Capstone Project Report

Deony Barrington
Department of Computer Science
BRRDEO001@myuct.ac.za

Matthew Scott
Department of Computer Science
SCTMAT003@uct.ac.za

Gyashka Manilall
Department of Computer Science
MNLGYA001@myuct.ac.za

Abstract

This report aims to provide a comprehensive representation of the WGame project. The project goal was to create a desktop application aimed at children between the ages of 9-14 that would serve as an effective medium to learn Afrikaans. This was chosen taking into consideration the team's familiarity with the language as well as it's prevalence in the South African education system. The team planned this project using Waterfall methodology and the approach decided upon was to gamify the learning aspect in three ways (three game modes): Matching Game, Guessing Game and Hangman. These games will be discussed in greater detail further on. In addition to these game modes, the application offers a Dictionary function that has no gamification and is there as a knowledge resource to users. Using Visual Studio (winforms, c#), WGame follows a multi-layered architecture and pulls data from an MS Access database.

1.1 Introduction

Now more than ever people turn to software applications to learn a new language. Our project aims to create an application to supplement the learning one receives at an educational institution and help ingrain and expand the user's vocabulary. To encourage interaction with the application and to enhance the desire to engage with the content, our application gamifies the experience to a certain degree by offering the content in the form of games like our hangman, guessing, and matching games.

Our program is geared towards usage rather than getting correct answers. There are no leader boards as we were not trying to engage with the user's competitive side, but instead focussed on repetition and vocabulary expansion to enhance learning. The desired outcome after a user has had continuous interactions with our games, is that the user will be able to communicate and understand basic words in Afrikaans, effectively and correctly. Our program focuses on delivering a proficient learning experience for Afrikaans; however, the UI is offered in multiple languages to braoden the target audience. The game currently offers the ability to switch between English, Xhosa and Afrikaans for the UI, however the system is designed as such that more could be introduced.

This project was completed using a Waterfall methodology. The planning for the project and analysis of the problem took place before any coding began. We made sure to keep up with the deliverables and the team met between them to make sure the project was on track. We constructed an evolutionary prototype and expanded it into what would eventually become our full game.

1.2 Requirements Captured

Use Case Narratives

Use Case: Select Language to Learn In
Actor: User Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).
Description: When the game launches for the first-time, User will be prompted with a "Welcome" screen, on the top right there exists a gears icon which when clicked, sends User to a new screen form to select a language they wish to set the UI in. Available language options will be displayed on screen in the form of a combobox, from which User will provide their selection via mouse-click. If the User selects an available option, the box closes with that language displayed in the preview portion of the combobox, they will then need to confirm the option via selecting the "Accept" button, if they don't wish to change the UI language they can select "Cancel". If User selects "Accept", the application's UI will be displayed in this language. If User selects "Cancel" then the screen will return to the previous "Welcome" screen. If there is no option appropriate for the User, the User can submit the language as a request where development can work on providing support for this language in the future, this is done via "Language Not On List? Click Here" button. This use case is related to "Home Screen", <<includes>> "Confirm Language Option" and <<extends>> "Make Language Submission".

Use Case: Select Word Category
Actor: User Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).
Description: On the Main Menu screen there will be a "Play" button that upon being selected, prompts the user with an exercise category to select. We offer 6 categories: Animals, Transport, Colours, Numbers, Common Objects and Food. Team decided on these categories being a good foundation for learning a new language and expanding vocabulary with high usage words. The categories will be displayed on screen as buttons, User will select one via mouse-click. Once an option is selected, GameChoice screen will be displayed. This use case is related to "Select Play", <<includes>> "Confirm Category Option" and <<extends>> "Choose Game".

Use Case: Get More Info

Actor: User

Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).

Description:

For certain screens in the app, there will be some form of player aid as the primary users for WGame are children, so they may be unfamiliar with the application or forget how it works - hence we tried to aid the players with useful info.

Users will be able to get more information via a "?" or "Help" button located at the top of the screen either as a right corner button or a menu strip item respectively. These can be accessed via mouse-click.

In "Welcome", the aid is given for first time users to know what the buttons do as well as provide a general overview of WGame, for example explaining the "GEARS ICON" or the Game Modes.

In text-input games, the hint is provided via a censored version of the correct answer being displayed.

This use case is related to "Start Game" and <<includes>> "Close Info".

Use Case: Choose game

Actor: User

Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).

Description:

On the GameChoice form, User will be prompted to choose a game between Matching, Guessing, Hangman and Dictionary. These exist as mouse-click buttons in the middle of the form.

Possible Workflows:

- User selects "Matching" - Picture-Word matching game screen is displayed.
- User selects "Guessing" - Picture-Input game screen is displayed.
- User selects "Hangman" - Hangman game screen is displayed.
- User selects "Dictionary" - A dictionary screen containing all words we offer in that category is displayed.

The User will be able to access certain menu items from each of these screens and be able to exit the game via an "Exit'' menu strip item - this takes User back to "Welcome".

This use case <<extends>> "Start game", "Select Progress", "Choose Another Game", "Exit Game", "Restart/Reset State"

Use Case: Play

Actor: User

Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).

Description:

On the "Welcome" there exists a mouse-clickable button, "Play".

Upon selection, User is taken to the "Categories" screen. This page exists as a simplified version of a Main Menu or Home page and is where the language for the UI can be changed. If User does not wish to "Play" they may exit the application via the "Exit" button on the top right.

This use case is related to "Exit Game" and "Choose Game".

Use Case: Reset/Restart Game

Actor: User

Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).

Description:

To enhance User customizability, there is a "Reset" option available in Main Menu as well as on menu strips. As this app relies on the User managing their own time efficiently to learn, we must account for a User not being able to use the app for months at a time and wishing to restart their learning path.

The "Reset" option resets the group of words the game is currently running and User will no longer be able to continue cycling through it, essentially when starting the game again, it will be like the first time. "Reset" or "Restart" also resets timers.

"Reset"/"Restart" can be selected via mouse-click.

This use case <<extends>> "Start Game"

Use Case: Start Game

Actor: User

Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).

Description:

Since every Game besides "Dictionary" is timed, the Team decided a "Start" button is necessary as Users clicking a game option and immediately a timer being counted down might be jarring. Selecting "Start" for a timer is also a norm that Users will be familiar with. Selection is done via mouse-click.

"Start" exists as a menu strip item on Matching, Guessing and Hangman. Once selected, the "game" begins.

This use case <<extends>> "Reset/Restart Game"

Use Case: Choose A Different Game
Actor: User Stakeholders for this use case are Client (Maria Keet) and the Tutor (Yash Ramsamy).
Description: Since WGame has a "Start" button, if a "Back" button was introduced in-game, it would require 2 "Back" clicks and may be confusing to younger Users - hence Team decided on a button that takes User straight back to "GameChoice" screen as it will likely be a very common use case. "Choose A Different Game" exists as a mouse-clickable item located in a left aligned menu strip. This use case is related to "Exit Game" and <<extends>> "Choose Game".

Functional Requirements

Interface

WGame is a gamified approach to learning and as a result there are many interface requirements that are functional requirements. It must be simple to use for children who may not be familiar with technology or English.

The interface is offered in multiple languages to remove the limitation of an English only perspective. The application also considers that we do not cater for every language and will allow the User to submit a language they would like for the UI.

No Main Menu or complicated Settings Menu, for children who are more prone to user errors, if they were to choose different languages, it may be difficult to return to the default English setting should there be a crowded Menu or Settings screen in a different language.

The interface has menu strips aligned left as well as appropriate buttons in industry standard positions.

Comic Sans font is chosen for its child friendliness, bright colours, larger buttons, and picture orientated games are used for this purpose as well.

Operational

WGame can teach simple Afrikaans vocabulary to its Users through three gamified approaches to learning. WGame offers a non-timed learning resource as well in the form of a Dictionary, so as to aid the User in their understanding. The team realised that the game modes do not allow for hints without removing the challenge/gamification aspect.

WGame must populate the word lists for the game modes by pulling data from a MS Access database. WGame does not have a convoluted workflow and screen progressions are as simple and efficient as possible so as to enhance the user experience for our target audience. Users are able to traverse the application seamlessly and not face any redundant inputs. The system is able to cycle through the database without repeating words in a single run of the game.

