**GARBAGE GRABBEROOS – Jade adedokun**

candidate no: 120763032

2023

**AH Computing Project**

Contents

[ANALYSIS 3](#_Toc129942909)

[Description of Problem 3](#_Toc129942910)

[Advanced Higher concepts and integration 4](#_Toc129942911)

[Constraints 5](#_Toc129942912)

[UML Use Case Diagram 7](#_Toc129942913)

[Persona 7](#_Toc129942914)

[User Stories 8](#_Toc129942915)

[User Scenarios 8](#_Toc129942916)

[Requirements specification 9](#_Toc129942917)

[End-user requirements 9](#_Toc129942918)

[Functional requirements 9](#_Toc129942919)

[Project plan 17](#_Toc129942920)

[Initial Plan (Dates Estimated) 17](#_Toc129942921)

[Final Plan (Dates Confirmed) 21](#_Toc129942922)

[Gantt Chart 25](#_Toc129942923)

[Resources 26](#_Toc129942924)

[Appendix A – User Survey Results 27](#_Toc129942925)

[DESIGN 37](#_Toc129942926)

[Wireframes 37](#_Toc129942927)

[Data Dictionary 64](#_Toc129942928)

[Entity – Relationship Diagram 64](#_Toc129942929)

[Query Designs 64](#_Toc129942930)

[Query to create Users table 65](#_Toc129942931)

[Class Diagram 66](#_Toc129942932)

[Pseudocode 68](#_Toc129942933)

[Design Structure of 2D Array & Arrays of Records 90](#_Toc129942934)

[IMPLEMENTATION 91](#_Toc129942935)

[Program Code 91](#_Toc129942936)

[Sign Up Class 91](#_Toc129942937)

[Unsuccessful SignUp or Login Class 96](#_Toc129942938)

[Successful SignUp Class 96](#_Toc129942939)

[Account Existing Class 97](#_Toc129942940)

[Login Class 97](#_Toc129942941)

[No Account Found Class 103](#_Toc129942942)

[Game Menu Class 103](#_Toc129942943)

[How To Play Class 104](#_Toc129942944)

[Game Grid Class 105](#_Toc129942945)

[Instructions Class 114](#_Toc129942946)

[Sweet Wrapper Class 115](#_Toc129942947)

[Plastic Straw Class 115](#_Toc129942948)

[Crisp Packet Class 115](#_Toc129942949)

[Plastic Bottle Class 116](#_Toc129942950)

[Large Car Class 116](#_Toc129942951)

[Paused Class 116](#_Toc129942952)

[Losing Message Class 117](#_Toc129942953)

[Winning Message Class 117](#_Toc129942954)

[Speedy Searchers Board Class 118](#_Toc129942955)

[Screenshots of User Interface 122](#_Toc129942956)

[Research and development of new skills and/or knowledge 139](#_Toc129942957)

[Log of ongoing testing 141](#_Toc129942958)

[TESTING 160](#_Toc129942959)

[Final Test Plan 160](#_Toc129942960)

[Interface Testing Plan 160](#_Toc129942961)

[Final and Requirements Testing Plan 168](#_Toc129942962)

[Testing with Persona and Test Cases Plan 179](#_Toc129942963)

[End User Testing Plan 182](#_Toc129942964)

[Completed Final Testing 182](#_Toc129942965)

[Interface Testing Completed 182](#_Toc129942966)

[Final and Requirements Testing Completed 210](#_Toc129942967)

[Testing with Persona and Test Cases Completed 257](#_Toc129942968)

[End User Testing Completed 287](#_Toc129942969)

[EVALUATION 297](#_Toc129942970)

[Fitness for Purpose 297](#_Toc129942971)

[Results of Testing 298](#_Toc129942972)

[Log of Ongoing Testing 298](#_Toc129942973)

[Interface Testing 298](#_Toc129942974)

[Final and Requirements Testing – coupled with Integration Testing 299](#_Toc129942975)

[Testing with Personas, User Stories and Test cases 299](#_Toc129942976)

[End User Testing 299](#_Toc129942977)

[Future Maintainability 300](#_Toc129942978)

[Robustness 300](#_Toc129942979)

[Usability 300](#_Toc129942980)

# ANALYSIS

## Description of Problem

In Garbage Grabberoos, the user will try to find pieces of rubbish that are environmentally damaging hidden somewhere on an 11 by 10 grid, and discard them by clicking on all of the pieces of rubbish in the fastest time possible, and without running out of guesses. The quicker they locate all of the pieces, the quicker their time will be on the Speedy Searchers leaderboard.

I will integrate Software Design and Development with Database Design and Development in Garbage Grabberoos. The two areas of the course will link using a user database, where the username, password and userID of every user will be stored upon sign up. The project will feature the “2D array” detailed in the AH computing project specification using an 11 by 10 2D array that will form the Garbage Grabberoos game grid. The project will also feature the “array of records” detailed in the AH computing project specification using the ‘time’ array of records and the ‘date’ array of records.

When the user has successfully located and discarded all of the pieces of rubbish hidden on the game grid, they will have the time it took them and the date of completion written to a file, then read into arrays of records. The ‘time’ array of records will be bubble sorted (as detailed in project specification) from low to high, which will allow the data to be displayed in order of fastest to slowest time on the Speedy Searchers leaderboard.

The game’s end users will be upper primary students in geography classes/environmental groups as a means of learning about climate change and gaining an awareness of how they are treating the environment currently. They can have any level of technological experience, since the program will have a user interface which will be very easy to navigate, and an instruction button will be present in the game. It should present the issue of climate change in a relatively light-hearted way, encouraging them to take an active approach to removing rubbish from our planet in real life, as well as virtually. It will also help to inform about the sustainability goals detailed by UNICEF, particularly No. 13 which encourages us to “take urgent action to combat climate change and its impacts”, and most importantly it will educate the user’s about some of the most severe items causing climate change, through colourful infographics.

### Advanced Higher concepts and integration

The program will meet Advanced Higher project requirements since it will be an **integration of concepts from database design and development and software design and development.** This integration will take place as follows: Before playing the game, the user will have to sign up for an account. The sign up will be validated based on length of username (needs to be 8 characters) and length of password (needs to be 6 characters). Once these variables are validated, **the inputted username and password at sign up will be checked against the Garbage Grabberoos Users Database for an already existing account using a SELECT query**. If this account exists, the user will be prompted to login. **If this account does not exist, a new user will be inserted into the Garbage Grabberoos Database and have their username and password stored there, using an INSERT INTO query**. Lastly, at login, the user’s inputted username and password will be validated **and checked against the Garbage Grabberoos Users Database for an account that matches those details using a SELECT query**, in order for the user to be allowed to continue to the game menu. Once at the game menu, the user can either view the Speedy Searchers leaderboard or begin the game. If they begin the game, they will be shown the **11 by 10 game grid, which will be formed using a 2D array.** On the game grid, the input will be validated based on whether a user has already clicked that grid before as a guess, and if they have, an error message will be displayed that informs them of this. If the user is not able to find all of the pieces of rubbish before their number of guesses left reaches 0, then they will be shown a losing message and allowed to restart. However, if the user is able to find all of the pieces of rubbish before their guesses run out, then they will have the time taken for them to complete the game and the date of completion written to file, and be shown a winning message. **This file will subsequently be read into arrays of records**. **The ‘time’ array of records will be bubble sorted from low to high**, which will allow the data to be displayed in order of fastest to slowest time on the Speedy Searchers leaderboard. The project will also meet AH requirements since **procedural programming** will be used throughout.

#### List of Covered Mandatory Requirements:

Software Design & Development Project

*Concept 1*

* **Procedural programming** with a 2-D array AND array of records

*Concept 2*

* **Bubble sort** of an array of records

*Integration*

* **Integrated with a database containing one table**: one for player details (username and password)

### Constraints

Technical Constraints

* My knowledge of the topics that I have chosen to integrate (Database Design & Development and Software Design & Development) is limited to Advanced Higher level, so I will need to do extra research to find new concepts to implement that are beyond the Advanced Computing Science course in terms of difficulty
* I will use Visual Basic as the main programming language for my project. Using Visual Basic means that I will need to dedicate extra time to creating visually appealing forms, as this is typically harder on VB and the project’s target audience is children
* A portable version of XAMPP (contained on my flashdrive) will used to create and store the databases required for my project
* The size of my project cannot exceed 8GB, as this is the capacity of the flash drive that I use to transfer my progress between my home laptop and the school computers
* A backup copy of each aspect of my project (e.g. the VB code for my program) will need to be created in a Notepad++ document in case of the event of a misplaced flash drive

Time Constraints

* There is a short length of time for the completion of my project (6 months) due to its submission date of February 27th meaning that the number of features that I can incorporate into my project is limited, and I must adhere to my proposed project plan

Business Constraints

* I am limited by the capabilities of the operating system (Windows 10) on my school computers
* As the single developer of my project, I am limited by my own performance – any days that I am absent will affect the progress of my project, and my speed will determine how quickly it is completed
* The budget of my project is £0, meaning that I am only using the readily available and free school computers, as well as my own pre-owned personal laptop
* The budget of my project is £0, so I will only be using software that allows for the creation of free accounts, such as Survey Monkey and Gantt Project

Legal Constraints

* My program will not store the IP addresses of players or their forename/surname so I will not need to take account of GDPR Laws
* I will be creating all the structures present in the program myself (including graphics) so the Copyright, Design and Patents Act will not apply
* In order to adhere to GDPR laws, I will not be taking the emails, forenames or surnames of my users. This means that I will not be able to implement a “forgot password” feature

## UML Use Case Diagram

*Diagram

Description automatically generated*

\**Dotted lines have been used for all extend/include processes apart from those linked to the use cases associated with the “unregistered user” actor due to my Lucid Chart free subscription not covering this. They would otherwise be dotted lines.*

## Persona

Sara is 11 years old, and a passionate member of her school’s Environmental Club. Her favourite subjects at school are Geography and Computing Science, meaning that she is a relatively experienced user of computerised devices, and finds particular enjoyment in any computer software that incorporates her love of sustainability and environmentalism. As an academic overachiever with a streak of perfectionism, she enjoys playing simple video games like *Crossy Road* or *Flappy Bird* to unwind whenever she completes the large number of both homework and extracurricular tasks (such as hockey or basketball) that she takes on daily. Unfortunately for Sara, the pocket money that she receives from her parents for doing household chores does not cover much more than a few bars of chocolate on tiring days after school, meaning that she cannot afford to spend money on video games. While this means that free games are the only ones readily available to her, she is still picky and expecting of features like background music, and a computerised opponent. This may be because as the youngest child of three she is used to getting what she wants, the perfect free video game included.

My persona is based on a Primary 7 girl with a variety of extracurricular activities on her roster and a satisfactory knowledge of both how to use computers and the effects of climate change, because as can be seen from my survey – this is the target demographic of my program.

### User Stories

*Story 1*

As a young girl who is busy with homework and other extracurriculars, I want to be able to learn more about actions that I can take to prevent climate change in a format that doesn’t feel as rigid and academic as a paper-based task like reading a newspaper, so that I am not overwhelmed by information after a school day, and take it in with a brain that is relaxed and ready to be informed.

*Story 2*

As an avid environmentalist, I want to be able to play a slightly different version of the game each time, (different sequence of trash on the grid) which will allow me to become so accustomed to the act of throwing away different types of trash in different locations by virtual means that it becomes second nature in real life too.

*Story 3*

As a very passionate video game user, I want to be able to view my best times whenever I want, in order to be able to beat them with more practice.

*Story 4*

As a primary school student who plays several video games, I want to be able to make a fun and creative username to sign up with so I can be identified by it whenever I login.

### User Scenarios

*Scenario 1*

Sara is an athlete in P7 who plays basketball, hockey and badminton. Playing these sports to a high level has given her something of a competitive personality, which translates to her preference in video game features. Recently, Sara’s sports have finished running as they are all winter games and summer is closely approaching. This has left her without an outlet for her competitive spirit, meaning that she has turned to video games. She opens Garbage Grabberoos because she is feeling particularly bored, and knows that it will allow her to compete against someone else. The random rubbish positions prove to be a challenge to beat, but she manages to win. She feels that the game provided her with an incentive to score highly and finish quickly, just like the sports she is so fond of. She clicks off the game and onto the Speedy Searchers leader board, and relishes in the fact that her success has been acknowledged by the ranking system of times, and feels that this is a good substitute for the feeling of coming first in real life.

## Requirements specification

### End-user requirements

(See [Appendix A](#_Toc114751076) for reference)

Using the results of my user survey and the information in my persona and user stories, I have identified the following end-user requirements:

* End-users want to have their winning score times stored within the game
* End-users want the user interface to have a visually pleasing colour scheme and be easy to navigate
* End users want the user interface to be gender neutral so that it can appeal to everybody
* End-users want to learn how to recycle effectively, and know which pieces of rubbish to look out for
* End users would like to practice recycling virtually in order to replicate it more often in real life
* End-users want to learn more about climate change and the role that certain products have in catalysing it
* End-users want the game to have a short duration so that they can fit into their daily lives
* End-users want the topics of recycling, sustainability and vegetarianism to be incorporated into the game
* End-users want the game to pose a standard level of difficulty
* End users want to learn more facts about the environment and specific rubbish pieces

### Functional requirements

Using my proposed project plan and problem specification, I have determined that the system should:

* Be able to connect to the Garbage Grabberoos User Database
* Allow for a user to make an account that will update the Garbage Grabberoos User Database by adding their username and password with an INSERT INTO query
* Check the Garbage Grabberoos User Database for an existing account that corresponds to the user’s username and password during the sign up process with a SELECT query
* Validate the user’s entry for username and password upon both sign up and login
* Allow for a user to log on using the username and password associated with their created account by checking the Garbage Grabberoos User Database using a SELECT query
* Contain a grid (made up of an 11 by 10 2D array) that can be clicked on by the user
* List user winning scores sorted from low to high in a leader board that can be visited throughout gameplay
* Allow for the grid squares that have been clicked on to be remembered, so the user cannot use two guesses on the same square and will instead be displayed an error message if this happens
* Time each game in order to be able to write this to a file and then add this time to the time array of records if the user wins the game
* Contain forms that act as infographics and educate users on the rubbish they pick up during gameplay
* Contain buttons that will allow the user to pause and restart the game

##### Inputs, Outputs and Processes

The first form that the user will see will allow them or sign up to make a username and password, and both entries will be validated by length as well as checked against the Garbage Grabberoos User Database for an already existing account that is linked to these details. If there is no already existing account, then a new account will be created for this user and the details will be added to the Garbage Grabberoos User Database.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game  Sign-up | User will enter:   * An 8-character username of their choice * A 6-character password of their choice   User will then make a mouse input by clicking on the sign-up button  User can click on the link label on the sign up form to open the login form | When the sign up button is clicked on the input will be validated with:  A length check on username to check that it is 8 characters long  A length check on password to check that it is 6 characters long  A presence check on the username and password to ensure that they have both been entered and neither text box is empty  A SELECT query of the of Garbage Grabberoos Database to check to see if there is an already existing account that matches the entered and validated username and password at sign up  An INSERT INTO query of the Garbage Grabberoos Database to enter the username and password of a new user into the system if they do not already have an existing account linked to these details  If the link label is clicked then the login form will become visible and enabled and the sign up form will be hidden and disabled | If the entered username or password is invalid (not 6 characters for password or not 8 characters for username): An error message (“Please enter another username and/or password that matches the given criteria”) will be displayed  If one or both of the text boxes is empty, the same error message (“Please enter another username and/or password that matches the given criteria”) will be displayed  If an already existing account that matches the entered username and password is found, then an error message: “This account already exists. Please login,” will be displayed  If the entered username and password are valid and there is no previously existing account that corresponds to these details, then the user will be shown a successful sign up message  If the link label is clicked then the login form will be displayed |

The user will be able to login with their created username and password after a successful sign up that is then exited, or using the link label on the sign up form. The entries of username and password will both be validated by length, and checked against the Garbage Grabberoos Database for an account that corresponds to those details. If the account is found, then the user will be taken to the game menu after a successful login message and welcome message are both shown.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game  Login | User will enter:   * An 8-character username that they have previously created * A 6-character password that they have previously created   User will then make a mouse input by clicking on the login button  User can click on the link label on the login form to return to the sign up form | When the login button is clicked on the input will be validated with:  A length check on username to check that it is 8 characters long  A length check on password to check that it is 6 characters long  A presence check on the username and password to ensure that they have both been entered and neither text box is empty  A SELECT query of the of Garbage Grabberoos Database to check to see if there is an already existing account that matches the entered and validated username and password  If the link label is clicked then the sign up form will become visible and enabled and the login form will be hidden and disabled | If the entered username or password is invalid (not 6 characters for password or not 8 characters for username): An error message (“Please enter another username and/or password that matches the given criteria”) will be displayed  If one or both of the text boxes is empty, the same error message (“Please enter another username and/or password that matches the given criteria”) will be displayed  If no already existing account that matches the entered and validated username and password is found, then an error message: “This account does not exist. Please sign up or try again,” will be displayed  If an already existing account that matches the entered and validated username and password is found, then a message: “Welcome back to Garbage Grabberoos” & their validated username will be shown, before being taken to the game menu  If the link label is clicked then the sign up form will be displayed |

