platinum: | Personal Best: 10<br>Average: Greater than 94%<br>Other: Greater than 999 |
| ★ | Gold: | Personal Best: 8-9<br>Average: Greater than 84%<br>Other: Greater than 99 |
| ★ | Silver: | Personal Best: 7<br>Average: Greater than 74%<br>Other: Greater than 49 |
| ★ | Bronze: | Personal Best: 6<br>Average: greater than 59%<br>Other: Greater than 24 |
| ★ | None: | Not a high enough score for a skill level |

*Figure 15: Tātai rewards table - describes required statistics to earn rewards*

## V. Usability and User Interface Design

The Tātai menu and game window design was intended to be as simple as possible. Initial screen diagrams laying out the design of the user interface reflect a similar clean, high-contrast look and feel to the eventual product.

### A. *In-Game*

#### 1) *Equation Generation*

Decisions on how in-game equations were generated were related to the target user of Tātai, who, at this stage would be using the app to practice arithmetic. Seeing as the intended user will be in primary school, it was decided to restrict the number of operators in an equation to 1, which is consistent with the New Zealand curriculum up to year 5, which focuses mainly on developing intuition with the four basic operations (The New Zealand Curriculum Online, 2009). Furthermore, where addition and subtraction questions are limited by evaluating to between 1 and 99, it was decided to limit division and multiplication to questions based off 12 times tables and below. This means no operand in a multiplication or division question will be greater than 12, which is once again is consistent with the New Zealand curriculum.

#### 2) *Slide Color*

The color palette chosen for the in-game user interface is based off a selection of ~30 pre-determined colors stored in the game code. The color selection is partitioned into two categories: lighter, pastel colors intended for background color generation, and darker, bolder colors intended for on-screen text and button color generation. Any combination of these two partitioned color sets are possibly generated. The choice of the color tones was made in an attempt to make the user interface high contrast and easy to read. The intended effect was to ensure no two colour combinations, no matter their similarity, would clash by becoming difficult to read over one another, ensuing accessibility for all types of users. Furthermore, it was decided the color pallette in the game mode would be an enabling factor in user engagement, concerning the younger age bracket the application is targeted towards, The ever-changing, unpredictable look of the question window was considered a captivating enough feature to help keep the target users attention for a whole game.

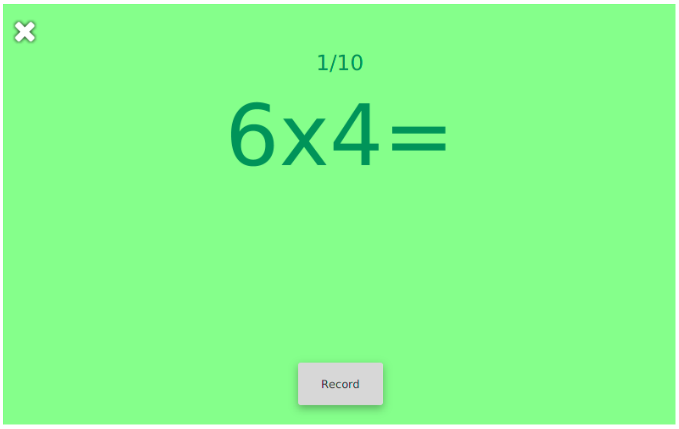


*Figure 16: Example of colors in game window clashing, still high contrast and readable*

#### 3) *Feedback*

Feedback received by a user for correct/incorrect answers displays different color schemes. When answering a question incorrect it was decided to have the background color turn a bold and commanding green, while all text turns white. The choice of green was to communicate a sense of achievement and encouragement to the user. Incorrect answers initially would cause a red shift in background color, however initial feedback through peer-reviews on this color decision was skeptical. It was later in production where alternatives were considered, and a dark-blue alternative was chosen. Reasoning behind the change was that red was considered too negative of a reaction by the application, and dark blue was still an alternative, way of giving more positive feedback.