The application is free to use and will not house any adware. WGame does not require any internet connectivity to run. The final build of the project is an executable application needing under 100mb of storage space.

Gamification

The games have a timer that can be reset. The games provide feedback for inputs given by Users (ie, "Correct Answer") as well as an option to restart their current level (ie, group of words pulled from database). Users can access other games at the click of a button and not have to go back twice - as this is a

common use case. Games have a way to be closed without closing the application in case Users wish to select a different category. The games can handle some level of user error..

Non-Functional Requirements

For a gamified approach to learning, speed is an important non-functional requirement. Slow games or loading screens will discourage users from playing the games for long periods of time or prevent use at all. WGame makes use of a local data repository for the images to enhance performance and reduce data consumption.

To minimise storage needs, all images were compressed into 300x225 resolution and fit well on the forms. MS Access is also a more efficient form of storage compared to the current market alternatives. In terms of compatibility the executable should be able to run on all Windows operating systems

The application does not contain any security for User data because other than selected language for the UI we do not store any User data. This was chosen as the target market may not fully understand what their privacy laws entail or the sensitivity of private information. WGame is an application centred on learning so there is anonymity and no leader boards, User information is unnecessary.

A non-functional requirement Team thought was necessary was the ability to improve and enhance the application at a later stage. Currently the coding for UI Translations can take in any language if we have full translations in the database and change the UI, we can also add more and more words onto our categories and the games will work. The application's use of MS Access is an important requirement for future endeavours.

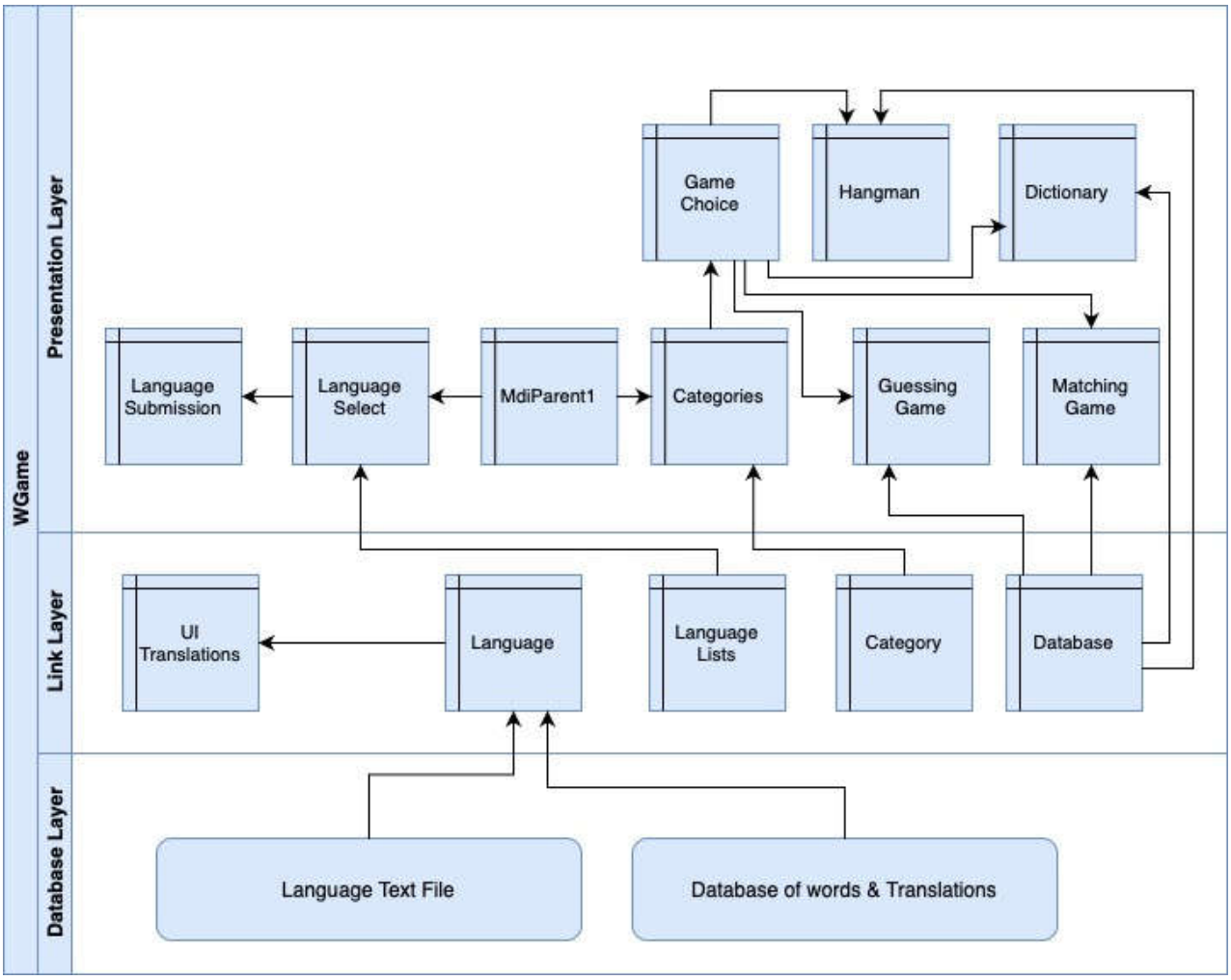
Usability

The application is aimed at a very young audience who may have varying levels of both technology and English proficiency, the UI and Game designs have been kept very simple. To aid usability there is help at what Team thinks may be confusing points of the application or where Users are most likely going to need additional information. On the Welcome Screen there is an info button providing detailed information of the buttons available to Users as well as an overview of the application offers. On the Categories pages the User can find more detailed information about the games via the Help button. To aid usability in non-native English speakers, the application offers the UI in multiple languages. It is often the case that to learn another language you need to know English first or have it be the base of your learning, WGame aims to remove this barrier and allow for language learning to be accessible by as many as possible. Team aims to provide this experience to an even larger market by improving the app to teach any language (to a rudimentary extent) in any language.