The user will be brought to a game menu after logging in, and the game menu will contain three buttons: one that links to the Speedy Searchers leaderboard, one that links to the Garbage Grabberoos game grid (for playing the game), and one that quits the game.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| View Leaderboard | User will make a mouse input on the game menu by pressing on the “View leaderboard” button  User can scroll along on the leaderboard to see their slower times  User can click on a link label to return to the game grid | The integrated ‘*Winning Times’* text file will be read into two arrays of records: one for dates and one for times  The time array of records will be bubble sorted from low to high (fastest to slowest winning time and this will determine the order of the data shown in the list box of the Speedy Searchers board | A list box will be shown containing the bubble sorted (from fastest to slowest winning time) details of each of the user’s winning time (the time itself & the date of the win) for the user to see and scroll along on  This list box will be present on the Speedy Searchers leaderboard form |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Open Garbage Grabberoos | User will make a mouse input on the game menu by pressing on the “Play Game” button  User will click on the Quit button of the ‘How to play’ form | Upon clicking the “Play Game” button, the ‘How to play’ form will be displayed on the screen to give the user instructions on how to navigate the game  Once the user has pressed on the quit button of the ‘How to play form’, the Garbage Grabberoos game grid will be shown on the screen to allow the user to make guesses on the grid | The first output will be a colourful form which will be displayed with instructions on it called ‘How to Play’  The second output will be the gameGrid form, which will be displayed on the user’s screen and contain the grid that the user will guess the location of rubbish pieces on |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| End Garbage Grabberoos | User can make a mouse input on the game menu by pressing on the “Quit Game” button  User can click on an exit button during sign up or login | The program will end, and all forms will be stopped. | No output |

The user will be able to perform two actions during game play that will need to be validated, and require the use of an integrated database, which are discarding their own trash and guessing the locations of the computer’s trash.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  Guessing Rubbish Piece Locations | The user will click on a square on the 11 by 10 grid  The user will repeat this action at least 12 times (most likely more) to successfully guess the location of 12 pieces of trash | The user’s click on the grid will be validated by:  Checking that the grid has not already been guessed previously  Checking whether a rubbish piece is present on the clicked on grid square  If a rubbish piece is present, then the grid square colour will be changed from white to the specific colour associated with that type of rubbish (eg. pink for a plastic bottle)  The infographic about that type of rubbish will then be displayed | An error message: (“You have already clicked this cell,”) will be displayed if the user has previously clicked on the grid square  For every successful (rubbish piece present) guess that the user makes, a placeholder colour is added to the grid square to represent piece of rubbish, and to allow the user to remember which grid squares have been guessed thus far  For every successful (rubbish piece present) guess that the user makes, a form will be shown that contains information about that piece of rubbish |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  Restart Garbage Grabberoos  Game | User will make a mouse input on the game grid form by pressing on the “Restart” button | The Garbage Grabberoos grid will be reloaded, and all controls will be reset to their original values (including number of guesses, rubbish count etc.) | The gameGrid form will be displayed on the user’s screen and it will contain the original empty grid |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  Pause  Game | User will make a mouse input on the game grid form by pressing on the “Pause” button | The Garbage Grabberoos game grid will be frozen (i.e., the user will not be able to interact with it  A “Paused” message will be displayed on the screen | The gameGrid form will remain displayed on the user’s screen but the user will be unable to interact with it or click any grid squares  The pausedForm will be displayed on the user’s screen and contain a button which can be pressed to continue the game |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  End  game | User will make a mouse input on the game grid form by pressing on the “quit” button | The program will end, and the game grid form will be stopped. | No output |

If the user manages to find all of the pieces of rubbish before their guesses run out, then their time will be inserted into the Speedy Seachers leaderboard which is re-sorted from low to high upon each new entry (fastest to slowest time), and they will be shown a congratulatory message.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  Winning Game | The user will win the game by correctly guessing all the locations of the pieces of rubbish before their allotted number of guesses (70) reaches 0. | A message (“YOU WIN”) will be displayed on the user’s screen if their number of guesses left is not 0 and the number of rubbish pieces that they have found is 15  The time taken for them to guess the locations of all of the pieces of rubbish along with the date that the win occurred on will be added to a text file titled *‘winningTimes.txt’*  The ‘w*inningTimes.txt’* text file will then be read into two arrays of records (one for time and one for date) where the time array of records will be ordered from low to high with a bubble sort, then extracted into the program for viewing  (see View Leaderboard in Functional Requirements: Inputs, Outputs, Processes) | The winningForm will be displayed on the user’s screen if they are successful, and this will contain a button that allows the user to restart the game and view leaderboard (which will now include their new time) |

If the user does not manage to find all of the pieces of rubbish before their guesses run out, then their time won’t be inserted into the Speedy Seachers leaderboard, and they will be shown a losing message.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inputs | Processes | Outputs |
| Game Play:  Losing Game | The user will lose the game if they are unable to successfully locate all of the pieces of rubbish before their allotted number of guesses (70) reaches 0. | A message (“YOU LOST”) will be displayed on the user’s screen if their number of guesses left is 0 and the number of rubbish pieces that they have found is not 15 | The losingForm will be displayed on the user’s screen if they are unsuccessful, and this will contain a button that allows the user to restart the game and view leaderboard (which will allow them to see old winning times) |

## Project plan

### **Initial Plan (Dates Estimated)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Analysis** *Overall Duration: 24/08/22 – 22/09/22* | | | | | | | |
|  | Project Research | 20/08/22 | 23/08/22 | Sequential |  | UNICEF and the Sustainable Development Goals |
|  | Description of problem | 24/08/22 | 27/08/22 | Sequential | 1 | Microsoft word |
|  | Create user survey | 28/09/22 | 01/09/22 | Sequential | 1,2 | Survey Monkey – create free account  <https://www.surveymonkey.com/> |
|  | Distribute user Survey | 03/09/22 | 10/09/22 | Sequential | 3 | Microsoft Teams  Primary school pupils |
|  | Collate and analyse survey data | 12/09/22 | 13/09/22 | Sequential | 3,4 | Survey Monkey (free account created) |
|  | Identify Constraints | 16/09/22 | 18/09/22 | Sequential |  | Software Design and Development Textbook – Feasibility Page |
|  | UML Use case diagram | 24/09/22 | 28/09/22 | Sequential |  | LucidChart – create free account  [https://www.lucidchart.com/](https://www.lucidchart.com/pages/)  <https://youtu.be/zid-MVo7M-E?t=33> |
|  | Create Persona and user stories | 13/09/22 | 16/09/22 | Sequential | 5 | Microsoft Word  Results of User Survey ([Appendix A](#_Toc114751076)) |
|  | Requirements specification:  end-user  requirements | 20/09/22 | 22/09/22 | Sequential | 1,5 | Microsoft Word  Results of User Survey  ([Appendix A](#_Toc114751076)) |
|  | Requirements specification:  functional  requirements | 27/09/22 | 10/10/22 | Sequential | 1,5,9 |  |
|  | Create a  Project Plan | 04/09/22 | 08/09/22 | Parallel |  | Gantt Project – create free account: [Introduction into GanttProject 2.5 - YouTube](https://www.youtube.com/watch?v=5rHCSa5ad34) |
| **Design** *Overall Duration: 22/09/22 – 26/10/22* | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Create Wireframes (UI Design) | 29/09/22 | 05/10/22 | Sequential | 10 | Balsamiq |
|  | UML class diagram | 30/09/22 | 01/10/22 | Sequential |  | LucidChart – create free account  [https://www.lucidchart.com/](https://www.lucidchart.com/pages/) |
|  | Database Entity-relationship diagram | 04/10/22 | 04/10/22 | Sequential |  | Microsoft Word |
|  | Create a Data dictionary | 02/10/22 | 04/10/22 | Sequential | 12, 15 | Microsoft Word |
|  | Database Query Design | 08/10/22 | 10/10/22 | Sequential |  | MySQL – contained on XAMPP  MariaDB (for testing before implementation in mySQL) |
|  | Code design in Pseudocode | 12/10/22 | 24/10/22 | Sequential | 2,10,12 |  |
| **Implementation**  *Overall Duration: 03/10/22 – 21/01/22* | | | | | | |
|  | Complete program code incl. AH concepts | 17/10/22 | 20/01/22 | Parallel | 17,22 | Visual Basic 2012 |
|  | Print Program Code | 21/01/22 | 21/01/22 | Sequential | 18,19,20 | Snipping Tool |
|  | Complete SQL Code for Leaderboard | 19/10/22 | 30/11/22 | Sequential |  | Microsoft Access |
|  | Print SQL Code | 15/01/22 | 15/01/22 | Sequential | 22 | Snipping Tool |
|  | Write description of new skills and knowledge acquired | 03/10/22 | 20/01/22 | Parallel |  | List of new skills acquired  Microsoft Word |
|  | Make note of ongoing testing of program | 10/10/22 | 0/01/22 | Parallel |  | Microsoft Word  Stack Overflow |
| **Testing** *Overall Duration: 12/01/22 – 23/01/22* | | | | | | |
|  | Write a description of final test plan | 12/01/22 | 16/01/22 | Sequential |  |  |
|  | Execute Test Plan | 19/01/22 | 19/01/22 | Sequential | 27 | Primary School Pupils |
|  | Gather results of persona and test cases | 19/01/22 | 23/01/22 | Sequential |  |  |
| **Evaluation** *Overall Duration: 2 days (24/01/22 – 26/01/22)* | | | | | | |
|  | Evaluation Report - Fitness for Purpose | 24/01/22 | 26/01/22 | Sequential | 9,10 |  |
|  | Evaluation Report – Maintainability and Robustness | 24/01/22 | 26/01/22 | Sequential | 28,29 |  |
| **Finalisation** *Overall Duration: 27/01/22 – 28/01/22* | | | | | | |
| 30. | Collate Evidence in Ring Binder | 27/01/22 | 28/01/22 | Sequential |  | Ring Binder, Subject Dividers |

### **Final Plan (Dates Confirmed)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Analysis** *Overall Duration: 24/08/22 – 10/10/22* | | | | | | | |
|  | Project Research | 24/08/22 | 27/08/22 | Sequential |  | UNICEF and the Sustainable Development Goals |
|  | Description of problem | 25/08/22 | 26/08/22 | Sequential | 1 | Microsoft word |
|  | Create user survey | 29/09/22 | 03/09/22 | Sequential | 1,2 | Survey Monkey – create free account  <https://www.surveymonkey.com/> |
|  | Distribute user Survey | 05/09/22 | 10/09/22 | Sequential | 3 | Microsoft Teams  Primary school pupils |
|  | Collate and analyse survey data | 14/09/22 | 16/09/22 | Sequential | 3,4 | Survey Monkey (free account created) |
|  | Identify Constraints | 20/09/22 | 21/09/22 | Sequential |  | Software Design and Development Textbook – Feasibility Page |
|  | UML Use case diagram | 27/09/22 | 31/09/22 | Sequential |  | LucidChart – create free account  [https://www.lucidchart.com/](https://www.lucidchart.com/pages/)  <https://youtu.be/zid-MVo7M-E?t=33> |
|  | Create Persona and user stories | 10/09/22 | 17/09/22 | Sequential | 5 | Microsoft Word  Results of User Survey ([Appendix A](#_Toc114751076)) |
|  | Requirements specification:  end-user  requirements | 24/09/22 | 26/09/22 | Sequential | 1,5 | Microsoft Word  Results of User Survey  ([Appendix A](#_Toc114751076)) |
|  | Requirements specification:  functional  requirements | 27/09/22 | 10/10/22 | Sequential | 1,5,9 |  |
|  | Create a  Project Plan | 04/09/22 | 08/09/22 | Parallel |  | Gantt Project – create free account: [Introduction into GanttProject 2.5 - YouTube](https://www.youtube.com/watch?v=5rHCSa5ad34) |
| **Design** *Overall Duration: 02/11/22 – 25/12/22* | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Create Wireframes (UI Design) | 02/11/22 | 14/11/22 | Sequential | 10 | Balsamiq |
|  | UML Class diagram | 13/11/22 | 18/11/22 | Sequential |  | LucidChart – create free account  [https://www.lucidchart.com/](https://www.lucidchart.com/pages/) |
|  | Database Entity-relationship diagram | N/A | N/A | Sequential |  | Microsoft Word |
|  | Create a Data dictionary | 29/11/22 | 04/12/22 | Sequential | 12, 15 | Microsoft Word |
|  | Database Query Design | 08/12/22 | 17/12/22 | Sequential |  | Microsoft Access |
|  | Code design in Pseudocode | 19/12/22 | 25/12/22 | Sequential | 2,10,12 |  |
| **Implementation**  *Overall Duration: 3 Months (09/01/23 – 20/03/23)* | | | | | | |
|  | Complete program code incl. AH concepts | 09/01/23 | 12/03/23 | Parallel | 17,22 | Visual Basic 2012  [Plastic Straw Facts](https://www.worldwildlife.org/magazine/issues/summer-2018/articles/a-small-straw-s-big-environmental-impact)  [Plastic Straw Facts](https://www.aza.org/connect-stories/stories/how-do-straws-hurt-the-environment?locale=en#:~:text=Most%20plastic%20straws%20simply%20break,%2C%20people%2C%20and%20the%20environment)  [Plastic Bottle Facts](https://healthyhumanlife.com/blogs/news/plastic-water-bottle-pollution-plastic-bottles-end)  [Sweet Wrapper Facts](https://www.scmp.com/comment/letters/article/3101940/candy-makers-must-face-bitter-truth-about-their-plastic-pollution)  [Crisp Packet Facts](https://www.fswaste.co.uk/can-you-recycle-crisp-packets/#:~:text=As%20a%20result%2C%20the%20vast,30%20years%20out%20at%20sea) |
|  | Print Program Code | 18/03/23 | 20/03/23 | Sequential | 18,19,20 | Snipping Tool |
|  | Complete SQL Code for Leaderboard | 14/02/23 | 27/02/23 | Sequential |  | Microsoft Access |
|  | Print SQL Code | 19/03/23 | 19/03/22 | Sequential | 22 | Snipping Tool |
|  | Write description of new skills and knowledge acquired | 12/03/23 | 14/03/22 | Parallel |  | List of new skills acquired - Notepad  Microsoft Word  StackOverflow  CodeProject |
|  | Make note of ongoing testing of program | 09/01/23 | 12/03/23 | Parallel |  | Microsoft Word  Stack Overflow  CodeProject |
| **Testing** *Overall Duration: 10/03/22 – 16/03/22* | | | | | | |
|  | Write a description of final test plan | 10/03/23 | 14/03/22 | Sequential |  |  |
|  | Execute Test Plan | 15/03/23 | 15/03/23 | Sequential | 27 | Primary School Pupils (used for end user testing) |
|  | Gather results of persona and test cases | 16/03/22 | 16/03/22 | Sequential |  |  |
| **Evaluation** *Overall Duration: 16/03/23 – 16/03/23* | | | | | | |
|  | Evaluation Report - Fitness for Purpose | 16/03/23 | 16/03/23 | Sequential | 9,10 |  |
|  | Evaluation Report – Maintainability and Robustness | 16/03/23 | 16/03/23 | Sequential | 28,29 |  |
| **Finalisation** *Overall Duration: 18/03/23 – 18/03/23* | | | | | | |
| 30. | Collate Evidence in Ring Binder | 18/03/23 | 18/03/23 | Sequential |  | Ring Binder, Subject Dividers |