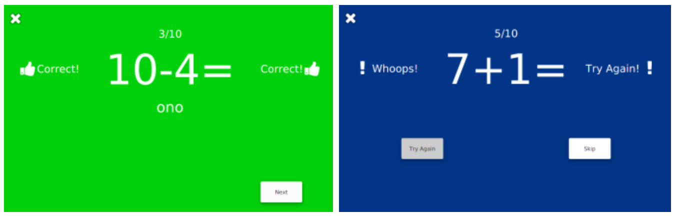


*Figure 17: Side-by-side comparison of user feedback: correct (left) and incorrect (right)*

Feedback labels in the game window appear in conjunction with the changing background color, as well as icons we hoped would further show proper question feedback. The thumbs up icon chosen for a correct answer was intended to display positivity and encouragement, whereas the exclamation mark chosen for incorrect answers was intended to indicate incorrect feedback while not inferring negativity or disapproval like an 'x' or thumbs down icon might have. The label phrases were chosen to give feedback in a spirited and constructive manner, and are designed to not detract from the users efforts. These additions not only complement the current user feedback mechanism but also improve the accessibility of the application for users with visual disabilities, such as color blindness, as they provide an alternative to receiving feedback other than color.

#### 4) *Live Statistics Tracking*

In the scenario where a user exits a game before completion, it was decided to track their statistics based off the questions they answered before ending. The reasoning behind this was to prevent users from being able to wipe a game worth of data from their overall statistics if they don't do well in a round. It would also ensure statistics have been tracked in the scenario where a game must be ended early due to shortness or time or other commitments.

### B. *Menu*

#### 1) *Keyboard Shortcuts*

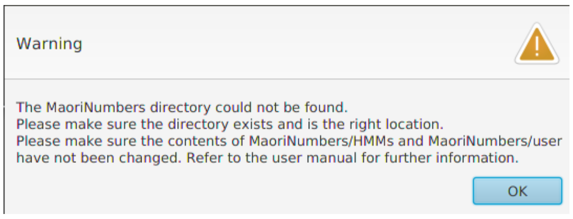
Across the application menus the user is able to use a selection of keyboard shortcuts to navigate and interact with the application. All registered keys are consistent with current key shortcuts found in most other established user interfaces, a decision made to make navigation as intuitive as possible. Refer to the user manual for a list of such shortcuts.

It is of note that keyboard shortcuts are heavily integrated into Tātai gameplay, and are designed to make the playing experience as straight forward as possible. The spacebar binds to the record, submit, try again and next buttons in, meaning it is the only button needing to be pressed to walk through a game.

#### 2) *Error Handling*

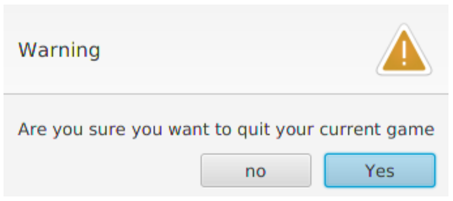
Careful consideration was put into preventing users from being able to enter the application into an error state. As mentioned, buttons are disabled/made invisible when not allowed to be pressed in both menus and game mode.

Other examples of error handling in the application include ensuring the proper HTK resources are located correctly. The error message for this window was carefully written to give the user a precise problem diagnosis, which otherwise may have been difficult to understand without some further technical knowledge (refer to Figure 18). Preventing invalid equations from being entered on the custom list equations creator and ensuring answers do not evaluate outside the range of 1-99 (refer to Figure 20) was another priority for user error handling, and again warning messages were shown to ensure the user was aware of their mistake. Furthermore, a maximum character length is in place for user and custom list names.


Warning

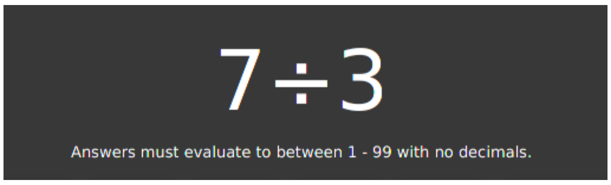
The MaoriNumbers directory could not be found.
Please make sure the directory exists and is the right location.
Please make sure the contents of MaoriNumbers/HMMs and MaoriNumbers/user have not been changed. Refer to the user manual for further information.