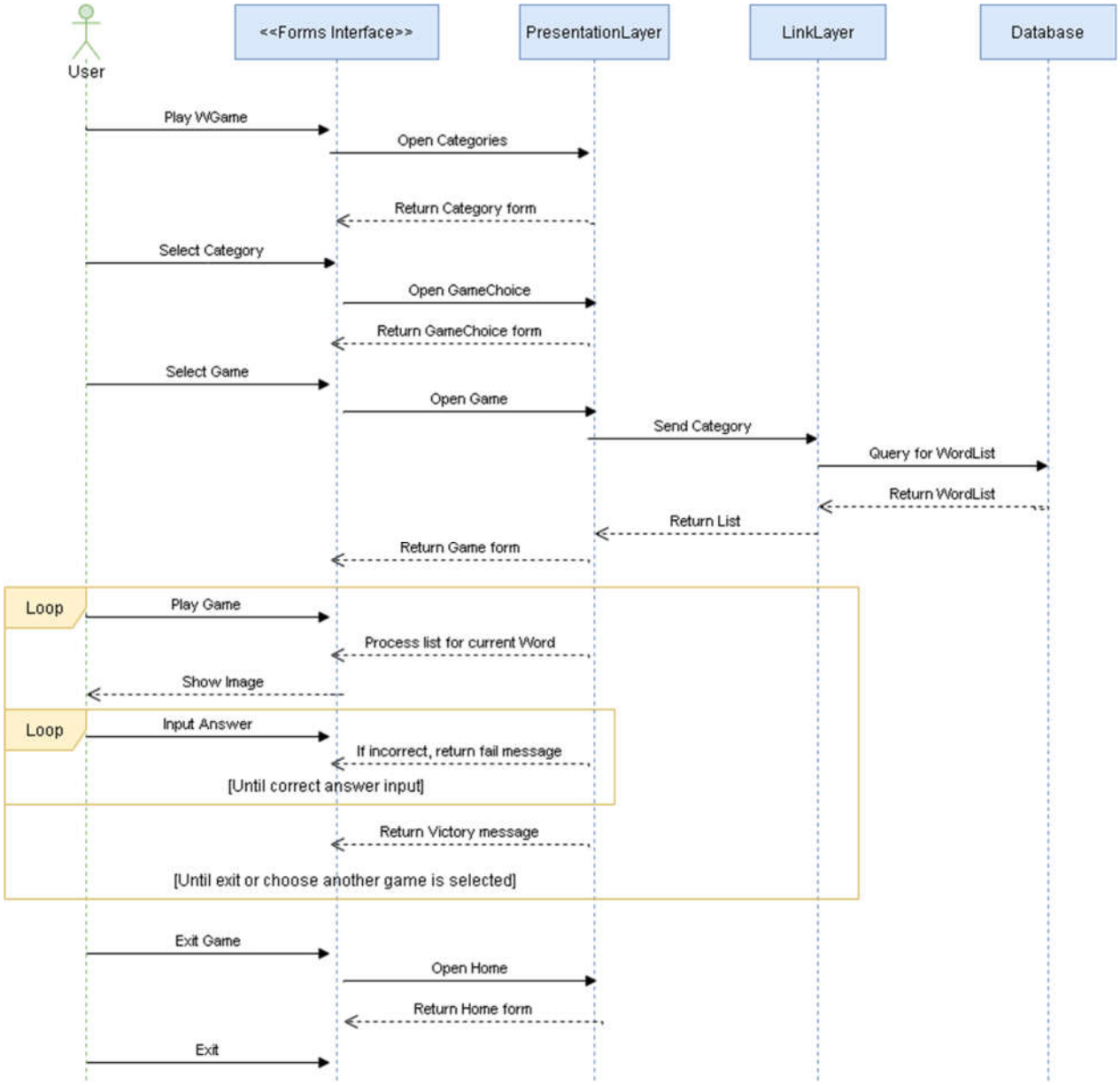
Discussion of Major Analysis Artifacts

The Artefacts most updated and discussed were the architectural diagram, activity diagram and use cases. These allow for a more comprehensive understanding of the system and its needs as well as gave Team, goals to work towards. Hints were one of the first use cases decided upon but ended up being removed to reduce program complexity at runtime and focus on getting the core functionality of WGame working at a desirable level. This use case was replaced by Aid Player instead in the form of additional info and a dictionary game mode, these are knowledge resources available to the User outside of the game. Changes like these were easier to implement and visualize due these artefacts. The sequence diagram is what helped visualize and optimise the workflow of our system.

Diagram of multi-layered architecture



Systems Sequence Diagram for Use Case: Play



1.3 Design Overview

The *WGame* project is an application aimed at expanding the Afrikaans vocabulary of the user. In terms of the design, the purpose of the program is to store data and present it to the user. The program should therefore extract the data and present it to the user in the most efficient manner. The *WGame* project followed a 3 tier-architecture layout, which includes a database layer, logical layer, and presentation layer in attempt to obtain the aimed for efficiency.

The database layer consisted of the physical database that stored all the data that *WGame* need to function. The database stores the words, the translations that the user will be learning, as well as information that aids in the functionality of the project. Information stored that aids in the functionality of the program includes the following:

- User interface (UI) component translations: *WGame* offers the user a range of options of languages in which to learn Afrikaans, e.g., they can select to learn Afrikaans.
- Levels: The program works based on levels. Words presented to the user are selected based on the level on which they are. The levels represent the number of words the user knows in various

categories and gives an indication of how well the user knows the words. The different levels have unique number of word allocations and times (in seconds) that the user should be able to identify/recognise a certain word. It further stores the number of points that should be allocated or deducted when I user gets a word correct/incorrect.

The logical layer contains all the classes that acts as a link between the database layer and the presentation layer. The logic layer is also a layered architecture. It consists of the following layers:

- Database logic layer
- Class entities
- Presentation Entity layer

Database Logic Layer

The database logic layer extracts the data into the List structures used throughout the rest of the program.

Class entities

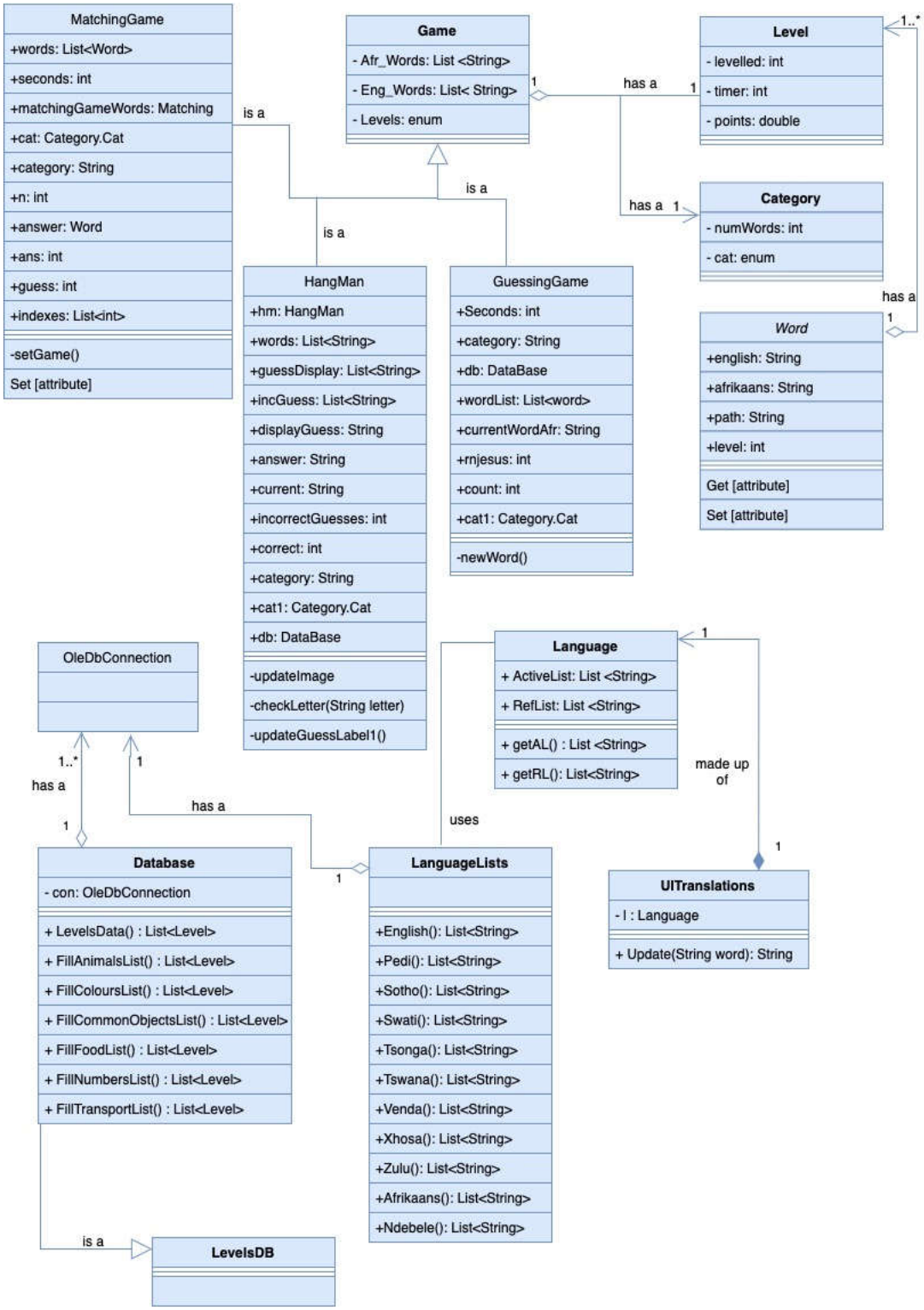
The class entities are classes that provide the skeleton to create objects from the database layers. These also include functions to manipulate the data according to the project's needs.

Presentation Entity Layers

The presentation entity layer consists of classes that represent the different games in the UI. These classes act as a link between the presentation classes and the database logic classes. It receives data from the presentation classes based on the user's input, for example which category the user selected, and information from the database logic classes, for example the level on which the user is for this category selected and combines this to provide a structure for the presentation classes of the games. It uses this information to display the correct words to the user with the correct timers, etc.

The presentation layer is responsible for receiving the user input and works with the presentation entity layer within the logic layer to determine the output (words) that should be presented to the user. It is also responsible for the actual output of the manipulated data because of the previously mentioned process.