### Gantt Chart

Analysis

Teams

Description automatically generated with medium confidence

Design

Chart

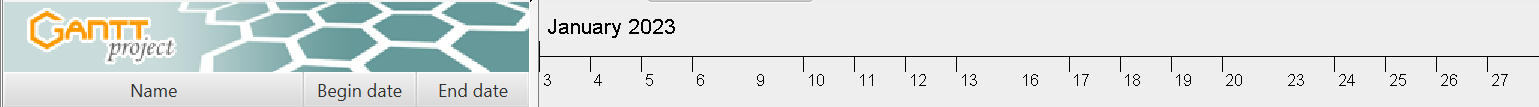
Description automatically generated

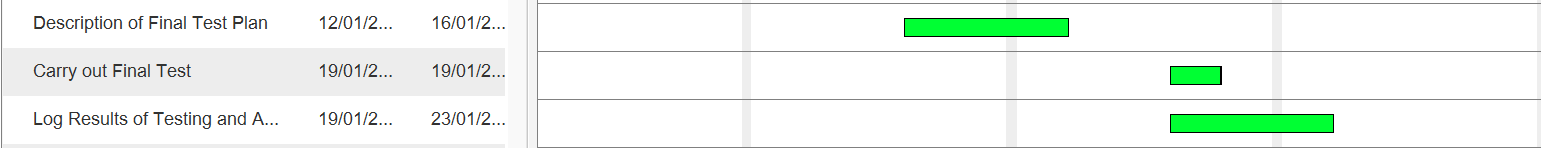
Implementation

A picture containing graphical user interface

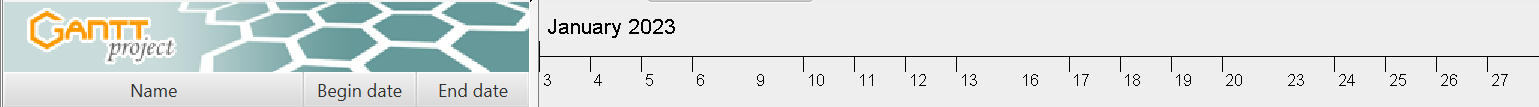
Description automatically generated

Testing





Evaluation

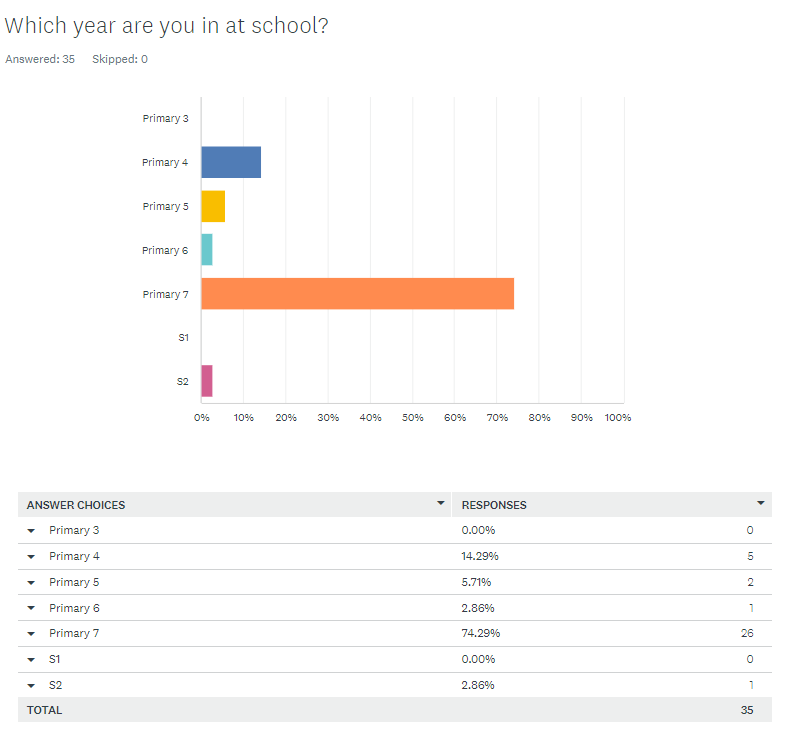




### Resources

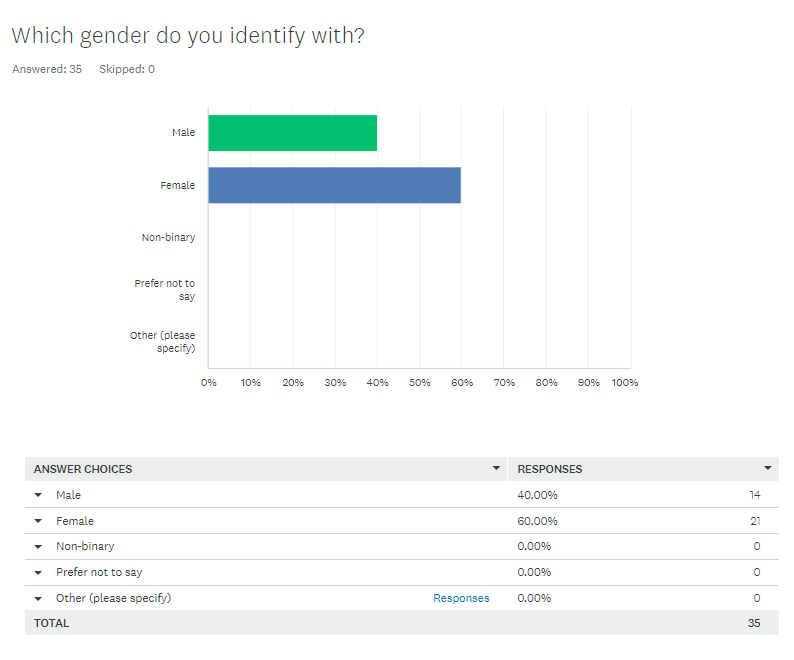
* [UNICEF and the Sustainable Development Goals](https://www.unicef.org/sdgs#sdg13)
* Primary School Pupils
* Microsoft Word – Office 365 account needed
* Gantt Project
* Lucid Chart
* CodeProject
* Microsoft Access – Office 365 account needed
* Youtube
* Stack Overflow
* Git Hub
* Balsamiq
* Notepad++
* SurveyMonkey
* Microsoft Teams – Office 365 account needed
* A4 Ring Binder
* Subject Dividers
* School Computers (all running on Windows 10 or above)
* Home Laptop
* Visual Basic 2012
* Snipping Tool

## Appendix A – User Survey Results



**Analysis**

Most of my end-users are at the end of their primary school education. This means that they will have had experience with playing video games, either in the computing classes taken at their schools or through using their own personal technology, such as a mobile phone or PC. This means that I can program the game to have a moderate difficulty level, but will need to create a “How to Play” instructional page in order for end-users to understand how to play the game.



**Analysis**

The majority of my end-users are female (60%), but the percentage of male end-users are not far behind at 40%. This means that I will design the game to be gender neutral in terms of graphics and interface, since using my large sample size it has been demonstrated that an almost equal number of end users of both genders will be playing the game. Both genders of end-users will want to feel engaged and included, so a gender-neutral game is the best way to navigate this.

Chart

Description automatically generated

**Analysis**

The majority of my end-users are recycling on a usual basis, which means that after using or eating something, throwing away their litter responsibly is a concept that occurs to them. However, 9 end users responded that they only recycle “sometimes”, which belies an inconsistency in their habits of picking up and discarding rubbish. This means that there is still a clear space for improvement in the rubbish disposal routine of my end users and informs me that steps should be taken to make sure that my end users are recycling in all scenarios. I will program Garbage Grabberoos in the hopes that if my end users play it regularly, then the act of throwing away rubbish will become automatic, and habitual, allowing them all to eventually be able to respond that they recycle “always”.

Chart

Description automatically generated

**Analysis**

The majority of my end-users only have a satisfactory or moderate knowledge of climate change, and therefore the topics that fall under this heading (including recycling). This means that there is still more to be learnt in order for them to become well informed environmentalists. This means that I will make sure to incorporate an educational element to the game, and have an infographic about a rubbish piece pop up whenever one of the rubbish pieces is picked up.

Chart

Description automatically generated

Analysis

Most of my end users (55%) deem recycling to be the aspect of environmental action that they are most interested in. However, sustainability is not very far behind at 40%. This is useful information as it informs which rubbish pieces that I will create for the purpose of being discarded, such as a plastic bottle, a sweet wrapper and a car with high carbon dioxide emissions. This will allow me to engage all of my end users since it will combine multiple topics within environmental action.

Chart

Description automatically generated

Analysis

The majority of my end users spend a short period of time playing video games every day. This means that I will create the game so that it can be played to completion in a relatively short amount of time (around 5 minutes). I will do this by only requiring a small amount of trash (5 pieces) to be discarded onto grids and then subsequently found. This will allow end-users to make the most of the brief time that they dedicate to playing video games.

Chart

Description automatically generated

**Analysis**

Most of my end users play video games on their mobile phone, Nintendo switch or Tablet/iPad. Since these are all devices of varying screen sizes, I will create forms that will always be centred on the user’s screen, regardless of what kind of screen they use. This means that all of my end-users will be able to enjoy the game fully, and my program will adhere to accessibility rules.

Chart

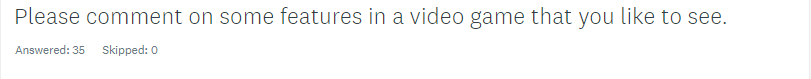
Description automatically generated with medium confidence

Analysis: This survey result shows that my end users all have varying levels of experience with navigating video games, which reiterates the point that the game will need to feature a ‘How to play’ page, as well as a help button that can be accessed throughout gameplay. Those who deem themselves to be ‘extremely confident’ only need to view the ‘How to play’ page once, while those between ‘somewhat confident’ and ‘not at all’ confident will be able to seek help as often as they need. This will make the game usable for all end users.

Chart

Description automatically generated

This survey result shows that the majority of my end users enjoy playing video games against other people, and are therefore competitive in nature. This means that the game will satisfy my end users using its competitive elements, such as a leader board that allows them to view the fastest winning times that they have achieved over their entire game history.



Graphical user interface, text, application, email

Description automatically generated

A picture containing shape

Description automatically generated

Background pattern

Description automatically generated with low confidence

This survey result demonstrates that graphics are important to my end users, and that they expect a coherent and engaging colour scheme. I aim to do this by using colours that are relevant but still pleasing to the eye (green, blue etc.) and making the rubbish piece infographics as visually detailed as possible. One end user says that they “want to learn more about mother nature”, showing how learning facts and increasing their education on climate change is something that appeals to my end users. Because of this, I will take care to make sure that learning happens during gameplay, and that end-users will finish the game knowing more about climate change than they did at the start

# DESIGN

## Wireframes

**Sign Up**

Diagram

Description automatically generated

*Successful Sign Up*

**Diagram

Description automatically generated**

*Invalid Sign Up – empty textbox, non 8-character username and/or non 6-character password*

**Diagram

Description automatically generated**

*Invalid Sign Up – existing account in database that matches entered details*

**Graphical user interface, diagram, application

Description automatically generated**

**Login**

**Diagram

Description automatically generated**

*Invalid Login - empty textbox, non 8-character username and/or non 6-character password*

*Diagram

Description automatically generated*

*Invalid Login – no account found in database that matches entered details*

*Graphical user interface, application

Description automatically generated*

*Successful Login (welcome message + access to homepage)*

*Diagram

Description automatically generated*

**Diagram

Description automatically generated**

**Game Play**

*Start game button clicked – instruction form displayed*

*Graphical user interface, text, application, email

Description automatically generated*

*Empty game grid (at game start)*

*Text

Description automatically generated with low confidence*

*Successful guess of sweet wrapper grid square on game grid*

*Chart

Description automatically generated*

*Sweet wrapper infographic (displayed after successful guess depicted above)*

*Diagram

Description automatically generated with medium confidence*

*Successful guess of plastic straw grid square on game grid*

*Chart

Description automatically generated*

*Plastic straw infographic (displayed after successful guess depicted above)*

*Graphical user interface

Description automatically generated with medium confidence*

*Successful guess of crisp packet grid square on game grid*

*Table

Description automatically generated with medium confidence*

*Crisp packet infographic (displayed after successful guess depicted above)*

*Graphical user interface

Description automatically generated with medium confidence*

*Successful guess of plastic bottle grid square on game grid*

Chart, whiteboard

Description automatically generated

*Plastic bottle infographic (displayed after successful guess depicted above)*

Diagram

Description automatically generated

*Successful guess of large car grid square on game grid*

*A picture containing table

Description automatically generated*

*Large car infographic (displayed after successful guess depicted above)*

*Graphical user interface

Description automatically generated*

*Double clicked on the same grid square – error message*

*Chart

Description automatically generated with low confidence*

*Clicked on the pause button – paused game message*

*Diagram

Description automatically generated with medium confidence*

*Help Button Clicked – Instructions Page*

*Graphical user interface, application, Excel

Description automatically generated*

*Discarded all 15 pieces of rubbish – winning message*

Text

Description automatically generated

*Failed to discard all 15 pieces of rubbish – losing message*

Graphical user interface, text

Description automatically generated

**Viewing Speedy Searchers Leaderboard**

Diagram, text

Description automatically generated

## Data Dictionary

Garbage Grabberoos Users Database – **Users Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Key | Required | Size | Validation |
| Username | Varchar(8) |  | Yes | 8 | Length Check = 8 |
| Password | Varchar(6) |  | Yes | 6 | Length Check = 6 |
| UserID | Int | PK | Yes | 100 | Autonumber |

## Entity – Relationship Diagram

There is no requirement for an entity-relationship diagram as my database only contains a single table.

## Query Designs

Checks if the username and password entered by the user correspond to an already existing account when signing in

|  |  |
| --- | --- |
| **SELECT** | \* |
| **FROM** | users |
| **WHERE** | username = validated inputted username AND password = validated inputted password |

Adds a new user into the users table of the Garbage Grabberoos Users Database in the event of a successful sign up

|  |  |
| --- | --- |
| **INSERT INTO** | users |
| **VALUES** | username, password |

Checks for an existing account during login using a validated user input of username and password

|  |  |
| --- | --- |
| **SELECT** | \* |
| **FROM** | users |
| **WHERE** | loginUsername = validated inputted username AND loginPassword = validated inputted password |

## Query to create Users table

CREATE TABLE Users(

Name VARCHAR(8) NOT NULL,

Password VARCHAR(6) INT NOT NULL)

userID INT PRIMARY KEY NOT NULL AUTO\_INCREMENT,

## Class Diagram

name: String

Login

+ checkUsername() + checkPassword()

- username: String - password: String

+ checkUsername() + checkPassword()

- username: String - password: String

SignUp

speedySearchersBoard

gameGrid

- DelimitedData: string - usertime(): string - userdate(): string

- grid(9,9): integer - rubbishCount: integer - SW: stopwatch - Tmr: windows timer - time: string - paused: boolean

+ SplitArrays() + BubbleSortByTime()

+ placeSweetWrapper() + placePlasticStraw() + placeCrispPacket() +placePlasticBottle() + placeLargeCar()

## Pseudocode

**Class Sign Up**

Declare usernamee as string

Declare passwordd as string

Set the usernamee variable to the value returned by the function CheckUsername

Set the password variable to the value returned by the function validPassword

Function checkUsername

Set the value of the variable usernamee to the user’s input for username in the corresponding textbox

Declare valid as Boolean = false

Initially hide and disable the unsuccessful sign-up or login error message

Do

If the length of the usernamee variable is 8 characters

Set valid to true

Else

Set valid to false

Display and enable the unsuccessful sign-up or login error message for 3 seconds using a timer

Disable and hide the existing account error message

Exit the function

End If

If the length of the username variable is 0 characters or the username textbox of the sign up form is empty then

Display and enable the unsuccessful sign-up or login error message

Set valid to false

End If

Loop until valid is true

Return the usernamee variable

End Function

Function validPassword

Set the value of the variable passwordd to the user’s input for passsword in the corresponding textbox

Declare valid as Boolean = false

Initially hide and disable the unsuccessful sign-up or login error message

Initially hide and disable the successful sign-up message

If the length of the passwordd variable is 8 characters

Set valid to true

Else

Set valid to false

Display and enable the unsuccessful sign-up or login error message for 3 seconds using a timer

Disable and hide the existing account error message

Exit the function

End If

If the length of the passwordd variable is 0 characters or the password textbox of the sign up form is empty then

Display and enable the unsuccessful sign-up or login error message

Set valid to false

End If

Loop until valid is true

If valid = true

Call the new user function

End If

Return the passwordd variable

End Function

When the timer ticks

Disable Timer 1

Hide and disable the sign-up/login error message

Show and enable the sign up form

End Sub

Sub Procedure newUser

Try

Establish connection to database

Set query to (“SELECT \* FROM users WHERE username = the validated username variable AND password = the validated password variable)

Execute query

If a row is retrieved from the user database

The existing account error message is shown and enabled

Exit procedure

The successful sign-up message is hidden and disabled

Close the database connection

Else

Set query1 to (“INSERT INTO users VALUES (the validated username variable, the validated password variable)

Execute query

Close the database connection

The successful sign-up message is shown and enabled

End If

Catch any unexpected or runtime errors and show an error message

End Sub

When the linklabel is clicked

The login form is shown and enabled

The sign-up form is hidden and disabled

The unsuccessful sign-up or login message is hidden and disabled

End Sub

When the exit button is clicked

End the program

End Sub

When the sign up form is loaded

Centre the sign-up form on screen

End Sub

End Class

**Class Login**

Declare loginUsername as string

Declare loginPassword as string

Declare loginUsernamee as string

Declare loginPasswordd as string

Set the loginUsername variable to the value returned by the function CheckUsername

Set the loginPassword variable to the value returned by the function validPassword

Call the checkUserDatabase function

Function checkUsername

Set the value of the variable loginUsernamee to the user’s input for username in the corresponding textbox

Declare valid as Boolean = false

Initially hide and disable the unsuccessful sign-up or login error message

Do

If the length of the loginUsernamee variable is 8 characters

Set valid to true

Else

Set valid to false

Display and enable the unsuccessful sign-up or login error message for 3 seconds using a timer

Exit the function

End If

If the length of the loginUsernamee variable is 0 characters or the username textbox of the login form is empty then

Display and enable the unsuccessful sign-up or login error message

Set valid to false

End If

Loop until valid is true

Return the loginUsernamee variable

End Function

Function validPasssword

Set the value of the variable loginPasswordd to the user’s input for password in the corresponding textbox

Declare valid as Boolean = false

Initially hide and disable the unsuccessful sign-up or login error message

Do

If the length of the loginPasswordd variable is 6 characters

Set valid to true

Else

Set valid to false

Display and enable the sign-up/login error message for 3 seconds using a timer

Exit the function

End If

If the length of the loginPasswordd variable is 0 characters or the password textbox of the login form is empty then

Display and enable the unsuccessful sign-up or login error message

Set valid to false

End If

Loop until valid is true

If valid is true then

Call the checkUserDatabase function

Return the loginPasswordd variable

End Sub

Sub Procedure checkUserDatabase

Establish connection to database

Set query to SELECT \* FROM users WHERE username = the validated loginUsername variable AND password = the validated loginPassword variable

Execute query

If a row is retrieved from the user database

The login form is hidden and disabled

A message box is displayed containing the text (“Welcome back to Garbage Grabberoos” & the validated loginUsername variable)

The game menu is shown and enabled

Set found to true

End While

Else

Display and enable the no account found error message for 3 seconds using a timer