*Figure 18: Warning dialogue shown when attempting to start a g a game where proper hmm files aren't located correctly*


Warning

Are you sure you want to quit your current game

*Figure 19: Warning dialogue shown when attempting to exit a game half way through*


$7 \div 3$

Answers must evaluate to between 1 - 99 with no decimals.

*Figure 20: Warning label shown when invalid equation is input to custom equation suite creator, 'add' button also disables*

For iterations of the program beyond the prototype it would be beneficial to move away from a command line startup. Tātai can be started by double clicking the Tatai.jar icon if desired, on the condition that the TataiData directory has been moved to the users home directory. This is an indication of further improving ease of use for an average, non-technical user.

## VI. Design Constraints

In development of the program several limiting factors were encountered regarding the capabilities of the software. Compromises in design and features had to be made to have the final product meet client specifications.

The hardware that Tātai is designed for was a limiting factor in what could be added to the program. In accordance with the design specifications, the program is to be run on a provided Virtual Linux Machine, through virtual client such as Oracle VM Virtual Box. The buttons throughout the app were initially intended to resize by 120% horizontally and vertically when hovered over, which would further aid in communication to the user what they are able to be clicked. However due to the nature of the performance the virtual machine provided, buttons were observed to 'jump around' and disappear when hovered over, which was an unacceptable standard for the software to meet. The buttons were decided to instead change color when hovered over, which was less demanding on the machine running the program.

A further result of the restrictive hardware demands was the performance of recording the user. It was found during our phase of testing that most audio recorded during gameplay was distorted and choppy as a direct result of the virtual machines performance. In the code, we had implemented file checking processes during the recording that had to be restructured as a result, in an attempt to improve recording quality.

A further constraint was the limited range of numbers the voice recognition model was trained to recognize. Only the numbers 1 through 99 can be recognized, which limits the output of equations generated as well as the complexity of numbers able to be tested in the application. As well as this, there were some issues with the software being able to recognize words spoken in an environment with moderate background noise. Because this issue was out of our control we could only state in the user manual the conditions to which the application should be used – in a quiet location with recording equipment of at-least moderate quality.

## VII. Testing and Evaluation

### *A. Testing*

Some testing methods were implemented in the stages of developing the application. As mentioned earlier, the quality, correctness and robustness of our software was to be ensured by frequent testing of the source code itself. Several layers of testing were utilized in an attempt to work in accordance with industry standards.

1. Unit and functional testing played a significant part in ensuring class functionality was correct and could be relied upon. Backend classes often related to data manipulation and storage had unit tests written using the Java testing framework JUnit. Further discussion of this can be found in section II.
2. Integration testing was executed predominantly through manual testing of the application when new elements were added. It was ensured thorough user path coverage was used to test as many possible interactions between modules.
3. Exploratory testing was done predominantly by myself and my group partner, as well as fellow peers when it came time for peer evaluations. Results of this are discussed in the next part.

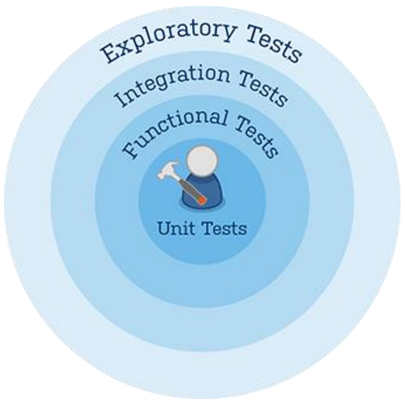


*Figure 21: Industry standard process of testing, provided by Atlassian*

### *B. Evaluation*

As the production of Tātai moved on it was essential to receive feedback not only from clients, but also from peers working on their own version of the project. A process was put in place where each working pair submitted their working beta version of Tātai to 3 random peers, and received feedback on the core and additional features. Specifically, evaluation forms prompted feedback for the following points detailed below:

1. Code quality and software architecture
2. Feedback for users of the application
3. Correctness of functionality and quality of requirements met
4. User Interface design

Feedback we received was generally positive. Features including the statistics view, main menu simplicity and good color contrast were received well. The backend code structure was also generally very well reviewed. Features that were given some critical feedback are as follows:

#### *1) Button Hover*

Issues with button hovering moving buttons around on virtual machine, discussed in section VI.