Analysis Class Diagram



Algorithms & Data Organisation

The data is stored in an access database for ease of updating/extending and to keep the amount of data in working memory as low as possible. The data is read in its entirety from the file and then unnecessary elements are removed from the list during the execution of a game to maintain a list containing words of the correct level and category for the current user, due to this frequent insertion and deletion the List structure remained the optimal choice for WGame. When pulling in data for use in the games/ other aspects the database tables are read into lists of objects of the "Word" type - the List data structure was chosen as the best for this use case since all the parts of the program need to search and filter the data extensively which is the strongest part of the List structure. Another key strength of Lists that influenced their use was the variable capacity since each category has a fluctuating quantity of words. With the potential need to expand to have even more words and categories, a data structure which allows reuse of code without any changes needed to accommodate the differing resources is extremely beneficial.

A *linear search algorithm* was predominantly used throughout the project. Data was read from the database into various array lists according to a specified category, following the linear search algorithm. It was also used when the words needed to be extracted from within the games according to the user's level. Since the List data structure is used extensively, the inbuilt searching and access algorithms were primarily used for the program. The "MatchingGame" class does implement a simple divide and conquer algorithm for extracting the three indexes to be used in the game, a randomizer generates one index at a time and that index is added to a list if that index isn't already present. By finding each index individually and combining them later the algorithm builds in capacity for higher and lower capacity wordlists and gives the game greater variety.

1.4 Implementation

Data structures

Database: A database was used to permanently store all data that is presented to the user, as well as the information that aids in the functionality of the program, namely the levels table and the UI Translations table. This structure was chosen because permanent storage of data is necessary.

Lists: Upon reading from the database, the data was stored in lists. Lists are dynamic data structures which means they can grow and shrink however necessary. This enables the creators to freely add more information to the database. Lists were also used in the game entity classes. This has proven beneficial since the words presented to the user are based on the user's input (which category they had chosen) as well as the level on which the user is.

Text file: A text file was used to store the selected language. Text files are small and are versatile. Translating the UI involved a little more processing power as each text of each component need to be changed throughout the entire project. Loading all these translations into a list would therefore reduce efficiency. Reading straight from the text file was therefore the most efficient.

Variables: Variables were used throughout the program for data that needed to be manipulated or referenced

Images: Images are present throughout the project. These were stored in the project resource folder and only referenced via a path when the picture needed to be displayed. This increased efficiency, because the image would only be loaded into memory when it is called upon, as opposed to storing it in the UI components which would store the images in memory when the program is run and significantly increase the loading time of the project.

User Interface

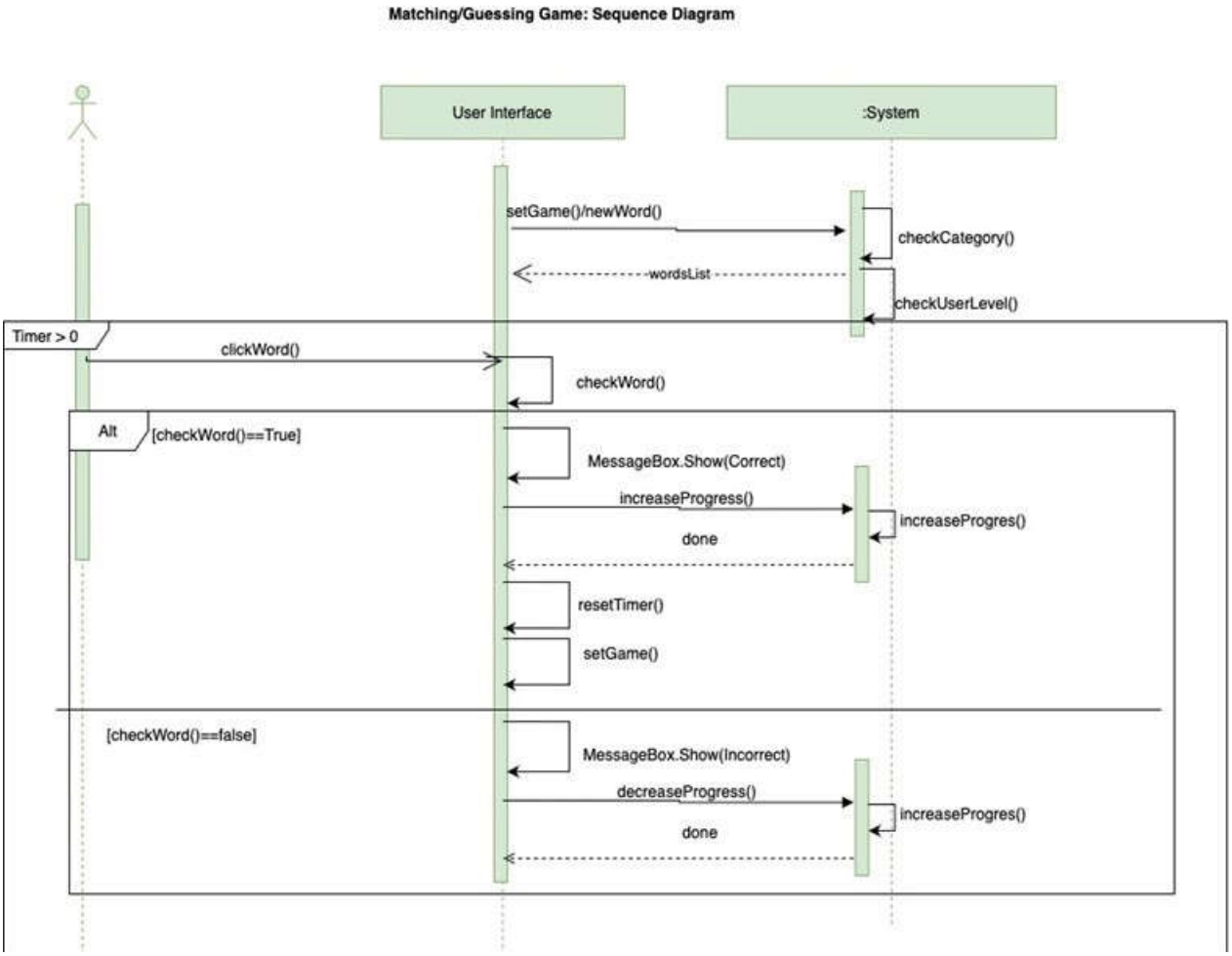
The user interface is a key feature in this project. The user provides input, the categories and game that they want to use, which allows the program to determine what needs to be presented (output) to the user. The user interface was implemented with visual studio (c#) which allowed us to organise components on various forms. The UI is the most important part of the user experience and we have chosen this method for creating the UI because it allowed us to design our user interface in the most timely and accurate manner. Resources (pictures) were extracted from the database and project folder that aided in the aesthetics of the user interface to make it more appealing and fun to use.