Set found to false

Exit the sub procedure

End If

Close the database connection

End Sub

When the timer ticks

Disable Timer1

Hide and disable the unsuccessful sign-up or login error message

Show and enable the login form

Hide and disable the sign-up form

End Sub

When the linklabel is clicked

Hide and disable the login form

Show and enable the sign up form

End Sub

When the login form is loaded

Send Panel1 to back

Send Panel2 to back

Centre the login form on screen

End Sub

When the exit button is clicked

End the program

End Sub

End Class

**Class GameGrid**

Declare a 2D array “grid” as integer

Declare rubbishCount As Integer = 0

Declare noOfguesses As Integer = 70

Delcare SW As a New Stopwatch

Declare Withevents Tmr As a New Windows Forms Timer

Declare time As String = “”

Declare paused As Boolean

When the GameGrid form loads

Centre the game grid form on screen

Hide the Date Time Picker on the game grid form

Set the timer interval to 100

Set the Data Grid View row count to 11

Set the Data Grid View column count to 10

For counter = 0 to the Data Grid View column count – 1

Set the Data Grid View column width to 55

For counter = 0 to the Data Grid View row count – 1

Set the Data Grid View row width to 30

Next

Disallow user to add rows to the Data Grid view

Call the placeSweetWrapper sub procedure once

Call the placePlasticStraw sub procedure twice

Call the placeCrispPacket sub procedure three times

Call the placePlasticBottle sub procedure four times

Call the placeLargeCar sub procedure five times

Sub Procedure placeSweetWrapper

Declare row as a random integer between 0 and 9

Declare col as a random integer between 0 and 9

Declare direction as a random integer between 0 and 1

If direction = 0 then

For counter = 0 to length – 1

If row + counter > 9 then

Exit for

End if

Set the grid square with the coordinate (row + counter, col) to 1

Next

ElseIf direction = 1

For counter = 0 to length – 1

If col + counter > 9 then

Exit for

End If

Set the grid square with the coordinate (row, col + counter) to 1

Next

End If

End Sub

Sub Procedure placePlasticStraw

Declare row as a random integer between 0 and 9

Declare col as a random integer between 0 and 9

Declare direction as a random integer between 0 and 1

If direction = 0 then

For counter = 0 to length – 1

If row + counter > 9 then

Exit for

End if

Set the grid square with the coordinate (row + counter, col) to 2

Next

ElseIf direction = 1

For counter = 0 to length – 1

If col + counter > 9 then

Exit for

End If

Set the grid square with the coordinate (row, col + counter) to 2

Next

End If

End Sub

Sub Procedure placeCrispPacket

Declare row as a random integer between 0 and 9

Declare col as a random integer between 0 and 9

Declare direction as a random integer between 0 and 1

If direction = 0 then

For counter = 0 to length – 1

If row + counter > 9 then

Exit for

End if

Set the grid square with the coordinate (row + counter, col) to 3

Next

ElseIf direction = 1

For counter = 0 to length – 1

If col + counter > 9 then

Exit for

End If

Set the grid square with the coordinate (row, col + counter) to 3

Next

End If

End Sub

Sub Procedure placePlasticBottle

Declare row as a random integer between 0 and 9

Declare col as a random integer between 0 and 9

Declare direction as a random integer between 0 and 1

If direction = 0 then

For counter = 0 to length – 1

If row + counter > 9 then

Exit for

End if

Set the grid square with the coordinate (row + counter, col) to 4

Next

ElseIf direction = 1

For counter = 0 to length – 1

If col + counter > 9 then

Exit for

End If

Set the grid square with the coordinate (row, col + counter) to 4

Next

End If

End Sub

Sub Procedure placeLargeCar

Declare row as a random integer between 2 and 5

Declare col as a random integer between 0 and 9

Declare direction as a random integer between 0 and 1

If direction = 0 then

For counter = 0 to length – 1

If row + counter > 9 then

Exit for

End if

Set the grid square with the coordinate (row + counter, col) to 5

Next

ElseIf direction = 1

For counter = 0 to length – 1

If col + counter > 9 then

Exit for

End If

Set the grid square with the coordinate (row, col + counter) to 5

Next

End If

End Sub

When the timer ticks

Set the value of a form label to the value stored in the Stopwatch

Set the value of a form label to the value stored in the rubbish count variable

End Sub

When a cell is clicked on the DataGridView

Start the Stopwatch

Set paused to false

Set the value of a form label to the value stored in the noOfguesses variable

If the grid square at the RowIndex and ColumnIndex has the value of 6 then

Send “You already hit that cell” to display

End If

If the grid square at the RowIndex and ColumnIndex has the value of 1 then

Set the grid square colour to red

Set the grid square value to 6

Set the value of the variable rubbishCount to current value of rubbishCount + 1

Set the value of the variable noOfguesses to current value of noOfguesses - 1

Show and enable the sweetWrapper form

ElseIf the grid square at the RowIndex and ColumnIndex has the value of 2 then

Set the grid square colour to blue

Set the grid square value to 6

Set the value of the variable rubbishCount to the current value of rubbishCount + 1

Set the value of the variable noOfguesses to the current value of noOfguesses - 1

Show and enable the plasticStraw form

ElseIf the grid square at the RowIndex and ColumnIndex has the value of 3 then

Set the grid square colour to green

Set the grid square value to 6

Set the value of the variable rubbishCount to the current value of rubbishCount + 1

Set the value of the variable noOfguesses to the current value of noOfguesses - 1

Show and enable the crispPacket form

ElseIf the grid square at the RowIndex and ColumnIndex has the value of 4 then

Set the grid square colour to pink

Set the grid square value to 6

Set the value of the variable rubbishCount to the current value of rubbishCount + 1

Set the value of the variable noOfguesses to the current value of noOfguesses - 1

Show and enable the plasticBottle form

ElseIf the grid square at the RowIndex and ColumnIndex has the value of 5 then

Set the grid square colour to plum

Set the grid square value to 6

Set the value of the variable rubbishCount to the current value of rubbishCount + 1

Set the value of the variable noOfguesses to the current value of noOfguesses - 1

Show and enable the largeCar form

Else set the grid square colour at the RowIndex and ColumnIndex to white

Set the grid square value to 6

Set the value of the variable noOfguesses to the current value of noOfguesses - 1

End If

If value of the variable rubbishCount is 15 then

Show the winningMessage form

Call the sub procedure newWinningTime

Stop the stopwatch

Reset the stopwatch

Hide the game grid form

Reset the game grid form to its original state

Revert the value of the number of guesses variable to 70

Initialise the rubbish count

End If

If the value of the variable noOfguesses is 0 and the value of the rubbish count variable is not 15 then

Show the losingMessage form

Hide the game grid form

Stop the stopwatch

Reset the stopwatch

Reset the game grid form to its original state

Revert the value of the number of guesses variable to 70

Initialise the rubbish count

End If

End Sub

Sub Procedure newWinningTime

Set the value of the variable time to the value held in the stopwatch

Declare file As System.IO.Streamwriter

Specify the filepath for the file to be written to

Write the stopwatch time and the date to file

Close the file

End Sub

When the help button is clicked

Show the Instructions form

End Sub

When the menu button is clicked

Show and enable the gameMenu form

Hide and disable the gameGrid form

End Sub

When the paused button is clicked

Set the variable paused to true

Stop the stopwatch

Disable the gameGrid form

Show the pausedForm

End Sub

When the end button is clicked

End the program

End Sub

When the restart button is clicked

Reset the game grid form to its original state

Reset the stopwatch

Revert the value of the number of guesses variable to 70

Initialise the rubbish count variable

End Sub

End Class

**Class gameMenu**

When the start game button is clicked

Hide and disable the gameMenu form

Show and enable the gameGrid form

End Sub

When the quit game button is clicked

End the program

When the view leaderboard button is clicked

Hide and disable the game menu form

Show and enable the speedy searchers board form

End Sub

When the game menu is loaded

Center the game menu form on screen

End Sub

End Class

**Class accountExisting**

When the account existing form loads

Centre the account existing form on screen

End Sub

When the exit button is clicked

Close the accountExisting form

Show and enable the login form

End Sub

When the linklabel is clicked

Show and enable the login form

Hide and disable the account existing form

End Sub

End Class

**Class successfulSignUp**

When the exit button is clicked

Close the successful sign up form

Show and enable the login form

End Sub

When the successful sign up form is loaded

Centre the successful sign up form on screen

End Sub

End Class

**Class unsuccessfulSignUpOrLogin**

When the exit button is clicked

Close the unsuccessful sign up or login form

End Sub

When the unsuccessful sign up or login form is loaded

Centre the unsuccessful sign up or login form on screen

End Sub

When the login button is clicked

Close the unsuccessful sign up or login form

Show and enable the login form

End Sub

When the sign up button is clicked

Close the unsuccessful sign up or login form

Show and enable the sign up form

End Sub

End Class

Class noAccountFound

When the linklabel is clicked

Close the no account found error message form

Show and enable the sign up form

When the exit button is clicked

Close the no account found error message form

Show and enable the login form

When the no account found form is loaded

Centre the no account found error message form on user’s screen

Class howToPlay

When the how to play form is loaded

Centre the how to play form on the user’s screen

End Sub

When the exit button is clicked

Close the how to play form

Show and enable the game grid form

End Sub

End Class

**Class sweetWrapper**

When the exit button is clicked

Close the sweet wrapper form infographic

End Sub

When the sweet wrapper form infographic is loaded

Centre the sweet wrapper form infographic on screen

End Sub

End Class

**Class plasticStraw**

When the exit button is clicked

Close the plastic straw form infographic

End Sub

When the plastic straw form infographic is loaded

Centre the plastic straw form infographic on screen

End Sub

End Class

**Class crispPacket**

When the exit button is clicked

Close the crisp packet form infographic

End Sub

When the crisp packet form infographic is loaded

Centre the crisp packet form infographic on screen

End Sub

End Class

**Class plasticBottle**

When the exit button is clicked

Close the plastic bottle form infographic

End Sub

When the plastic bottle form infographic is loaded

Centre the plastic bottle form infographic on screen

End Sub

End Class

**Class largeCar**

When the exit button is clicked

Close the large car form infographic

End Sub

When the large car form infographic is loaded

Centre the large car form infographic on screen

End Sub

End Class

**Class pausedForm**

When the continue game button is clicked

Enable the game grid

Close the paused message form

End Sub

When the paused form is loaded

Centre the paused message form on screen

End Sub

End Class

**Class Instructions**

When the exit button is clicked

Hide and disable the instructions form

When the instructions form is loaded

Send Panel1 to back

Centre the instructions form on the user’s screen

**Class winningMessage**

When the restart button is clicked

Show and enable the game grid form

Close the winning message form

End Sub

When the view leaderboard button is clicked

Show the speedy searchers board

Close the winning message form

End Sub

When the winning message form is loaded

Centre the winning message form on screen

End Sub

End Class

**Class losingMessage**

When the restart button is clicked

Show and enable the game grid form

Close the losing message form

End Sub

When the view leaderboard button is clicked

Show the speedy searchers board

Close the losing message form

End Sub

When the losing message form is loaded

Centre the losing message form on screen

End Sub

End Class

## Design Structure of 2D Array & Arrays of Records

**2D Array: ‘grid’**

2D array of Integer

The entire grid will contain 11 rows and 11 columns, however count starts from 0 in VB so the grid will be declared as grid(10,10)

Only the coordinates in the grid that fall between both rows 1 to 10 and columns 1 to 10 will be assigned the value of a number between 1 and 10 for identification of either a piece of rubbish or an empty grid. This is because the first column will not contain coordinates since it used as an alphabet and the first row will not contain coordinates since it is used as a number count.

Visual Layout (using DataGridView):



**Record Structure: searchDetails (forms 2 Arrays of records: ‘time’ & ‘date’)**

|  |  |  |
| --- | --- | --- |
| **Field Name** | Time | Date |
| **Field Type** | String | String |

The number of values present in both the Time and Date array of records is incremented each time a user successfully completes the game. The Date array of records will contain the date provided by the DateTimePicker on the GameGrid form (i.e., the date of the user’s win), and the Time array of records will contain the time provided by the stopwatch on the GameGrid form (i.e., the time taken for the user to win)

# IMPLEMENTATION

AH Computing Coursework

## Program Code

### Sign Up Class

'Garbage Grabberoos Sign up

Imports System.Data.OleDb

Public Class SignUp

Private Sub btnSignup\_Click(sender As Object, e As EventArgs) Handles btnSignup.Click

'declares variables to contain the values of the user's 8 letter username and six letter password

Dim usernamee As String = ""

Dim passwordd As String = ""

'reads the user input for username and password and assigns them to the corresponding variables

usernamee = checkUsername(usernamee)

passwordd = validPassword(usernamee, passwordd)

End Sub

Public Function checkUsername(ByRef usernamee As String)

'reads the user input for username and assigns it to a variable which will be checked for validity

usernamee = txtUsername.Text

'declares variable to hold the validity of the username

Dim valid As Boolean = False

'initially the error message is not enabled

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'conditional loop - prompts user to re-enter an input for username until it is valid

Do

'fixed loop to check that the username length is correct

If Len(usernamee) = 8 Then

'the boolean variable valid is assigned the value of true which allows the conditional loop to end

valid = True

Else

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

'display error message to notify user that they have entered an unsuitable username/password and haven't created a Garbage Grabberoos account

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'disable and hide the existing account error message

accountExisting.Visible = False

accountExisting.Enabled = False

'timer used to only display the message for 3 seconds before re-opening the sign up form

Timer1.Enabled = True

Exit Function

'end fixed loop

End If

'input validation for an empty textbox

If Len(usernamee) = 0 Or txtUsername.Text = "" Then

'show and enable the unsuccessful sign up message

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

End If

'the conditional loop will continue until an appropriate username is created

Loop Until valid = True

'the validated username is returned for use in the main program

Return usernamee

End Function

Public Function validPassword(ByRef usernamee As String, ByRef passwordd As String)

'reads the user input for password and assigns it to a variable which will be checked for validity

passwordd = txtPassword.Text

'declares variable to hold the validity of the password

Dim valid As Boolean = False

'initially both the error message and success message are invisible and not enabled

successfulSignUp.Visible = False

successfulSignUp.Enabled = False

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'conditional loop - prompts user to re-enter an input for password until it is valid

Do

'fixed loop to check that the username length is correct

If Len(passwordd) = 6 Then

'the boolean variable valid is assigned the value of true which allows the conditional loop to end

valid = True

Else

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

'display error message to notify user that they have entered an unsuitable username/password and haven't created a Garbage Grabberoos account

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'timer used to only display the error message for 3 seconds before re-opening the sign up form

Timer2.Enabled = True

Exit Function

'exit fixed loop

End If

'input validation for an empty textbox

If Len(passwordd) = 0 Or txtUsername.Text = "" Then

'show and enable the unsuccessful sign up message

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

End If

'the conditional loop will continue until an appropriate password is created

Loop Until valid = True

'if both the username and password have been succesfully validated

If valid = True Then

'call a function to add the new user to the Garbage Grabberoos Users database (as long as they do not have an already existing account associated with the entered username and password)

Call newUser(usernamee, passwordd)

End If

'the validated password is returned for use in the main program

Return passwordd

End Function

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

'disables timer

Timer1.Enabled = False

'hides and disables error message

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'allows sign up form to be interacted with by user

Me.Visible = True

Me.Enabled = True

End Sub

'adds a new user to the Garbage Grabberoos user database

Public Sub newUser(ByRef usernamee As String, ByRef passwordd As String)

'input validation to prevent the user from signing up with- details linked to an account that already exists

Try

Dim sqlreader As OleDbDataReader

'specifies the file path - database resides in the bin/Debug folder for the project

Dim connectionType As String = "Provider=Microsoft.Jet.OLEDB.4.0;"

Dim fileLocation As String = "Data Source=GarbageGrabberoosUsers.mdb"

'opens the database connection

Dim conn As OleDbConnection

conn = New OleDbConnection(connectionType & fileLocation)

conn.Open()

'sets up the query (selecting for accounts associated with the username and password entered by the user)

Dim query As String = "SELECT \* FROM users WHERE username = '" & usernamee & "' AND password = '" & passwordd & "'"

'runs the query

Dim command As New OleDbCommand(query, conn)

sqlreader = command.ExecuteReader

'if an account linked to the username and password exists

If sqlreader.Read() Then

'a notification is displayed informing the user that they already have an account and prompting them to login

accountExisting.Visible = True

accountExisting.Enabled = True

'the procedure is ended to prevent the loop from continuing

Exit Sub

'the successful sign up notification is blocked from being shown

successfulSignUp.Visible = False

successfulSignUp.Enabled = False

'closes the database connection

conn.Close()

'if there is no account linked to the entered username and password

Else

'sets up the query (inserting the user's entered username and password to the Garbage Grabberoos users database)

Dim query1 As String = "INSERT INTO users([Username],[Password]) VALUES (""" & usernamee & """,""" & passwordd & """)"

'runs the query

Dim command1 As New OleDbCommand(query1, conn)

sqlreader = command1.ExecuteReader

'closes the database connection

conn.Close()

'display successful sign up message to notify user that they have created a Garbage Grabberoos account and prompt them to login

successfulSignUp.Visible = True

successfulSignUp.Enabled = True

End If

'handles unexpected or runtime exceptions

Catch ex As Exception

MsgBox(ex.Message)

End Try

End Sub

'establishes a link between login and sign up for users who already have an account

Private Sub LinkLabel1\_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles LinkLabel1.LinkClicked