#### *2) In-Game Background Color Changing*

A general trend in the feedback of gameplay was that simply changing background colors to either green or (at the time) red was not sufficient as a feedback indicator. Further comments indicated that the background color changing without context was often confusing, and not accessible for people with vision imparities. It was this feedback that lead to the introduction of feedback labels and icons appearing when a question is answered. A final comment that was considered important was that the choice of red as an indicator of getting a question wrong. Many peers thought it was too negative, especially for kids. It was this comment that lead us to changing the background feedback here to a more neutral blue.

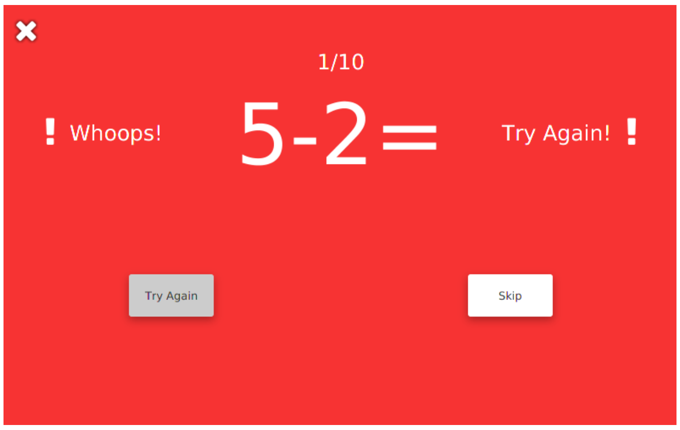


*Figure 22: Incorrect feedback in game window for previous Tātai iteration*

### *3) Menu Colors*

A further common comment received was that the menu color scheme was too dark and not child friendly. It was decided we could justify our use of a darker, slightly more mature theme by reasoning that most time spent in the application by students would be in gameplay, where the color scheme was sufficient to keep user engagement. Because the statistics window was also well received, which utilized a similar base color scheme, it was considered acceptable to leave the menu as it was. Furthermore, the teachers custom list suite was only intended to be used by teachers or a parent/guardian, so child-friendly accessibility was not warranted there. Finally, it was considered the importance of the color choices for the menu that were made at the beginning of development. These colors had always been intended to represent the Māori culture and bring some form of authenticity to the look-and-feel of the application and user experience.

## VIII. Conclusion

To conclude, I found the development process of Tātai to be incredibly humbling and rewarding. I am extremely pleased the finished product my partner and I have been able to deliver, and I have learnt much about working in with team members across the lifetime of this project.

I believe the most significant thing I have been able to take away from the project is the importance of communication above all, and the benefits of following a well-established software development practice. I found the experience of working with the XP process engaging and useful for the manner in which the project was structured, i.e. working in small groups with several small iterations over several weeks.

I am pleased with the functionality of the application as well as the additional features we as a team were able to bring to the clients. I believe our finished product stacks up well against our peers, and the specifications have been met to the best of our ability.

## Acknowledgment

I would like to acknowledge my partner Nathan Cairns for his consistent and quality work in bringing our vision to a reality. Also, to my peers of SOFTENG 206 I owe much thanks for their honest and open opinions, criticisms and help with aspects of the project. Finally, I would like to acknowledge the lecturers, tutors and course coordinators of SOFTENG 206 for preparing a high quality course and providing us with the opportunity to take on a sufficiently challenging, engaging and enjoyable project.

## References

[1] The New Zealand Curriculum Online. (2009). *By the End of Year 5*. Retrieved from http://nzcurriculum.tki.org.nz/National-Standards/Mathematics-standards/The-standards/End-of-year-5

[2] Extreme Programming. (2013). *Extreme Programming: A Gentle Introduction.* Retrieved from http://www.extremeprogramming.org/

[3] Atlassian. (2016). *Software Testing: A Culture of Quality.* Retrieved from https://www.atlassian.com/software-testing