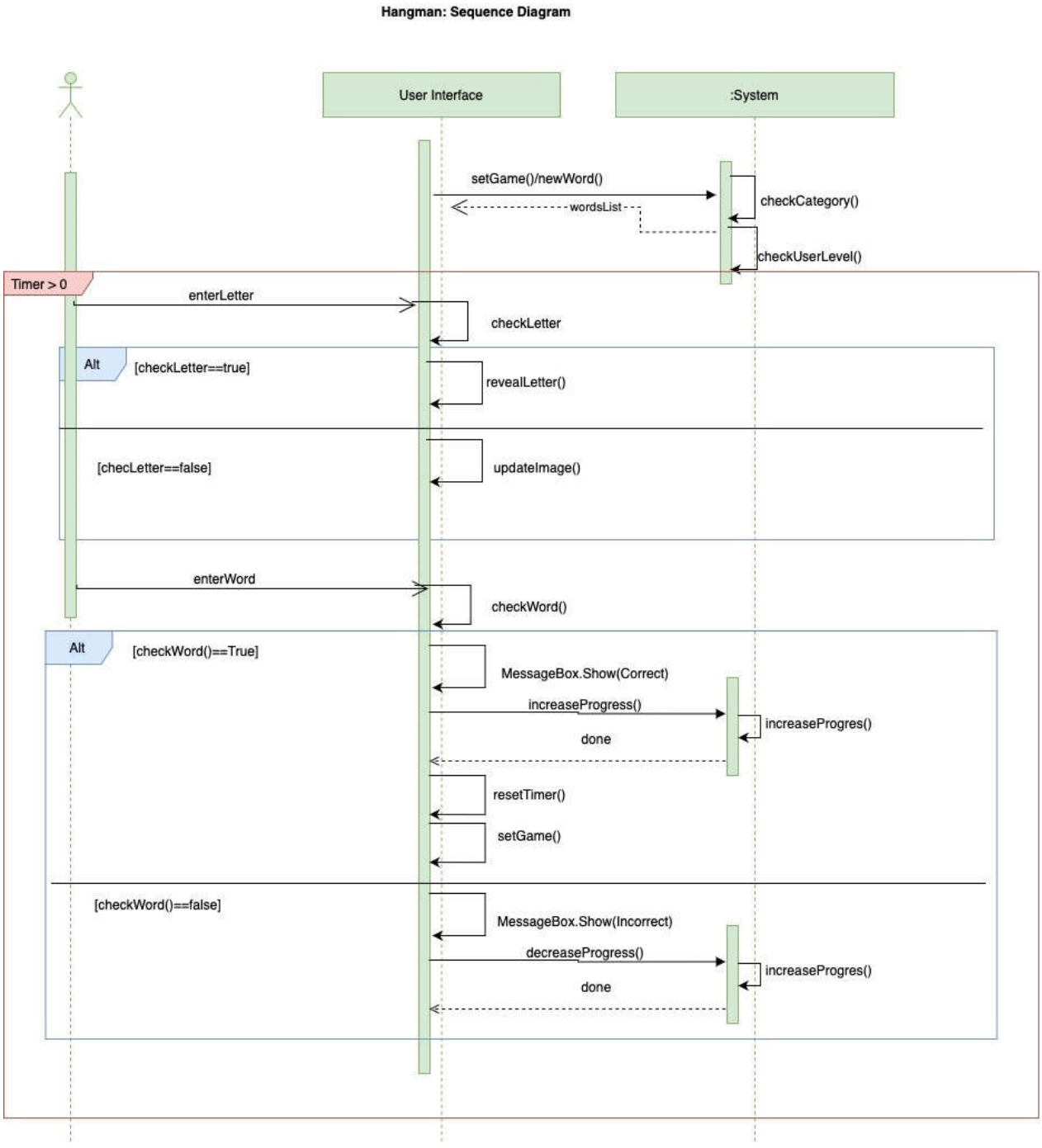
Class Functions

Important Classes and Important Functions

Class	Function description
Database	Reading from the database and storing it in array lists. Made access to the data from other classes in the program possible.
Game classes (Hangman, Matching, Guessing)	extractByLevel() methods were important across these classes. This ensures that the user is not presented with words that are more advanced that they can handle, which is determined by the level on which they are.
LanguageLists	This class had functions to extract each of the translations based on the user's input, to translate the components
UI Translations	The Update() function is responsible for changing the language of the user interface components across the project. This is essential so that the program's usability is not limited to English-speaking users.
Guessing Game	newWord() is responsible for selecting a word at random, given a specific category and user level. This method is what makes the guessing game work.
Matching Game	setGame() is responsible for selecting a word at random, given a specific category and user level, and displaying the corresponding picture. This method makes the game work.
Hangman	In the hangman class, there are 3 essential methods. The first is responsible for selecting a word at random, given a specific category and user level. The second is the updateImage() method. Whenever a user guesses incorrectly, the hangman has a new state (e.g. first just a head, then a head and body, then a head, body and leg, etc.). This method ensures that the image updates, showing the user how far they are from running out of guesses. The last important method is the checkLetter() method. This method checks if the letter that is input by the user is in the word, and if so, reveal the position of the letter, else add it to the list of incorrect guesses.

Sequence Diagrams for Games





Libraries

.NET Class Library was used for the following:

- Represent base data types and exceptions.
- Encapsulate data structures.
- Perform input and output used for reading from the text file
- Provide data access, rich client-side GUI, and server-controlled, client-side GUI.
- Read from database

1.5 Program Validation and Verification

WGame is implemented within the windows forms structure and runs on the .net framework. Due to the complexity of class interaction and the interfacing, automated testing was not feasible in the time constraints. Certain functions were tested outside of the project before being implemented within the classes to maintain a form of validation. With the amount of interrelated code and the database access, an alternative testing approach was created. Focusing on usability and core function over fully exhaustive testing, the program will offer a stable and enjoyable experience for most use cases. An additional reasoning for this approach comes from the game being relatively limited in how much an end user can do within it, the faults that would come from game logic errors were tested extensively before being implemented in the program code. Table 1 provides an overview of the testing plan.

Process	Technique
1. Class Testing: test methods and state behaviour of classes	Boundary Value Analysis, White-Box Tests, and Equivalence Class Partitioning
2. Integration Testing: test the interaction of sets of classes	State Transition and Ad-Hoc testing
3. Validation Testing: test whether customer requirements are satisfied	Use-case based black box and Acceptance tests
4. System Testing: test the behaviour of the system as part of a larger environment	Stress testing, and Ad-Hoc testing

Table 1: Testing plan defining techniques used.

Quality Management Plan

The quality management for WGame is tailored to each section of the game's components. The user interface is critical as the front facing part of the program and its ease of use/readability are key, to this end a child friendly, and large font was used throughout the program along with a calming background colour theme. A menu toolstrip is available on all forms past home and provides contextual navigation around the program.

The database forms the basis for all the learning opportunities throughout the program and contains the translation keys for the multilingual support of the program. Each category has a minimum of 40 entries for English words and their Afrikaans translations, the colours category is an exception with only 15 entries however, this category is not something that needs to be explored in detail. With such a sizeable dictionary to pull from, the game elements of the program can provide hours of education and entertainment for users of all ages.

Similarly, the "dictionary game" allows a detailed look at the information stored in the database, in an easy to read and clean layout (list of items in a specialised control, the image of the current item along with the English and Afrikaans words for that item).

Game functionality for Testing Purposes

Matching game pulls a list of words and file paths from the database, based on the category, and asks a user to select the correct word for a given image, correctly guessing the word ends the game at which point the user may select play game to continue playing, from a list of 3 randomly selected ones. Guessing game is a more difficult game requiring the user to type in the Afrikaans word for a given image within a set duration, the list of words is pulled from the database based on category and words are removed once correctly guessed, the game shows 5 images per round and these are removed once correctly guessed. Hangman functions like the classic game with users guessing either letters or words, each incorrect guess draws on another appendage to the "hanged man" until he his fully hung or the word is guessed, correct guesses have their location revealed in the word instead.

Validation

Data Set and reasoning	Test Cases		
	Normal Functioning	Extreme Boundary Cases	Invalid Data (program should not crash)
Select Index	Passed	N/A	N/A
Choose different game	Passed	N/A	N/A
Exit	Passed	N/A	N/A

Table 2: Test cases for Dictionary class

Data Set and reasoning	Test Cases		
	Normal Functioning	Extreme Boundary Cases	Invalid Data (program should not crash)
Start	Passed	N/A	N/A
Submit	Passed	Passed	Errors Handled
Restart	Passed	N/A	N/A
Choose a different game	Passed	N/A	N/A
Exit	Passed	N/A	N/A

Table 3: Test cases for Guessing Game class

Data Set and reasoning	Test Cases		
	Normal Functioning	Extreme Boundary Cases	Invalid Data (program should not crash)
Start	Passed	N/A	N/A
btnWord1	Passed	N/A	N/A
btnWord2	Passed	N/A	N/A
btnWord3	Passed	N/A	N/A
Reset	Passed	N/A	N/A
Choose a different game	Passed	N/A	N/A
Exit	Passed	N/A	N/A

Table 4: Test cases for Matching Game class

Data Set and reasoning	Test Cases		
	Normal Functioning	Extreme Boundary Cases	Invalid Data (program should not crash)
Play Game	Passed	N/A	N/A
Check	Passed	N/A	Handled, with pop-up prompt describing valid data.
Reset	Passed	N/A	N/A

Choose a different game	Passed	N/A	N/A
Exit	Passed	N/A	N/A

Table 5: Test cases for Hangman class

With the user input limited through buttons and text fields, there are very few points of failure within the running of the application, the stability of the program is mostly reliant on the core game logic and correct storage of the referenced image files. As a learning game for children, the program fits the needs of its target audience while being robust enough to deal with the possible invalid data when the need arises. Stringent testing of the game logic using boundary value, stress, and equivalence class partitioning ensures that the games will run correctly. The equivalence partitioning was a potent testing approach for WGame since the data entries used are mostly uniform but needed to be tested extensively for false reads from the database, having the working data split and tested for valid and invalid cases alike gives Team confidence that our logic and coding are functioning as intended. WGame is simple and functional with a fun twist to learning a new language.