'shows and enables the login form

Login.Visible = True

Login.Enabled = True

'hides and disables the sign up form

Me.Visible = False

Me.Enabled = False

'hides and disables the error message

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

End Sub

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'the program will end if the exit button is clicked

End

End Sub

Private Sub SignUp\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centers the sign up form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Unsuccessful SignUp or Login Class

'Garbage Grabberoos Unsusccesful SignUp And Login Error Message

Public Class unsuccessfulSignUpOrLogin

'when the exit button is clicked the error message will close

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

Me.Close()

End Sub

Private Sub unsuccessfulSignUpOrLogin\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the unsuccessful sign up or login form on the user's screen

Me.CenterToScreen()

End Sub

Private Sub btnLogin\_Click(sender As Object, e As EventArgs) Handles btnLogin.Click

'when the login button is clicked the error message will close

Me.Close()

'the login form will be made visible and able to be filled in

Login.Visible = True

Login.Enabled = True

End Sub

Private Sub btnSignUp\_Click(sender As Object, e As EventArgs) Handles btnSignUp.Click

'when the sign up button is clicked the error message will close

Me.Close()

'the sign up form will be made visible and able to be filled in

SignUp.Visible = True

SignUp.Enabled = True

End Sub

End Class

### Successful SignUp Class

'Garbage Grabberoos Successful Sign Up Message

Public Class successfulSignUp

'once the user has signed up successfully and is shown a message that tells them this, they will press an exit button

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'when the exit button is clicked the success message will close

Me.Close()

'the exit button will make the login screen visible and enabled so that they can log in with their newly formed account

Login.Visible = True

Login.Enabled = True

End Sub

Private Sub successfulSignUp\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the successful sign up form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Account Existing Class

'Garbage Grabberoos existing account notification

Public Class accountExisting

Private Sub Panel1\_Paint(sender As Object, e As PaintEventArgs) Handles Panel1.Paint

'sends the panel to the back - used to create a border arouund form text which will provide a cohesive user interface

Panel1.SendToBack()

End Sub

Private Sub accountExisting\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centres the account existing form on screen

Me.CenterToScreen()

End Sub

Private Sub btnExit\_Click\_1(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the account existing form will be closed

Me.Close()

'show the login form to allow the user to login with the existing account

Login.Visible = True

Login.Enabled = True

End Sub

'the link label is used to provide a link to the login form

Private Sub LinkLabel1\_LinkClicked\_1(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles LinkLabel1.LinkClicked

'shows the login form

Login.Visible = True

Login.Enabled = True

'closes and hides the account existing form

Me.Close()

End Sub

End Class

### Login Class

'Garbage Grabberoos Login

Imports System.Data.OleDb

Public Class Login

Public Sub btnLogin\_Click(sender As Object, e As EventArgs) Handles btnLogin.Click

'declares variables to contain the values of the user's 8 letter username and six letter password

Dim loginUsername As String = ""

Dim loginPassword As String = ""

Dim loginUsernamee As String = ""

Dim loginPasswordd As String = ""

'calls a function to validate the username

loginUsername = checkUsername(loginUsernamee)

'calls a function to validate the password

loginPassword = validPassword(loginUsernamee, loginPasswordd)

End Sub

Public Function checkUsername(ByRef loginUsernamee As String)

'reads the user input for username and assigns it to a variable which will be checked for validity

loginUsernamee = txtUsername.Text

'declares variable to hold the validity of the username

Dim valid As Boolean = False

'initially the error message is not enabled

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'conditional loop - promps user to re-enter an input for username until it is valid

Do

'fixed loop to check that the username length is correct

If Len(loginUsernamee) = 8 Then

'the boolean variable valid is assigned the value of true which allows the conditional loop to end

valid = True

Else

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

'display error message to notify user that they have entered a username/password of unsuitable length that cannot correspond to an existing account

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'timer used to only display the message for 3 seconds before re-opening the login form

Timer1.Enabled = True

Exit Function

'end fixed loop

End If

'input validation for an empty textbox

If Len(loginUsernamee) = 0 Or txtUsername.Text = "" Then

'show and enable the unsuccessful sign up message

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

End If

'the conditional loop will continue until an appropriate username is entered for login

Loop Until valid = True

'the validated username is returned for use in the main program

Return loginUsernamee

End Function

Public Function validPassword(ByRef loginUsernamee As String, ByRef loginPasswordd As String)

'reads the user input for password and assigns it to a variable which will be checked for validity

loginPasswordd = txtPassword.Text

'declares variable to hold the validity of the password

Dim valid As Boolean = False

'initially the error message is invisible and not enabled

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'conditional loop - promps user to re-enter an input for password until it is valid

Do

'fixed loop to check that the username length is correct

If Len(loginPasswordd) = 6 Then

'the boolean variable valid is assigned the value of true which allows the conditional loop to end

valid = True

Else

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

'display error message to notify user that they have entered a username/password of unsuitable length that cannot correspond to an existing account

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'timer used to only display the error message for 3 seconds before re-opening the login form

Timer1.Enabled = True

Exit Function

'exit fixed loop

End If

'input validation for an empty textbox

If Len(loginPasswordd) = 0 Or txtPassword.Text = "" Then

'show and enable the unsuccessful sign up message

unsuccessfulSignUpOrLogin.Visible = True

unsuccessfulSignUpOrLogin.Enabled = True

'the boolean variable valid is assigned the value of false which will not allow the conditional loop to end

valid = False

End If

'the conditional loop will continue until an appropriate password is entered for login

Loop Until valid = True

'if valid is true then both the username and password have been successfully validated

If valid = True Then

'calls the check user database procedure to search the database for an account that corresponds to the validated username and password

Call checkUserDatabase(loginUsernamee, loginPasswordd)

End If

'the validated password is returned for use in the main program

Return loginPasswordd

End Function

Public Sub checkUserDatabase(ByRef loginUsernamee As String, ByRef loginPasswordd As String)

'declares a boolean variable for the validity of the password

Dim found As Boolean = False

Dim sqlreader As OleDbDataReader

'specifies the file path - database resides in the bin/Debug folder for the project

Dim connectionType As String = "Provider=Microsoft.Jet.OLEDB.4.0;"

Dim fileLocation As String = "Data Source=GarbageGrabberoosUsers.mdb"

'opens the database connection

Dim conn As OleDbConnection

conn = New OleDbConnection(connectionType & fileLocation)

conn.Open()

'sets up the query (selects all data that corresponds to the entered username and password)

Dim query As String = "SELECT \* FROM USERS WHERE username = '" & loginUsernamee & "' AND password = '" & loginPasswordd & "'"

'runs the query

Dim command As New OleDbCommand(query, conn)

sqlreader = command.ExecuteReader

'loops through the rows in the result set to determine whether the user has an existing account

If sqlreader.HasRows Then

While sqlreader.Read

'if an account is found (i.e rows count is more than one)

'hides and disables the login form

Me.Visible = False

Me.Enabled = False

'displays a message box informing the user of a successful login

MsgBox("Welcome back to Garbage Grabberoos " & loginUsernamee)

'shows and enables the game menu form

gameMenu.Visible = True

gameMenu.Enabled = True

'changes the value of the boolean variable 'found' to true instead of false, demonstrating that the user's entered username and password correspond to an existing account

found = True

End While

Else

'displays error message to notify user that they have entered a username/password that does not correspond to an existing account

noAccountFound.Visible = True

noAccountFound.Enabled = True

'timer used to only display the error message for 3 seconds before re-opening the sign up form

Timer1.Enabled = True

'the boolean variable 'found' will remain as false, demonstrating that the user's entered username and password does not correspond to an existing account

found = False

Exit Sub

End If

'closes the database connection

conn.Close()

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

'disables timer

Timer1.Enabled = False

'hides and disables error message

unsuccessfulSignUpOrLogin.Visible = False

unsuccessfulSignUpOrLogin.Enabled = False

'allows login form to be seen and interacted with by user

Me.Visible = True

Me.Enabled = True

'hides and disables sign up form

SignUp.Visible = False

SignUp.Enabled = False

End Sub

'establishes a link between login and sign up for users who do not already have an account

Private Sub LinkLabel1\_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles LinkLabel1.LinkClicked

'hides and disables the login form

Me.Visible = False

Me.Enabled = False

'opens and enables the sign up form

SignUp.Visible = True

SignUp.Enabled = True

End Sub

Private Sub Login\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to create a border around both parts of the login form (for a cohesive user interface)

Panel1.SendToBack()

Panel2.SendToBack()

'used to centre the login form on the user's screen

Me.CenterToScreen()

End Sub

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'the program will end if the exit button is clicked

End

End Sub

### No Account Found Class

'Garbage Grabberoos 'no account found' notification

Public Class noAccountFound

'this linklabel establishes a link between the error message and sign up form if the user does not already have an account

Private Sub LinkLabel1\_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles LinkLabel1.LinkClicked

'the noAccountFound form will be closed

Me.Close()

'the sign up form will be enabled and made visisble

SignUp.Visible = True

SignUp.Enabled = True

End Sub

'if the user clicks on the exit button instead of the sign up link label on the form, it can be assumed that the failed login was due to a syntax error and not because they do not already have an account

'the login form will re-open as a result

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'the noAccountFound form will be closed

Me.Close()

'the login form will be enabled and made visisble

Login.Visible = True

Login.Enabled = True

End Sub

Private Sub Panel1\_Paint(sender As Object, e As PaintEventArgs) Handles Panel1.Paint

'sends the panel to the back - used to create a border arouund form text which will provide a cohesive user interface

Panel1.SendToBack()

End Sub

Private Sub noAccountFound\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centres the noAccountFound form on screen

Me.CenterToScreen()

End Sub

End Class

### Game Menu Class

'Garbage Grabberoos Game Menu

Public Class gameMenu

'when the start button is clicked on game menu

Private Sub btnGameStart\_Click(sender As Object, e As EventArgs) Handles btnGameStart.Click

'the game menu is hidden and disabled

Me.Visible = False

Me.Enabled = False

'the how to play form is opened and made visible

howToPlay.Visible = True

howToPlay.Enabled = True

End Sub

'when the quit button is clicked on game menu

Private Sub btnQuitGame\_Click(sender As Object, e As EventArgs) Handles btnQuitGame.Click

'the game is closed and ended

End

End Sub

'when the speedy searchers button is clicked on the game menu

Private Sub btnSpeedySearchers\_Click(sender As Object, e As EventArgs) Handles btnSpeedySearchers.Click

'the game menu is hidden and disabled

Me.Visible = False

Me.Enabled = False

'the speedy searchers board is opened and made visible

speedySearchersBoard.Visible = True

speedySearchersBoard.Enabled = True

End Sub

Private Sub gameMenu\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centres the game menu on the user's screen

Me.CenterToScreen()

End Sub

End Class

### How To Play Class

'Garbage Grabberoos Opening Instructions Form

Public Class howToPlay

Private Sub howToPlay\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centres the how to play form on the user's screen

Me.CenterToScreen()

End Sub

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'closes the how to play form

Me.Close()

'shows the game grid on user's screen

GameGrid.Visible = True

GameGrid.Enabled = True

End Sub

Private Sub btnStart\_Click(sender As Object, e As EventArgs) Handles btnStart.Click

'shows and enables the game grid

GameGrid.Visible = True

GameGrid.Enabled = True

'hides and disables the how to play form

Me.Visible = False

Me.Enabled = False

End Sub

Private Sub Panel2\_Paint\_1(sender As Object, e As PaintEventArgs) Handles Panel2.Paint

'used to create a border around title - maintains a cohesive user interface

Panel2.SendToBack()

End Sub

End Class

### Game Grid Class

'Garbage Grabberoos Game Grid

'predefined modules that need to be added

Imports System.Data.OleDb

'is needed for the StreamWriter.

Imports System.IO

Public Class GameGrid

'global variables for Garbage Grabberoos

'declares a 2D array to store the grid

Private grid(10, 10) As Integer

'declares a variable to count how many pieces of rubbish have been found by the user

Private rubbishCount As Integer = 0

'declares a variable to guess how many squares have been guessed by the user in attempts to find rubbish

Private noOfguesses As Integer = 70

'declares variables to create a stopwatch to time the user's game

Private SW As New Stopwatch

Private WithEvents Tmr As New System.Windows.Forms.Timer

'declares a variable to store the value held by the stopwatch as the length of the time taken by the user to win the game

Private time As String = ""

'declares a variable to pause the game

Private paused As Boolean

Private Sub GameGrid\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centres the game grid on the user's screen

Me.CenterToScreen()

'hides the date time picker on the form as it is only needed to provide the date for the Speedy Searchers leaderboard

DateTimePicker1.Visible = False

'sets the interval for the stopwatch timer

Tmr.Interval = 100

'initialises the DataGridView control

'allows the grid to have 11 rows

DataGridView1.RowCount = 11

'allows the grid to have 10 rows

DataGridView1.ColumnCount = 10

'loops from the beginning to the end of the grid (by column)

For i As Integer = 0 To DataGridView1.Columns.Count - 1

'sets the column width of the grid to fully fit the space assigned to the Datagridview

DataGridView1.Columns(i).Width = 55

Next

'loops from the beginning to the end of the grid (by row)

For i As Integer = 0 To DataGridView1.Rows.Count - 1

'sets the row width of the grid to fully fit the space assigned to the Datagridview

DataGridView1.Rows(i).Height = 30

Next

'prevents the user from adding rows and altering the grid

DataGridView1.AllowUserToAddRows = False

'places pieces of rubbish of varying size and type on the game board

placeSweetWrapper(1)

placePlasticStraw(2)

placeCrispPacket(3)

placePlasticBottle(4)

placeLargeCar(5)

End Sub

Private Sub placeSweetWrapper(length As Integer)

'changes the positions of the sweet wrapper each time

'determines the starting cell for the sweet wrapper

Dim row As Integer = CInt(Int(Rnd() \* 9))

Dim col As Integer = CInt(Int(Rnd() \* 9))

'determines the direction for the sweet wrapper by choosing randomly between 0 and 1 (horizontal or vertical)

Dim direction As Integer = CInt(Int(Rnd() \* 1))

'places the sweet wrapper on the game board

If direction = 0 Then

For i As Integer = 0 To length - 1

If row + i > 9 Then

Exit For

End If

'places the sweet wrapper in a vertical line onto the grid and assigns the squares it reside on the value of 1 (to show that it is present here and allow it to be found)

grid(row + i, col) = 1

Next

ElseIf direction = 1 Then

For i As Integer = 0 To length - 1

If col + i > 9 Then

Exit For

End If

'places the sweet wrapper in a horizontal line onto the grid and assigns the squares it reside on the value of 1 (to show that it is present here and allow it to be found)

grid(row, col + i) = 1

Next

End If

End Sub

Private Sub placePlasticStraw(length As Integer)

'changes the positions of the plastic straw each time

'determines the starting cell for the plastic straw

Dim row As Integer = CInt(Int(Rnd() \* 9))

Dim col As Integer = CInt(Int(Rnd() \* 9))

'determines the direction for the plastic straw by choosing randomly between 0 and 1 (horizontal or vertical)

Dim direction As Integer = CInt(Int(Rnd() \* 1))

'places the plastic straw on the game board

If direction = 0 Then

For i As Integer = 0 To length - 1

If row + i > 9 Then

Exit For

End If

'places the plastic straw in a vertical line onto the grid and assigns the squares it reside on the value of 2 (to show that it is present here and allow it to be found)

grid(row + i, col) = 2

Next

Else

For i As Integer = 0 To length - 1

If col + i > 9 Then

Exit For

End If

'places the plastic straw in a horizontal line onto the grid and assigns the squares it reside on the value of 2 (to show that it is present here and allow it to be found)

grid(row, col + i) = 2

Next

End If

End Sub

Private Sub placeCrispPacket(length As Integer)

'changes the positions of the crisp packet each time

'determines the starting cell for the crisp packet

Dim row As Integer = CInt(Int(Rnd() \* 9))

Dim col As Integer = CInt(Int(Rnd() \* 9))

'determines the direction for the plastic straw by choosing randomly between 0 and 1 (horizontal or vertical)

Dim direction As Integer = CInt(Int(Rnd() \* 1))

'places the crisp packet on the game board

If direction = 0 Then

For i As Integer = 0 To length - 1

If row + i > 9 Then

Exit For

End If

'places the crisp packet in a vertical line onto the grid and assigns the squares it reside on the value of 3 (to show that it is present here and allow it to be found)

grid(row + i, col) = 3

Next

Else

For i As Integer = 0 To length - 1

If col + i > 9 Then

Exit For

End If

'places the crisp packet in a horizontal line onto the grid and assigns the squares it reside on the value of 3 (to show that it is present here and allow it to be found)

grid(row, col + i) = 3

Next

End If

End Sub

Private Sub placePlasticBottle(length As Integer)

'changes the positions of the plastic bottle each time

'determines the starting cell for the plastic bottle

Dim row As Integer = CInt(Int(Rnd() \* 9))

Dim col As Integer = CInt(Int(Rnd() \* 9))

'determines the direction for the plastic straw by choosing randomly between 0 and 1 (horizontal or vertical)

Dim direction As Integer = CInt(Int(Rnd() \* 1))

'places the bottle on the game board

If direction = 0 Then

For i As Integer = 0 To length - 1

If row + i > 9 Then

Exit For

End If

'places the plastic bottle in a vertical line onto the grid and assigns the squares it reside on the value of 4 (to show that it is present here and allow it to be found)

grid(row + i, col) = 4

Next

Else

For i As Integer = 0 To length - 1

If col + i > 9 Then

Exit For

End If

'places the plastic bottle in a horizontal line onto the grid and assigns the squares it reside on the value of 4 (to show that it is present here and allow it to be found)

grid(row, col + i) = 4

Next

End If

End Sub

Private Sub placeLargeCar(length As Integer)

'changes the positions of the large car each time

'determines the starting cell for the large car

'code for row of the large car is different from the rest of the rubbish due to it being the largest (needs to be adjusted in order to fit on the grid)

Dim row As Integer = CInt(Int(Rnd() \* 3) + 2)

Dim col As Integer = CInt(Int(Rnd() \* 9))

'determines the direction for the plastic straw by choosing randomly between 0 and 1 (horizontal or vertical)

Dim direction As Integer = CInt(Int(Rnd() \* 1))

'places the large car on the game board

If direction = 0 Then

For i As Integer = 0 To length - 1

If row + i > 9 Then

Exit For

End If

'places the large car in a vertical line onto the grid and assigns the squares it reside on the value of 5 (to show that it is present here and allow it to be found)

grid(row + i, col) = 5

Next

Else

For i As Integer = 0 To length - 1

If col + i > 9 Then

Exit For

End If

'places the large car in a horizontal line onto the grid and assigns the squares it reside on the value of 5 (to show that it is present here and allow it to be found)

grid(row, col + i) = 5

Next

End If

End Sub

Private Sub Tmr\_Tick(sender As Object, e As System.EventArgs) Handles Tmr.Tick

'assigns label 14 the value contained in the stopwatch to allow the user's time to be seen on the form

Label4.Text = SW.Elapsed.ToString

'assigns label 25 the value contained in the variable 'rubbishCount' which counts how many pieces of rubbish have been successfully found by the user (allows this number to be seen on the form)

Label25.Text = rubbishCount

End Sub

'when a cell on the Datagridview is clicked

Private Sub DataGridView1\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles DataGridView1.CellClick

'the stopwatch timer is started

SW.Start()

Tmr.Start()

'initially the pause button has not been pressed and the game is running

paused = False

'assigns label the value contained in the variable 'noOfguesses' which counts how many cells have been clicked by the user in order to guess the location of pieces of rubbish

Label1.Text = noOfguesses

'checks if the cell has already been hit

If grid(e.RowIndex, e.ColumnIndex) = 6 Then

'notifies the user with a message box if the cells has already been hit

MessageBox.Show("You already hit that cell.")