Document management:

The program files were hosted on a Microsoft Teams server, and through the CS department GitLab server. The redundancy was necessary due to difficulties accessing the GitLab server throughout the development lifecycle. Teams offers a check out/check in system that ensured no work was overwritten however, this did force the team to work sequentially on the program rather than in parallel which did slow development. Project files were only uploaded when in a working state to allow for other members to test the new features and add new code to a functional build.

1.6 Conclusion

The WGame team did not meet full features first planned out but achieved core requirements, producing a game that will teach rudimentary Afrikaans vocabulary to Users. Utilising gamification and knowledge resources to keep Users engaged and entertained with the learning process. Unfortunately, due to the skill level and time constraints, Team was not able to implement Progress tracking in the final build despite it being a prominent part of UI; however there remains incentives in completing the games and enjoying the exciting images chosen for the games. Technical features meet the Client requirements with a moderate final build size under 80mb (portability) and complete offline usability (accessibility). Additional features include multi-lingual support as well as an interactive Dictionary that is free for the User to access within the program.

References

Ang, C. (2006) Developing Enjoyable Second Language Learning Software Tools: A Computer Game Paradigm. User-Centered Computer Aided Language Learning, 1-21

Fogel, K. (2005) Producing Open Source Software, O'Reilly Media, Inc., Sebastopol, California, United States

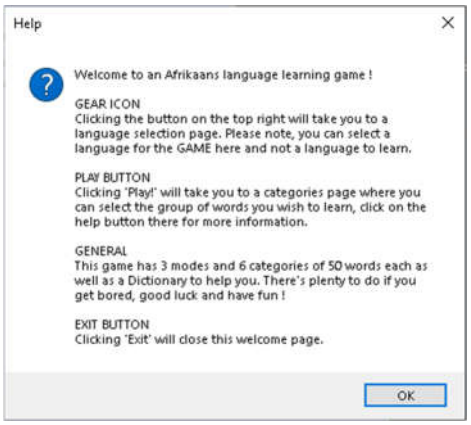
Sommerville, I. (2016) Chapter 6: Architectural Design, Software Engineering, 10th Edition, Pearson, Cape Town, South Africa

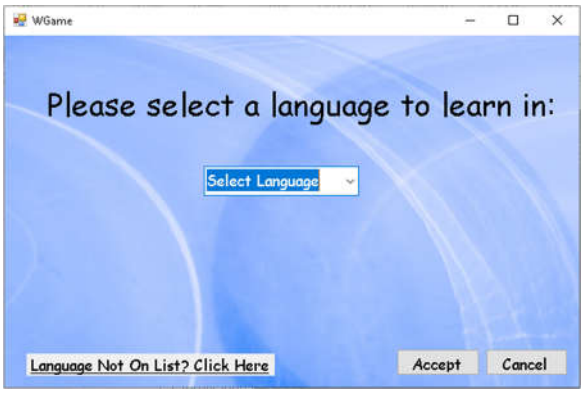
All used in the project were obtained legally from https://unsplash.com/

Appendix 1 - User Manual

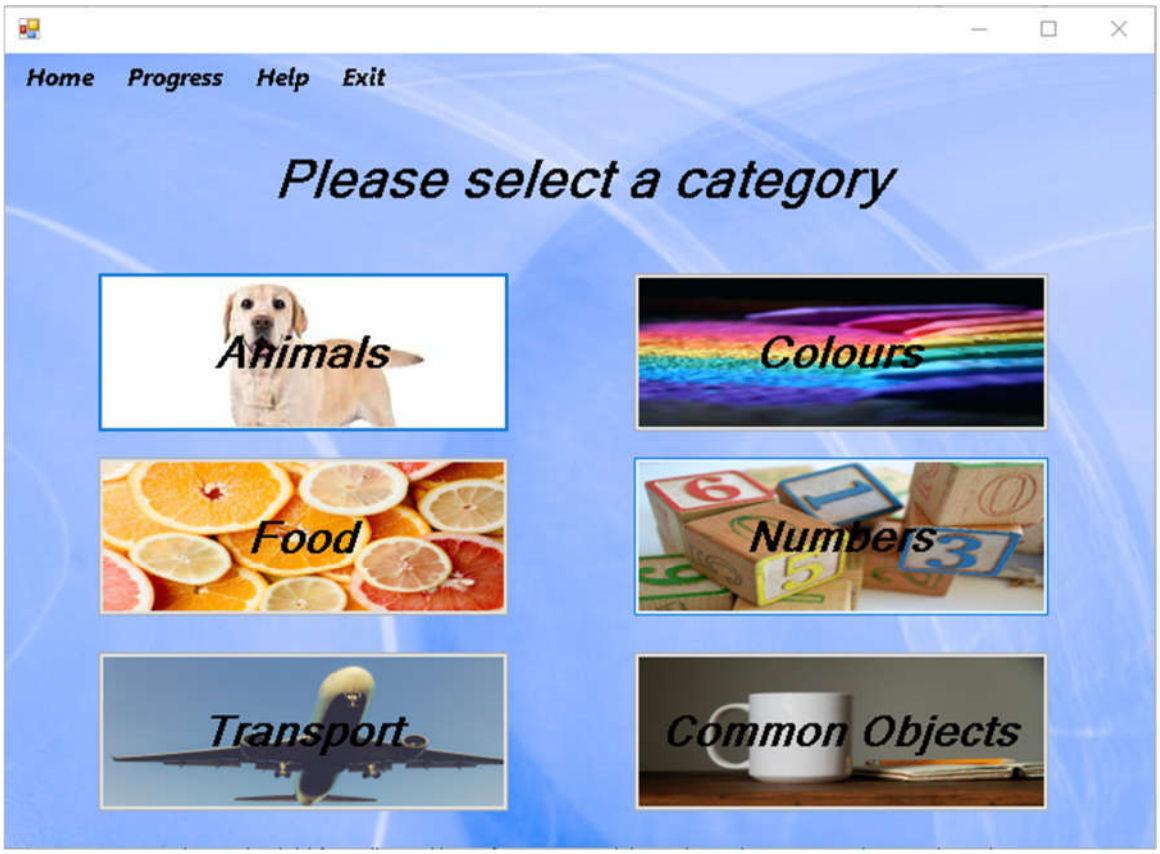
Start Page:

Once the executable (in the debug folder) is run, a window will pop up with the welcome page displayed. From here we can click the center 'Play!' button, which will take us to the categories window to begin the learning experience. The top 3 buttons from left to right are '? (Help)', '⚙ (Settings)' and 'Exit'. The Help button(provides detailed information of the buttons available to Users as well as an overview of the application offers, the Settings button takes us to the language options page (where we can change the language of the User Interface, as well as request new language additions), and the Exit button will close the program.