End If

'checks if the cell contains a type of rubbish piece (sweet wrapper, plastic straw, crisp packet, plastic bottle or large car)

'if the cell contains a sweet wrapper

If grid(e.RowIndex, e.ColumnIndex) = 1 Then

'the cell will turn red when clicked

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.Red

'the cell will be assigned the value of 6 which labels it as 'hit' - which will stop the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'rubbish count will be incremented

rubbishCount = rubbishCount + 1

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

'the sweet wrapper infographic form will be shown to the user

sweetWrapper.Visible = True

sweetWrapper.Enabled = True

'if the cell contains a plastic straw

ElseIf grid(e.RowIndex, e.ColumnIndex) = 2 Then

'the cell will turn blue when clicked

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.Blue

'the cell will be assigned the value of 6 which labels it as 'hit' - which will stop the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'rubbish count will be incremented

rubbishCount = rubbishCount + 1

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

'the plastic straw infographic form will be shown to the user

plasticStraw.Visible = True

plasticStraw.Enabled = True

'if the cell contains a crisp packet

ElseIf grid(e.RowIndex, e.ColumnIndex) = 3 Then

'the cell will turn green when clicked

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.Green

'the cell will be assigned the value of 6 which labels it as 'hit' - stopping the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'rubbish count will be incremented

rubbishCount = rubbishCount + 1

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

'the plastic straw infographic form will be shown to the user

crispPacket.Visible = True

crispPacket.Enabled = True

'if the cell contains a plastic bottle

ElseIf grid(e.RowIndex, e.ColumnIndex) = 4 Then

'the cell will turn pink when clicked

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.Pink

'the cell will be assigned the value of 6 which labels it as 'hit' - stopping the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'rubbish count will be incremented

rubbishCount = rubbishCount + 1

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

'the plastic bottle infographic form will be shown to the user

plasticBottle.Visible = True

plasticBottle.Enabled = True

'if the cell contains a large car

ElseIf grid(e.RowIndex, e.ColumnIndex) = 5 Then

'the cell will turn plum when clicked

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.Plum

'the cell will be assigned the value of 6 which labels it as 'hit' - stopping the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'rubbish count will be incremented

rubbishCount = rubbishCount + 1

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

'the large car infographic form will be shown to the user

largeCar.Visible = True

largeCar.Enabled = True

Else

'if the user guesses a cell with no rubbish placed on it

'the cell colour will remain white

DataGridView1.Rows(e.RowIndex).Cells(e.ColumnIndex).Style.BackColor = Color.White

'the cell will be assigned the value of 6 which labels it as 'hit' - stopping the user from reclicking the same cell

grid(e.RowIndex, e.ColumnIndex) = 6

'the number of guesses that the user has left will be decreased by one

noOfguesses = noOfguesses - 1

End If

'if the user has collected all of the rubbish

If rubbishCount = 15 Then

'the winning message will be displayed

winningMessage.Show()

'a procedure will be called to insert the time taken by the user to win the game into a text file

Call newWinningTime()

'the stopwatch timer will be stopped

SW.Stop()

Tmr.Stop()

'the stopwatch timer will be reset

SW.Reset()

'the game grid form will be hidden and disabled

Me.Visible = False

Me.Enabled = False

'refreshes the game grid incase the user chooses to play again

'removes all the controls on the Game Grid form

Me.Controls.Clear()

'loads all of the controls again

InitializeComponent()

'loads everything in the Game Grid form then loads the event again

GameGrid\_Load(e, e)

'resets the timer

SW.Reset()

'resets number of guesses

noOfguesses = 70

'resets number of rubbish pieces found

rubbishCount = 0

End If

'if the user has used all 70 guesses (by clicking on 70 cells in the grid) and not found all of the rubbish

If noOfguesses = 0 And Not rubbishCount = 15 Then

'a message informing the user of their loss will be displayed and enabled

losingMessage.Show()

'hides and disables the game grid

Me.Visible = False

Me.Enabled = False

'the stopwatch timer will be stopped

SW.Stop()

Tmr.Stop()

'the stopwatch timer will be reset

SW.Reset()

'refreshes the game grid incase the user chooses to play again

'removes all the controls on the Game Grid form

Me.Controls.Clear()

'loads all of the controls again

InitializeComponent()

'loads everything in the Game Grid form then loads the event again

GameGrid\_Load(e, e)

'resets the timer

SW.Reset()

'resets number of guesses

noOfguesses = 70

'resets number of rubbish pieces found

rubbishCount = 0

End If

End Sub

'inserts the user's winning time into the text file

Public Sub newWinningTime()

'assigns the variable 'time' the value contained in the stopwatch timer

time = SW.Elapsed.ToString

'gets the filename and path

'declare a variable to store filename

Dim filename As String = "E:\winningTimes.txt"

'if the specified file exists

If System.IO.File.Exists(filename) = True Then

'will write text to the file with the use of a StreamWriter object

Dim objWriter As New System.IO.StreamWriter(filename, True)

'writes the stopwatch time and the date that the user is playing on to file

objWriter.Write(vbNewLine & time & "," & DateTimePicker1.Text)

'closes the StreamWriter object

objWriter.Close()

'if the specified file does not exist then it will be created

Else

'will write text to the file with the use of a StreamWriter object

Dim objWriter As New System.IO.StreamWriter(filename, False)

'writes the stopwatch time and the date that the user is playing on to file

objWriter.Write(vbNewLine & time & "," & DateTimePicker1.Text)

'closes the StreamWriter object

objWriter.Close()

End If

End Sub

Private Sub btnHelp\_Click(sender As Object, e As EventArgs) Handles btnHelp.Click

'displays and enables the form which gives the user instructions on how to play the game

Instructions.Visible = True

Instructions.Enabled = True

End Sub

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

'allows the user to see and interact with the game menu

gameMenu.Visible = True

gameMenu.Enabled = True

'stops the user from being able to see and interact with the game grid

Me.Visible = False

Me.Enabled = False

End Sub

Private Sub btnPause\_Click(sender As Object, e As EventArgs) Handles btnPause.Click

'the paused variable is set to true, signinfying the user's click

paused = True

'the stopwatch timer is paused

SW.Stop()

Tmr.Stop()

'the game grid is disabled to prevent the user from interacting with it while paused

Me.Enabled = False

'the paused notification is shown to inform the user that they have paused the game successfully

pausedForm.Show()

End Sub

Private Sub btnEnd\_Click(sender As Object, e As EventArgs) Handles btnEnd.Click

'ends the game when clicked

End

End Sub

Private Sub btnRestart\_Click(sender As Object, e As EventArgs) Handles btnRestart.Click

'removes all the controls on the Game Grid form

Me.Controls.Clear()

'loads all of the controls again

InitializeComponent()

'loads everything in the Game Grid form then loads the event again

GameGrid\_Load(e, e)

'resets the timer

SW.Reset()

'resets number of guesses

noOfguesses = 70

'resets number of rubbish pieces found

rubbishCount = 0

End Sub

End Class

### Instructions Class

'Garbage Grabberoos Instruction Form

Public Class Instructions

'closes the instruction form when clicked on

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'hides instruction form

Me.Visible = False

'disables the instruction form

Me.Enabled = False

End Sub

Private Sub Instructions\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to create a border around the message (for a cohesive user interface)

Panel1.SendToBack()

'used to centre the instructions form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Sweet Wrapper Class

'Garbage Grabberoos Sweet Wrapper Infographic

Public Class sweetWrapper

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the sweet wrapper infographic form will close

Me.Close()

End Sub

Private Sub sweetWrapper\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the sweet wrapper form on the user's screen

Me.CenterToScreen()

End Sub

Private Sub Panel1\_Paint(sender As Object, e As PaintEventArgs) Handles Panel1.Paint

'used to create a border around image - creates a cohesive user interface

Panel1.SendToBack()

End Sub

End Class

### Plastic Straw Class

'Garbage Grabberoos Plastic Straw Infographic

Public Class plasticStraw

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the plastic straw infographic form will close

Me.Close()

End Sub

Private Sub plasticStraw\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the plastic straw form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Crisp Packet Class

'Garbage Grabberoos Crisp Packet Infographic

Public Class crispPacket

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the crisp packet infographic form will close

Me.Close()

End Sub

Private Sub crispPacket\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the crisp packet form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Plastic Bottle Class

'Garbage Grabberoos Plastic Bottle Infographic

Public Class plasticBottle

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the plastic bottle infographic form will close

Me.Close()

End Sub

Private Sub plasticBottle\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the plastic bottle form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Large Car Class

'Garbage Grabberoos Large Car Infographic

Public Class largeCar

Private Sub btnExit\_Click(sender As Object, e As EventArgs) Handles btnExit.Click

'if the quit button is clicked then the large car infographic form will close

Me.Close()

End Sub

Private Sub largeCar\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the large car form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Paused Class

'Garbage Grabberoos Paused Form

Public Class pausedForm

'the continue button allows the user to continue playing Garbage Grabberoos with the game grid

Private Sub btnContinue\_Click(sender As Object, e As EventArgs) Handles btnContinue.Click

'the game grid is enabled to allow the user to interact with it once more

GameGrid.Enabled = True

'the notification informing the the user that the game is paused is closed

Me.Close()

End Sub

Private Sub pausedForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the paused form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Losing Message Class

'Garbage Grabberoos Game Lost Message

Public Class losingMessage

'if the user clicks on the restart button

Private Sub btnRestart\_Click(sender As Object, e As EventArgs) Handles btnRestart.Click

'the game grid will open in its refreshed state

GameGrid.Visible = True

GameGrid.Enabled = True

'the losing message form will close

Me.Close()

End Sub

'if the user clicks on the show leaderboard button

Private Sub btnLeaderboard\_Click(sender As Object, e As EventArgs) Handles btnLeaderboard.Click

'the speedy searchers leaderboard will open

speedySearchersBoard.Visible = True

speedySearchersBoard.Enabled = True

'the losing message form will close

Me.Close()

End Sub

Private Sub losingMessage\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the losing message form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Winning Message Class

'Garbage Grabberoos Game Won Message

Public Class winningMessage

'if the user clicks on the restart button

Private Sub btnRestart\_Click(sender As Object, e As EventArgs) Handles btnRestart.Click

'the game grid will open in its refreshed state

GameGrid.Visible = True

GameGrid.Enabled = True

'the winning message form will close

Me.Close()

End Sub

'if the user clicks on the show leaderboard button

Private Sub btnLeaderboard\_Click(sender As Object, e As EventArgs) Handles btnLeaderboard.Click

'the speedy searchers leaderboard will open

speedySearchersBoard.Visible = True

speedySearchersBoard.Enabled = True

'the winning message form will close

Me.Close()

End Sub

Private Sub winningMessage\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'used to centre the winning message form on the user's screen

Me.CenterToScreen()

End Sub

End Class

### Speedy Searchers Board Class

'Garbage Grabberoos Leaderboard Form

Public Class speedySearchersBoard

'declares a record structure called searchDetails to store the details of the user's winning time

Public Structure searchDetails

Dim userTime As String

Dim userDate As String

End Structure

Private Sub speedySearchersBoard\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'centers the speedy searchers leaderboard on the user's screen

Me.CenterToScreen()

'declare the userGameResults array of records

Dim userGameResults(199) As searchDetails

'reads user's winning times and the dates that they occurred on from a text file into two arrays of records

ReadIntoArray(userGameResults)

'bubble sorts the time array of records from fastest to slowest time and displays this in a list box

BubbleSortByTime(userGameResults)

End Sub

'reads the data from the text file into two arrays of records

Private Sub ReadIntoArray(ByRef userGameResults() As searchDetails)

'procedure to fill the arrays of records with the correct details

'local variables needed

Dim delimitedData As String = ""

Dim delimeters As Char() = {vbCr, ","}

Dim record As Integer = 0

Dim allData() As String = {}

'reads the data from the file

delimitedData = ReadfromFile(ofdlg1)

'fills the allData array with the read in data

allData = delimitedData.Split(delimeters)

'sets the arrays of records to the correct size (allData - 1 is used to ignore the blank line present at the end of the text file)

ReDim userGameResults(Int((UBound(allData) - 1) / 2))

'loops through the data and stores in the two arrays of records

For counter = 0 To UBound(allData) - 1 Step 2 '(allData - 1 is used to ignore the blank line present at the end of the text file)

userGameResults(record).userTime = allData(counter)

userGameResults(record).userDate = allData(counter + 1)

record += 1

Next

End Sub

'bubbles sorts the data

Private Sub BubbleSortByTime(ByRef userGameResults() As searchDetails)

'declares a variable to represent the maximum size of the array of records

Dim sortingElements As Integer

'for loop control variable'

Dim i As Integer

'temporary storage 'time' variable which is compared to the next in the array

Dim temp As Object

'temporary storage 'date' variable which is moved in tandem with its corresponding time

Dim temp2 As Object

'variable used to indicate whether a swap has occurred

Dim swapped As Boolean

'assigns the variable 'sortingElements' the value of the last number in the time array of records

sortingElements = UBound(userGameResults)

Do

'initially no swap has occured

swapped = False

'loops from start to end of the record using the sortingElements variable

For i = 1 To sortingElements - 1

'if the next number in the time array of records is smaller than the current number

If userGameResults(i).userTime > userGameResults(i + 1).userTime Then

'the current number becomes the temporarily stored variable

temp = userGameResults(i).userTime

'the current date becomes the temporarily stored date

temp2 = userGameResults(i).userDate

'the current number is moved along by one place

userGameResults(i).userTime = userGameResults(i + 1).userTime

'the current date is moved along by one place

userGameResults(i).userDate = userGameResults(i + 1).userDate

'the next number becomes the temporarily stored variable

userGameResults(i + 1).userTime = temp

'the next date becomes the temporarily stored variable

userGameResults(i + 1).userDate = temp2

'swapped is set to true to indicate that a swap has occured

swapped = True

End If

Next

'sortingElements is reduced by one to show that there is one less number to sort

sortingElements = sortingElements - 1

'the loop will only end when no more swaps are occurring

Loop Until swapped = False

'resets the leaderboard each time it is shown

lstSpeedySearchers.Items.Clear()

'if no more swaps are occurring

If swapped = False Then

'loops from the start to the end of the records

For i = 0 To UBound(userGameResults)

'displays the times and dates in a list box

lstSpeedySearchers.Items.Add(" " & userGameResults(i).userTime & " " & userGameResults(i).userDate)

Next

End If

End Sub

'if the back to grid link label is clicked

Private Sub lblBackToGrid\_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles lblBackToGrid.LinkClicked

'the game grid is opened and enabled

GameGrid.Visible = True

GameGrid.Enabled = True

'the speedy searchers board is hidden and disabled

Me.Visible = False

Me.Visible = False

End Sub

'if the menu button is clicked

Private Sub btnMenu\_Click(sender As Object, e As EventArgs) Handles btnMenu.Click

'the game menu is opened and enabled

gameMenu.Visible = True

gameMenu.Enabled = True

'the speedy searchers board is hidden and disabled

Me.Visible = False

Me.Visible = False

End Sub

'if the quit button is clicked

Private Sub btnEnd\_Click(sender As Object, e As EventArgs) Handles btnEnd.Click

'the game will close and end

End

End Sub

End Class

## Screenshots of User Interface

For Sign Up

The user can sign up to form an account by entering an 8-character username and 6-character password. In this example “n*ewacc27”* is entered for username and *“testpa”* is entered for password.

Graphical user interface

Description automatically generated

When the user clicks the sign-up button, if the username and password both successfully pass validation (i.e. the username is 8 characters AND the password is 6 characters), and are not linked to an already existing account in the Garbage Grabberoos Users Database, then the below success message is shown.

Graphical user interface, text, application

Description automatically generated

This will result in the new user’s username and password being inserted into the Garbage Grabberoos Users Database as a new record. The new account can be seen as the most recent entry below.

Table

Description automatically generated

However, if the inputted username and password do not successfully pass validation when the sign-up button is clicked (i.e one of the textboxes is empty or the entered username isn’t 8 characters/the entered password isn’t six characters, then the below message is shown.

Graphical user interface, text, application

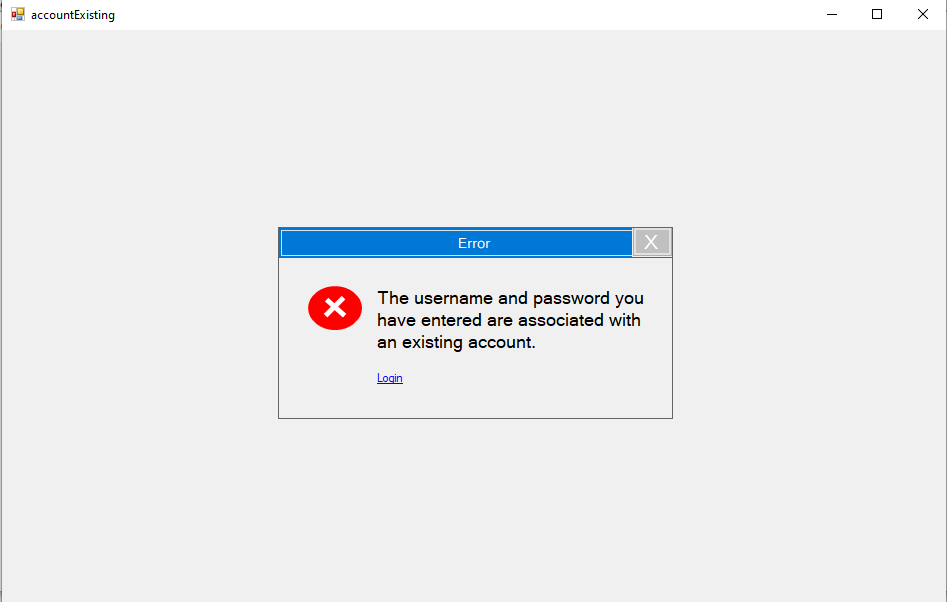
Description automatically generated

Alternatively, if the entered username and password correspond to an account that already exists in the Garbage Grabberoos Users Database (as shown below with the username *“ecoluvr8”* and the password “*iloveg”)*

Graphical user interface

Description automatically generated with medium confidence

Then the below ‘account existing’ error message will be displayed on the user’s screen and the user is prompted to login.



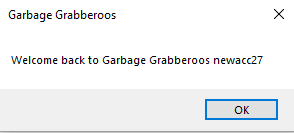
For login

The user can login to an existing account by entering a previously created 8-character username and a previously 6-character password. In this example “n*ewacc27”* is entered for username and *“testpa”* is entered for password (the account was already created during sign up – see [For Sign Up](#Bookmark1) for reference)

Graphical user interface

Description automatically generated

When the user clicks the login button, if the username and password both successfully pass validation (i.e. the username is 8 characters AND the password is 6 characters), and are linked to an already existing account in the Garbage Grabberoos Users Database, then the below welcome message is shown, where the username is specific to the one entered by the user (*“newacc27”* in this example)

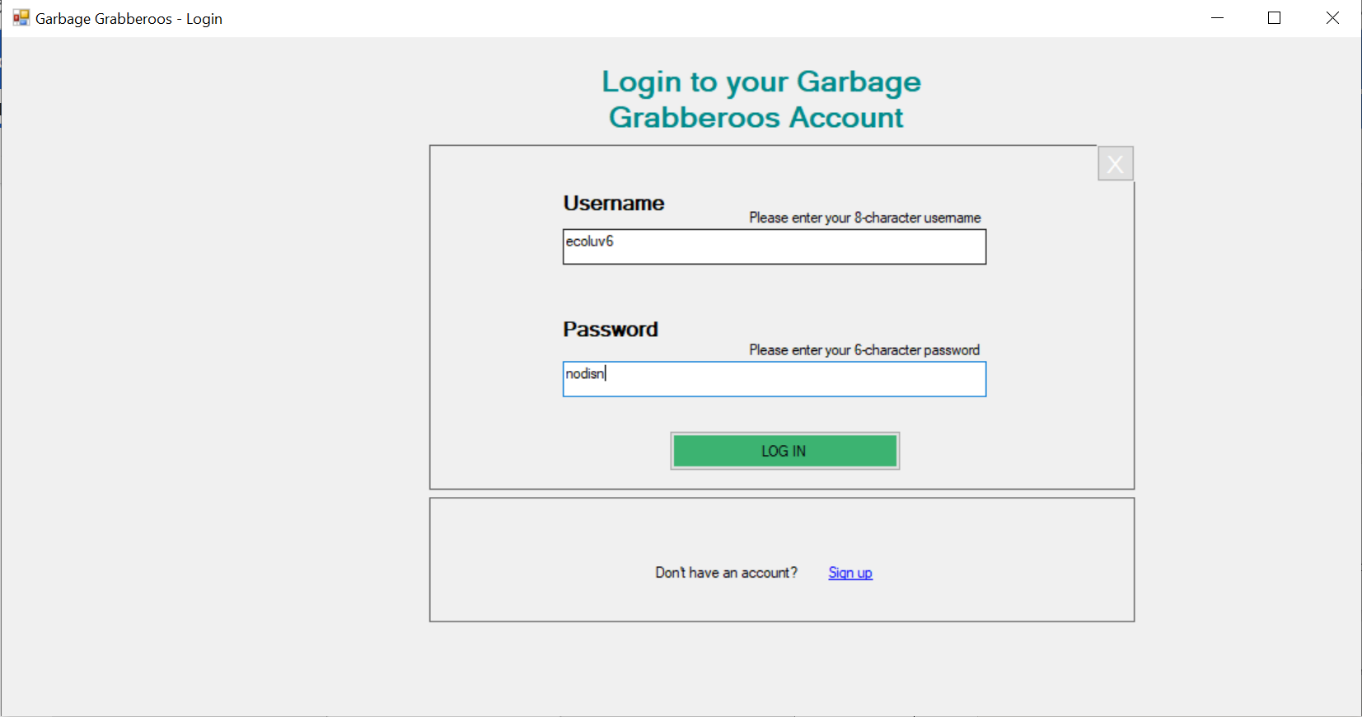


However, if the inputted username and password do not successfully pass validation when the login button is clicked (i.e. one of the textboxes is empty, the entered username isn’t 8 characters or the entered password isn’t 6 characters) then the below error message is shown.

Graphical user interface, text, application

Description automatically generated

Alternatively, if the inputted username and password both successfully pass validation when the login button is clicked (i.e no textboxes are empty, the entered username is 8 characters and the entered password is 6 characters) but they do not correspond to an account already in the Garbage Grabberoos Users Database (shown here with the username *“ecoluvr6”* and password *“nodisn”*)



Then the below error message is shown, and the user is prompted to sign up.

Graphical user interface, text, application

Description automatically generated

For gameplay

Once the user has successfully logged in, they will be taken to the game menu.

Graphical user interface, website

Description automatically generated

If the user clicks on the “START GAME” button on the game menu, then they will be shown an instruction form explaining how to play the game.

Graphical user interface, text, application

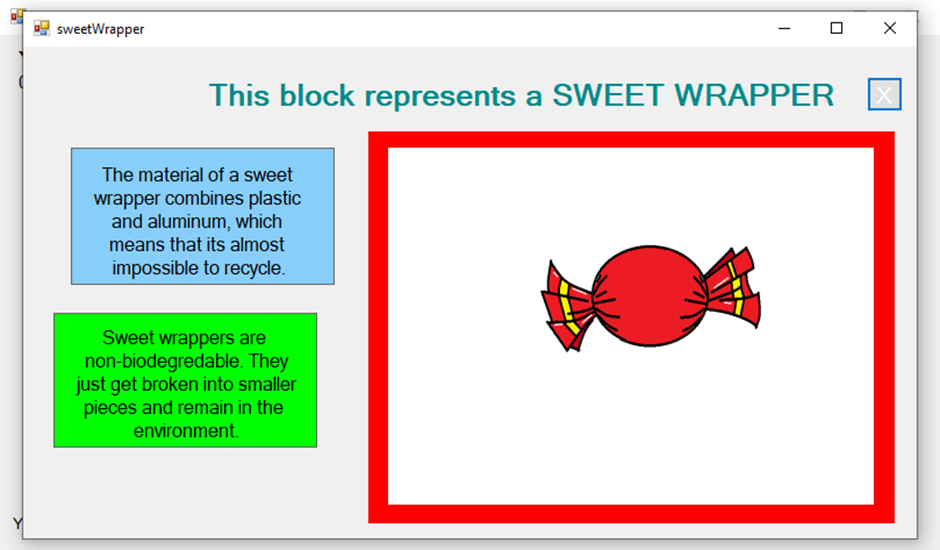
Description automatically generated

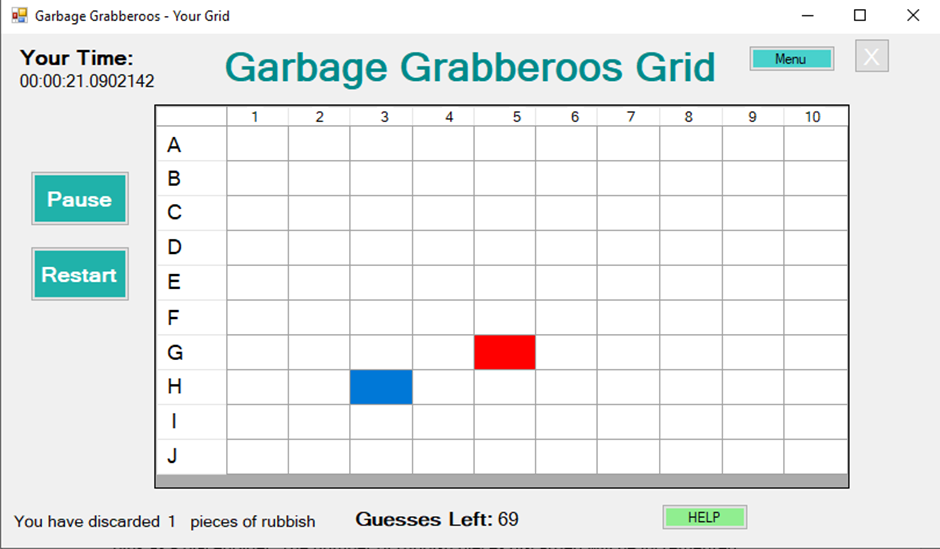
If the user clicks on the “START GAME” button on the How to Play form, then the game grid will be displayed on the user’s screen for gameplay. A user can click on a grid square to make a guess at the randomised location of a rubbish piece on the grid. This will increment the number of guesses.

Calendar

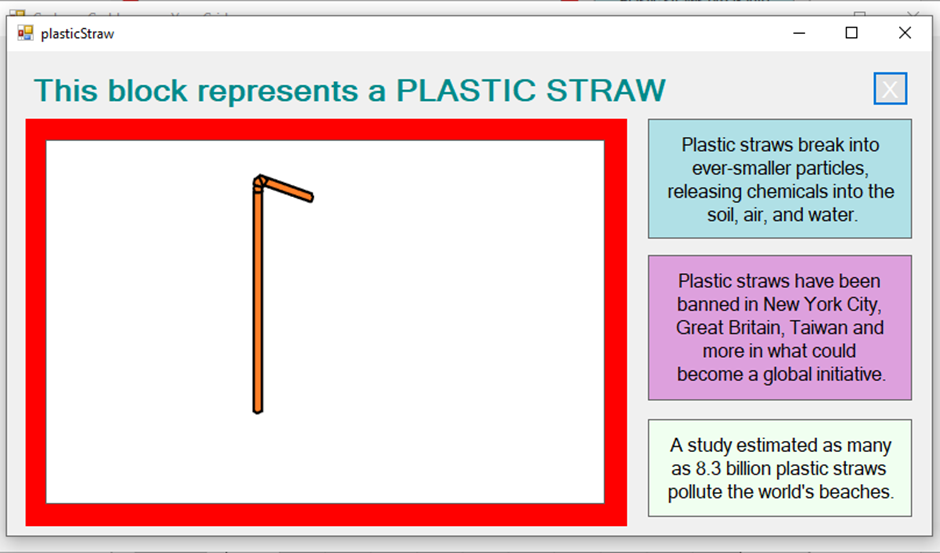
Description automatically generated

If the user clicks on the single grid square that represents a sweet wrapper, then the sweet wrapper infographic will be displayed on the user’s screen and the clicked grid square’s colour will be turned red as a placeholder. The number of rubbish pieces discarded will be incremented.





If the user clicks on the one of the two grid squares that represents a plastic straw, then the plastic straw infographic will be displayed on the user’s screen and the clicked grid square’s colour will be turned blue as a placeholder. The number of rubbish pieces discarded will be incremented.



Chart

Description automatically generated with low confidence

If the user clicks on the one of the three grid squares that represents a crisp packet, then the crisp packet infographic will be displayed on the user’s screen and the clicked grid square’s colour will be turned green as a placeholder. The number of rubbish pieces discarded will be incremented.

Graphical user interface, application

Description automatically generated

Chart

Description automatically generated with low confidence

If the user clicks on the one of the four grid squares that represents a plastic bottle, then the plastic bottle infographic will be displayed on the user’s screen and the clicked grid square’s colour will be turned pink as a placeholder. The number of rubbish pieces discarded will be incremented.

Graphical user interface, application

Description automatically generated

Chart

Description automatically generated

If the user clicks on the one of the five grid squares that represents a large car, then the large car infographic will be displayed on the user’s screen and the clicked grid square’s colour will be turned purple as a placeholder. The number of rubbish pieces discarded will be incremented.

Text

Description automatically generated with medium confidence

A picture containing chart

Description automatically generated

If the user clicks on the “Pause” button on the game grid form then the below message will be displayed on the user’s screen, and the game grid will be made unable to interact with.

Graphical user interface, application

Description automatically generated

If the user clicks on a grid square that they have previously clicked, then the below error message will be shown. The number of guesses that the user has made will not be changed.

A picture containing timeline

Description automatically generated

If the user clicks on the “Restart” button on the game grid form then all of the form controls will be reset, number of guesses left will be reverted back to 70, and the number of rubbish pieces discarded will be initialised.

Calendar

Description automatically generated

If the user clicks on the HELP button then an instructions page form will be shown outlining the types and sizes of the rubbish to be found and reminding the user of how to play the game (with a tip of clicking grid squares close to a successful grid square).

Graphical user interface, text, application

Description automatically generated

If the user does not successfully discard all 15 pieces of rubbish before their number of guesses left reaches 0 then the below losing message will be shown.

Graphical user interface, application, website

Description automatically generated

If the user successfully discards all 15 pieces of rubbish before their number of guesses left reaches 0 then the below success message will be shown.

Graphical user interface, application

Description automatically generated

When the user wins, the time that it took them to win the game along with the date of the win will be added into the Speedy Searcher’s Leaderboard in the appropriate place, as it is sorted from fastest time to slowest time. This can be seen when the “VIEW LEADERBOARD” button is clicked on the losing or winning messages, or on the game menu. Here the new entry is highlighted.

Graphical user interface, table

Description automatically generated

## Research and development of new skills and/or knowledge

**Timer**

I have chosen **displaying a form for a specified period of time** as my new skill to research and develop that is beyond Advanced Higher level, since it adds functionality and was dually interesting to research:

Due to the input validation incorporated into my program, I required several error and success messages to be shown to the user at various points in the game. These points are during sign up, during login and during game play. The user can fail to sign up for an account because their entries do not meet the required criteria for username and/or password. This means that they have either entered a username or password that is too long or too short, or left one or both of the textboxes empty. If this occurs, a notification will be displayed informing the user of this.

The visual basic code for this process involved the error message form called *unsuccessfulSignUporLogin* being shown upon the above instance of failed sign up. However, when I implemented this, I was met with an error where all of the form controls on the *unsuccessfulSignUporLogin* form were white, and the *unsuccessfulSignUporLogin* form could not be interacted with or clicked on as the program became unresponsive (see [Failed Sign Up Error](#Timer) for reference). This was an issue as I only wanted the error message to be displayed for a few seconds in order to allow the user to acknowledge that their entered username and/or password does not match the specified conditions, but then also allow them to enter another username and password into the form, correcting earlier mistake.