Category Select:

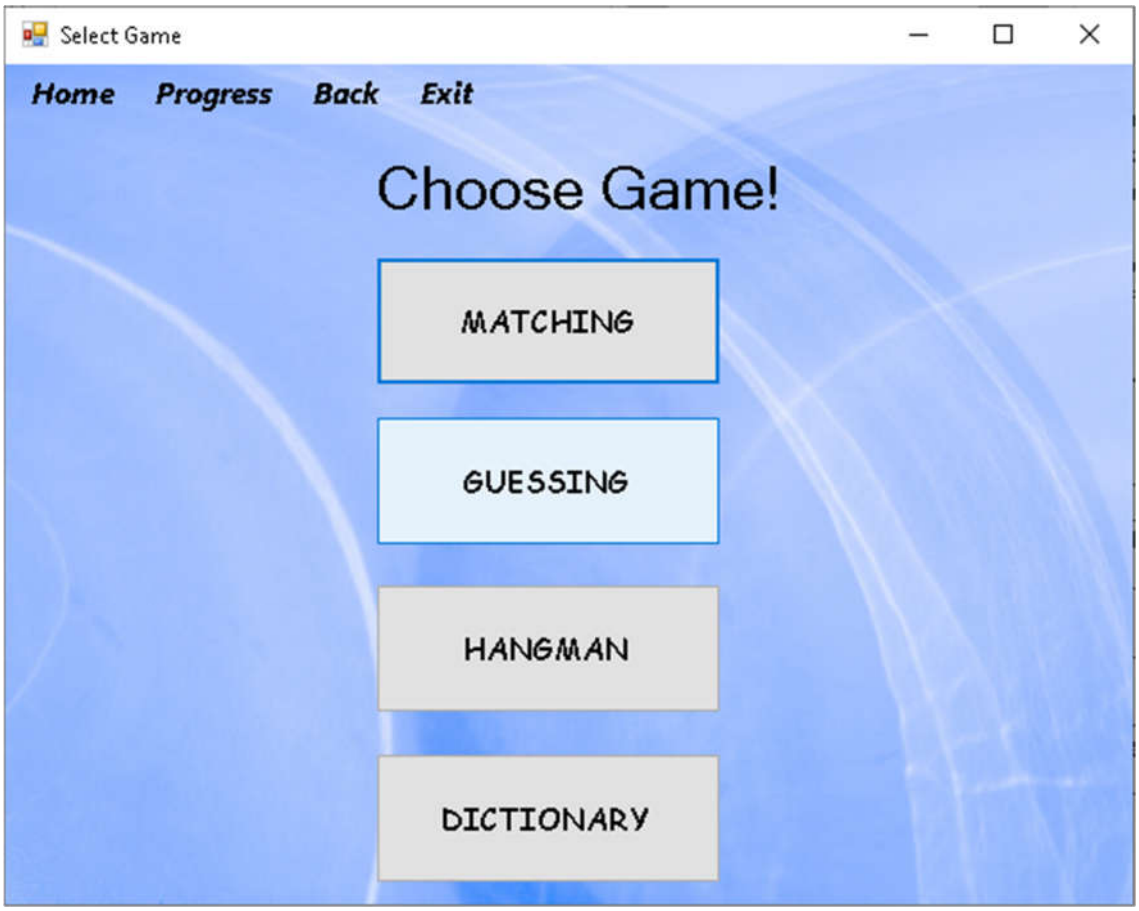


On this page you will be able to select from one of six categories. When you do you will be taken to the game selection page. Your selection on this page will determine what words are used in the games you will play. E.g. if you select Colours you will learn words like red (rooi), green (groen), etc.

The Menu Bar:

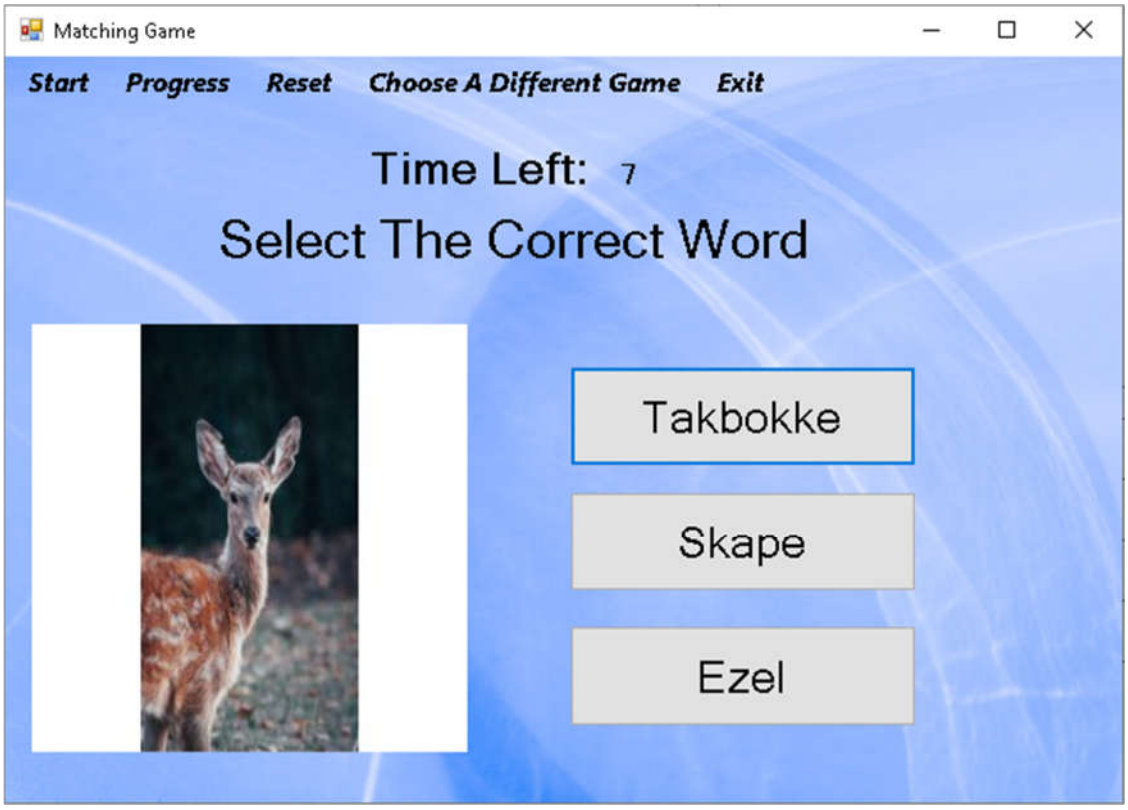
On the categories window and many other windows in the game you will come across a menu bar at the top of the window. This menu bar will help you navigate this program. Buttons you will find in this menu bar include Home (which will take you back to the welcome window, Help (which displays the same information as it did on the start page, Exit (which closes the game), Start/Play game which will start whatever game you are currently on, Restart/Reset (which allows the game to be restarted), and a Choose a Different Game button which navigates you back to the game selection window.

Game Select:



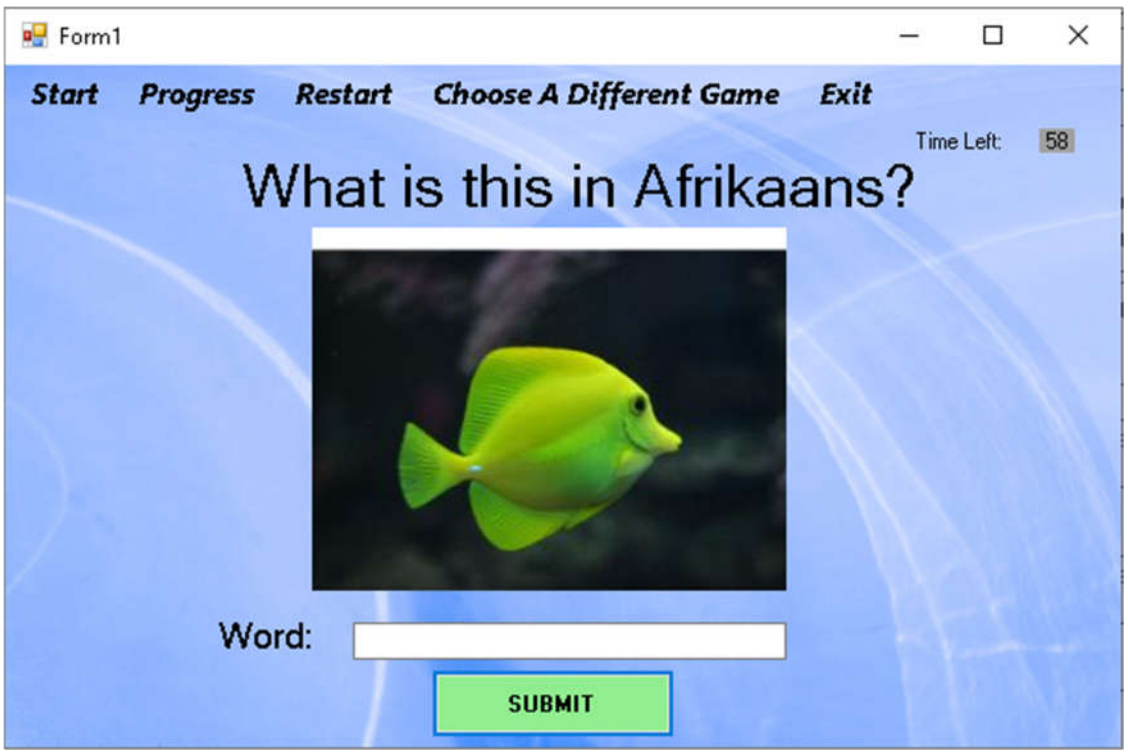
After selecting a category, you will need to select a game. There are three games to choose from. Matching, in which you must select the correct word of three available Afrikaans words that matches an image of the thing the word represents, within a time limit. Guessing, where you are shown a picture and must type the Afrikaans word of what is represented in the image. Hangman is traditional hangman, where you need to guess a word by suggesting letters within a certain number of guesses. There is also a fourth option: Dictionary. Dictionary is a non-gamified learning tool that consists of all the words that are part of that category. Using this function, you can look up words and their translations, so that you can practice the words that will appear in the games.

Matching Game:



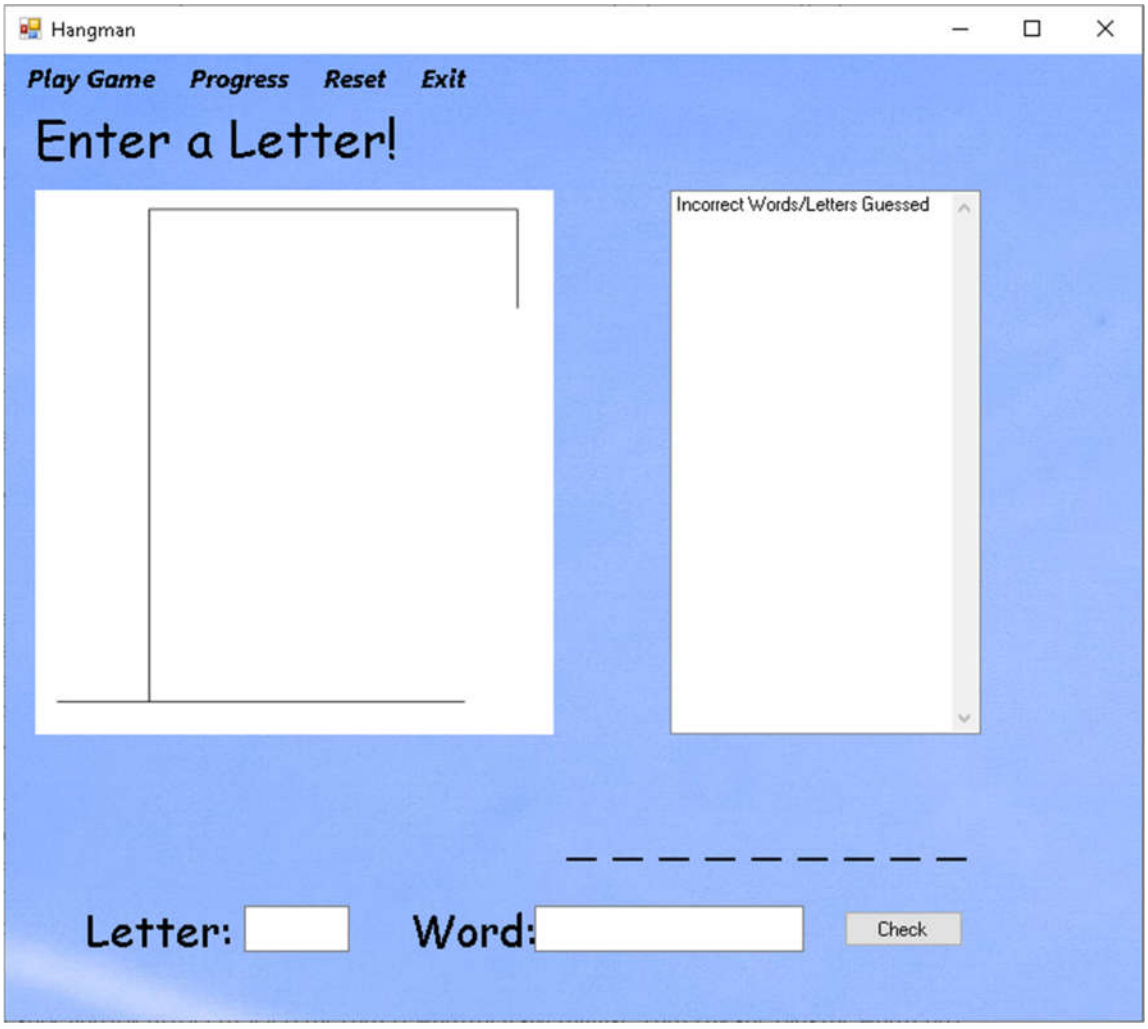
When the start button is clicked a timer will begin to count down. You will see an image appear to the left and three buttons with three separate words to the right. You will need to click the correct word that matches the image. When you select the word, the game will progress to the next image and next set of words until you are finished the game, or the timer runs out.

Guessing Game:



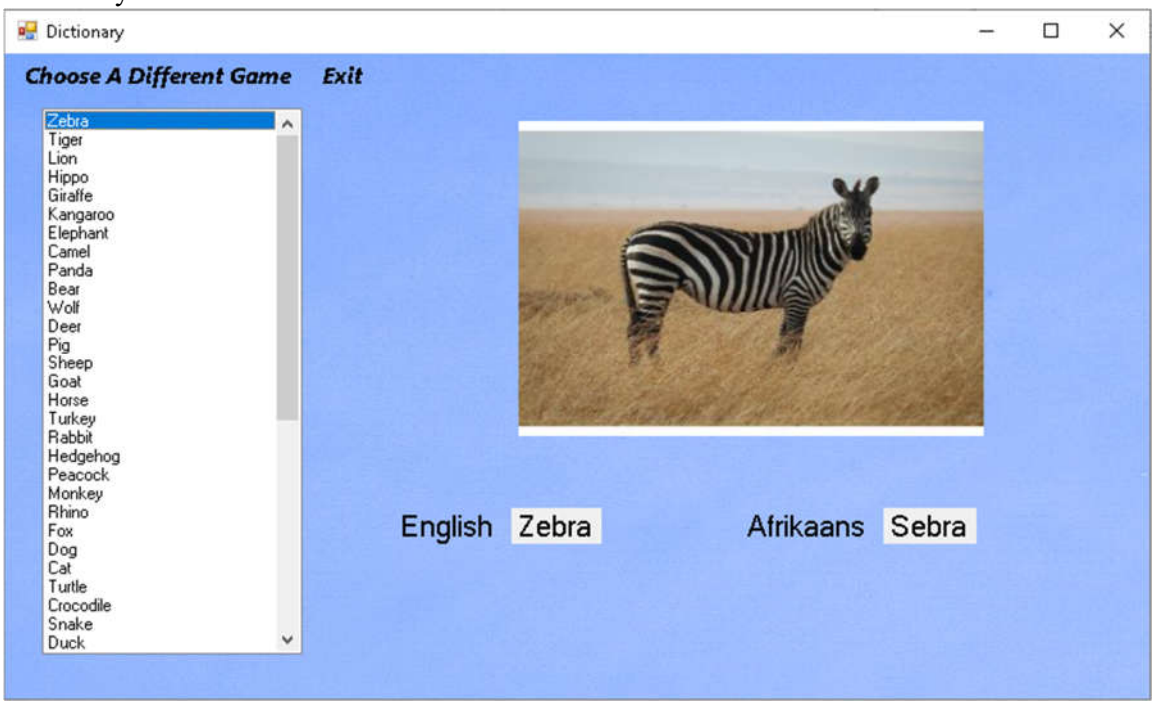
When the start button is clicked a timer will begin to count down. An image will appear in the middle of the screen. You will need to type the correct word to match up with the image. When you submit the word, the game will move on and load a new image. This will repeat until you finish the game or run out of time.

Hangman:



Once the start button is clicked you will be able to begin guessing what the word is. Type a letter into the letter field and it will replace one _ with the letter if the letter shows up in the word. If the letter is incorrect it will display in the incorrect words/letters list. You can also guess the full word, which if right wins the game and if wrong will display the word in the incorrect words/letters list. The game will end when either the full word is guessed, or you run out of time.

Dictionary:



In this window you will be able to select a word from the category selected previously. When a word is selected, the English word, its translation and its associated image will be displayed for you.