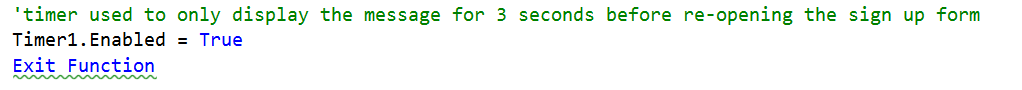
Upon using breakpoints to try to locate the code causing the issue, I was able to conclude that the error message was part of an infinite loop that had no way of being broken, and realised that I would have break this loop forcefully. This prompted me to research on StackOverflow for ways in which a form can be displayed for a limited amount of time. I consulted the following link [vb.net - Simple, non-blocking way to sleep? - Stack Overflow](https://stackoverflow.com/questions/2237567/simple-non-blocking-way-to-sleep), and it allowed me to discover the Sleep function. I attempted to resolve the error with the Sleep, but it did not work and left my program unresponsive and hanging. The user would not be able to dismiss the notification without doing so forcefully, and would have to manually reload the form.

Graphical user interface, text, application

Description automatically generated

This then prompted me to do further research using StackOverflow and CodeProject, but more specifically than before: for ways to show a form for only a short amount of time before hiding it, all while **maintaining form responsivity**. I found that the Timer function is able to do so using the following links: [[Solved] display a form for a period of time - CodeProject](https://www.codeproject.com/questions/169892/display-a-form-for-a-period-of-time) & <https://stackoverflow.com/questions/25778041/vb-net-create-and-close-forms-after-certain-tim>.

This inspired me to implement the Timer function within my program, using both StackOverflow and CodeProject research and individual trial and error to be able to do this successfully, as shown below.



Graphical user interface, text, application

Description automatically generated

**Try / Catch Function**

I have chosen ‘**exception handling with Try/Catch**’ as the second of my skills to research and develop that are beyond Advanced Higher level. As the Sign-Up system was difficult to implement and crashed my program in response to several issues: the unintentional use of a reserved access variable, the database file not being .mdb and more (see Log of ongoing testing), it occurred to me that it would have been useful to receive an error message on each of these occasions, instead of the system simply crashing. This prompted me to research ways to program my system’s reactions to an error with the insertion of a user’s new account into the database that could occur during the Sign-Up process, which in turn would improve the entire program’s robustness. I used the following links to teach myself about this function, before writing the code to implement it myself: <https://stackoverflow.com/questions/52881701/visual-basictry-catch-exception-handling-issues> & <https://stackoverflow.com/questions/41103931/how-to-solve-vb-net-database-connection-error-try-catch-database> & <https://www.homeandlearn.co.uk/NET/nets5p4.html> . In response to the failure of the addition of a new account (that has passed all validation) into the Garbage Grabberoos Database, the system will now display the standard Visual Basic exception message (ex.Message), without crashing the program.

A picture containing logo

Description automatically generated

Text

Description automatically generated

## Log of ongoing testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Location** | **Date** | **Issue Description** | **Screenshots (Error & Solution)** | **Solution** | **References Used** |
| SignUp Class | Error Occurred: 22/01/23  Error  Resolved:  24/01/23 | The Visual Basic Powerpack shape ‘rectangle’ cannot be added to the form without producing a Call Stack error, which was an issue as I wanted to use it to provide a border around the form text (allowing the final design to mirror my wireframes). | Error:  Graphical user interface, application  Description automatically generated  Solution: | I have used a panel to surround the form text instead, and enabled a ‘FixedSingle’ Border Style to provide the black bordered outline. | No references used |
| Resources.Rex | Error Occurred: 25/01/23  Error  Resolved:  25/01/23 | I deleted an image file which was no longer required for the program (as I had created another graphic), which produced an error where the specified file path in the Resources.Rex file could not be located. | Error:  A screenshot of a computer  Description automatically generated  Solution:  A screenshot of a computer  Description automatically generated  *Deleted the above lines of code* | I deleted the lines of code in the Resources.Rex file that referenced the unused image file, then did the same for two others as I no longer required those either. | [https://stackoverflow.com/visual-studio-creates-invalid-resx-files](https://stackoverflow.com/questions/4132399/visual-studio-creates-invalid-resx-files) |
| unsuccesfulSignUp or Login class | Error Occurred: 26/01/23  Error  Resolved:  29/01/23 | The unsuccessful signup or login message that was being displayed as part of input validation was the main component of a loop that could not be ended, causing form controls to become white and the form to stop being interactive. | Error:  Graphical user interface, application, Word  Description automatically generated  Solution: | I had to end the loop forcefully by exiting the function. I also had to find a way to limit the time that the form would be displayed for, which prompted my research and consequent implementation of the Timer function (see [New Skills](#_Toc128386047) for reference) | [https://stackoverflow.com/close-form-after-10-seconds](https://stackoverflow.com/questions/4506033/close-form-after-10-seconds)  [https://social.msdn.microsoft.com/auto-close-form-after-time-vb](https://social.msdn.microsoft.com/Forums/windows/en-US/9de7462a-da49-4eb9-8a5a-a74b08d9c482/auto-close-form-after-time-vb-vs05?forum=winforms)  [https://stackoverflow.com/simple-non-blocking-way-to-sleep](https://stackoverflow.com/questions/2237567/simple-non-blocking-way-to-sleep) |
| SignUp Class | Error Occurred: 29/01/23  Error  Resolved:  29/01/23 | The Garbage Grabberoos Users database could not be located by the program, which stopped the connection from opening. | Error:  Graphical user interface, text, application, email  Description automatically generated  Solution: | I modified the access database, choosing to save it as a .mdb file instead of a .aacdb file, which successfully resolved the error. | No references used |
| GameGrid Class | Error Occurred: 02/02/23  Error  Resolved:  03/02/23 | The DataGridView grid that I had created was not filling its allotted space, which decreased the game’s visual appeal as well as its usability since the user has less space to click on the game grid with. | Error:  Graphical user interface, application, table  Description automatically generated  Solution:  *This is the final result*    *This is the code that produced the result above* | I had to alter the row and column widths of the grid until I found a combination that allowed it to fill its allotted space. The solution result was achieved with a column width of 55 and a row width of 30. | [https://stackoverflow.com/adjust-datagridviews-columns-to-fill-available-space](https://stackoverflow.com/questions/9024967/adjust-datagridviews-columns-to-fill-available-space-if-the-grid-is-smaller-and#:~:text=Click%20over%20the%20DataGridView%20and,what%20you're%20looking%20for) |
| speedySearchersBoard Class | Error Occurred: 07/02/23  Error  Resolved:  10/02/23 | Upon loading the Speedy Searchers board, I received an error stating that it could not add any further information into the usertime array of records from the specified text file due to the index being outside of the bounds of the array. Since I did not specify a number to end the loop and opted for Ubound(allData), this error was unexpected. | Error:  Solution:  *Removed the extra line shown below*  A screenshot of a computer  Description automatically generated | I opened the text file that was being read from in Notepad and could not find any issues with it. However, when I opened the same text file in Notepad++, it became apparent that there was an extra blank line at the end of the file. I removed this blank line, which resolved the error. | [https://stackoverflow.com/index-out-of-bound-of-the-array-in-vb](https://stackoverflow.com/questions/24207079/index-out-of-bound-of-the-array-in-vb)  [https://www.vbforums.com/RESOLVED-Index-was-outside-the-bounds-of-the-array](https://www.vbforums.com/showthread.php?680469-RESOLVED-Index-was-outside-the-bounds-of-the-array) |
| speedySearchersBoard Class | Error Occurred:  10/02/23  Error Resolved: 11/02/23 | Although the Speedy Searcher’s Board would load successfully when I manually removed the extra blank line from Notepad++, I wanted the program to be able to automatically ignore the extra blank line after each new entry without producing the same error as above. | Error:  *This is the error produced*  Graphical user interface, text, application  Description automatically generated  *This is the code that caused the above error*  Graphical user interface, text, application  Description automatically generated  Solution:  Graphical user interface, text, application  Description automatically generated | I modified the userGameResults record so that all of the lines in the text file except the last line would be used to provide the information to be written into the two arrays of records by using “allData - 1” instead of “allData”, which resolved the error. | <https://www.vbforums.com/showthread.php?682296-RESOLVED-Remove-all-blank-lines-in-textfile>  <https://vbdotnetforums.com/threads/remove-blank-line-s-from-text-file.47467/>  *Neither reference used, but provided inspiration for my solution* |
| SignUp Class | Error Occurred:  15/02/23  Error resolved: 17/02/23 | When inserting a new user into the Garbage Grabberoos Users Database upon sign up, an error was produced. Since the syntax for the INSERT INTO statement was correct, the error was unexpected. | Error:  *This is the error produced*    *This is the code that caused the above error*    Solution: | Username and Password are both reserved variables required by access, so I added square brackets around variable names which resolved the error. | <https://stackoverflow.com/syntax-error-in-insert-into-statement-vb-net-and-ms-access> -  <https://stackoverflow.com/how-to-fix-syntax-error-in-insert-into-statement-in-vbnet> |
| Sign Up Class | Error Occurred:  16/02/23  Error resolved: 16/02/23 | There was no specific input validation for an empty textbox during sign up, which meant that both the existing account error message and the unsuccessful sign up or login error message would be displayed if a textbox was empty. This decreased usability as the existing account error message does not apply here. | Error:    Solution: | As well as the already present input validation on the lengths of username and password, I added input validation for an empty textbox where the unsuccessful sign up or login error message will be shown, and the loop will not be allowed to end. I also incorporated this same validation into the Login class. | No references used |
| Speedy Searchers Board Class | Error Occurred:  18/02/23  Error resolved: 20/02/23 | Only three out of the four results held in the userGameResults record were being displayed in the listbox, even though the loop was designed to display all of the results bubble sorted from low to high. | Error:  *This is the error produced*  Text  Description automatically generated with medium confidence  *This is the code that caused the above error*    Solution: A screenshot of a computer  Description automatically generated with medium confidence | I had to modify the For loop to incorporate the userGameResults record instead of the sorting elements variable in order to ensure that all of the data was being displayed, since the sorting elements variable loses some data each time that piece of data is successfully sorted. | No references used |
| Speedy Searchers Board Class | Error Occurred:  21/02/23  Error resolved: 22/02/23 | All of the data in the time array of records of the userGameResults record was being successfully sorted from low to high for the Speedy Searchers leaderboard, except the very first value which was not being sorted and was instead placed as the slowest time. | Error:  *This is the error produced*    *This is the code that caused the above error*  *Text, letter  Description automatically generated*  Solution:  *This is the final result*    *This is the code that produced the above result*  Text, letter  Description automatically generated | I had to modify the loop to begin at 1 instead of 0 to ensure that the first value in the userGameResults record was also being sorted. | [VB.Net program to sort an array in ascending order using bubble sort (includehelp.com)](https://www.includehelp.com/vb-net/sort-an-array-in-ascending-order-using-bubble-sort.aspx) |
| Speedy Searchers Board Class | Error Occurred:  22/02/23  Error resolved: 22/02/23 | The results in the Speedy Searchers leaderboard were not displaying as they have been, since the wrong dates were being displayed alongside their supposed corresponding times. | Error:      *The date and time values in the text file do not match the date and time values displayed in the textbox*  Solution:    Text  Description automatically generated | I concluded that the date array of records was being sorted in ascending order as well as the time array of records, while I wanted the time array of records to be the only array sorted. To resolve this, I added the date array of records into my bubble sort algorithm. I did this by creating another temp variable called ‘temp2’ and moving temp2 in response to the movement of the already created ‘temp’ variable. | No references used |
| GameGrid Class | Error Occurred:  23/02/23  Error resolved: 25/02/23 | After fixing the error detailed above, a new issue arose where each new entry for date and stopwatch time after a game win would be added onto the same line as the previous entry. This prevented the Speedy Searchers board from loading properly. | Error:  Graphical user interface, text, application  Description automatically generated  Solution:Graphical user interface, text, application  Description automatically generated | I modified the code for writing the Date and Stopwatch time to a file to include vbNewLine, which will create a carriage return and resolve the issue. | [How to go to new line in a text document using VB.Net - Stack Overflow](https://stackoverflow.com/questions/10095155/how-to-go-to-new-line-in-a-text-document-using-vb-net)  [Get input from user & display information in message box | vbNewLine | Visual Basic | Chapter # 4 - YouTube](https://www.youtube.com/watch?v=bbzub69B3oc) |
| Sign Up Class | Error Occurred:  27/02/23  Error resolved: 28/02/23 | An error occurred because of an issue with parameter passing during sign up. This was due to new entry record into the Garbage Grabberoos database not containing a value for password. | Error:  Graphical user interface  Description automatically generated    Solution:  *This is the final result*    *This is the code used to achieve the final result – formal and actual parameters now match* | The error occurred because variables containing no value were being passed into the actual parameters of the newUser function used during sign up, since they were not the values specified in the formal parameter. I modified this, which resolved the error. | No references used |
| Losing Message Class & Game Grid Class | Error Occurred: 01/03/23  Error  Resolved:  01/03/23 | Upon my loss of the game, the losing message was displayed. I clicked the Restart button on the losingMessage form to play the game again, but the game grid that was opened was unresponsive and I was unable to interact with it. | Error:  *This is the error produced*    *This is the code that caused the above error*  Graphical user interface, text, application  Description automatically generated  Solution:  *This is the final result*    *This is the code that produced the above result* | In my code, I had included a line that disabled the game grid if the user had not successfully discarded all of the pieces of rubbish. I did this along with making the game grid invisible in order for the losingMessage form to be shown, but I realised that it was not necessary and prevented the game grid from loading properly if the Restart button was clicked on the losingMessage form. I modified the code by removing this line, which resolved the error. I also did this for the Winning Message class, as I had made the same error there too. | No references used |

# TESTING

## Final Test Plan

### **Interface Testing Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Element | Test Description | Issues Identified |
| 1 | Sign Up Page:  Form(textboxes),  Sign Up button,  Login linklabel,  Quit button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sign Up form contains two textboxes – one for username and one for password * The Sign Up form contains a sign up button which opens the Login form when clicked if validation is passed and opens one of 2 error messages if validation is not is passed * The Sign Up form contains a login linklabel which closes the Sign Up form and opens the Login form when clicked * The Sign Up form contains a Quit button that ends the entire program if clicked   Expected Output:   * The elements listed above match the Sign Up form wireframes correctly | Passed: no issues identified |
| 2 | Login Page:  Form (textboxes), Login button  Sign Up linklabel,  Quit button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Login form contains two textboxes – one for username and one for password * The Login form contains a login button which opens a welcome message when clicked if validation is passed and opens one of 2 error messages if validation is not passed * The Login form contains a login linklabel which closes the Login form and opens the Sign Up form when clicked * The game menu page contains a Quit button that ends the entire program if clicked   Expected Output:   * The elements listed above match the Login form wireframes correctly | Passed: no issues identified |
| 3 | Game Menu Page:  Start Game Button,  View Leaderboard Button,  Quit Game Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The game menu page contains a START GAME button that closes the gameMenu form and opens the howToPlay form if clicked * The game menu page contains a VIEW LEADERBOARD button that closes the gameMenu form and opens the speedySearchersBoard form if clicked * The game menu page contains a QUIT GAME button that ends the entire program if clicked   Expected Output:   * The elements listed above match the Game Menu form wireframe correctly | Passed: no issues identified |
| 4 | How to Play Page:  Start Game Button,  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The how to play page contains a START GAME button that closes the howToPlay form and opens the GameGrid form if clicked * The game menu page contains a Quit button that closes the howToPlay form and opens the GameGrid form if clicked   Expected Output:   * The elements listed above match the How to Play form wireframe correctly | Passed: no issues identified |
| 5 | Game Grid Page:  DataGridView (changing colour grid squares in presence of different rubbish types),  Pause Button,  Restart Button,  Quit Button,  Help Button,  Menu Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The game grid form contains a 10 by 11 grid of squares formed from a 2D array * The clicking of a grid square on the grid that contains no rubbish will keep the square colour as white, but notify the user to know that they have just clicked there because of a temporary navy square * The clicking of a grid square on the grid representing a sweet wrapper will change the square colour from white to red * The clicking of a grid square on the grid representing a plastic straw will change the square colour from white to blue * The clicking of a grid square on the grid representing a crisp packet will change the square colour from white to green * The clicking of a grid square on the grid representing a plastic bottle will change the square colour from white to pink * The clicking of a grid square on the grid representing a plastic bottle will change the square colour from white to purple * The game grid page contains a Pause button which will open the paused form and disable the game grid if clicked * The game grid page contains a Restart button which will reset all form controls, initialise the number of rubbish pieces discarded, and revert number of guesses to 70 if clicked * The game grid page contains a HELP button which will open the Instructions form while keeping the GameGrid form open if clicked * The game grid page contains a Quit button which will close the entire program if clicked * The game grid page contains a Menu button will close the GameGrid form and open the gameMenu form if clicked   Expected Output:   * The elements listed above match the GameGrid form wireframe correctly | Passed: no issues identified |
| 6 | Speedy Searchers Leaderboard Page:  Menu Button,  Quit Button,  ‘Go back to grid’ Linklabel | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Speedy Searchers Leaderboard page contains a Menu button which will close the speedySearchersBoard form and open the gameMenu form if clicked * The Speedy Searchers Leaderboard page contains a Quit button which will close the entire program if clicked * The Speedy Searchers Leaderboard page contains a ‘Go back to grid’ linklabel which will close the speedySearchersBoard form and open the GameGrid form if clicked   Expected Output:   * The elements listed above match the speedySearchersBoard form wireframe correctly | Passed: no issues identified |
| 6 | Unsuccessful SignUp or Login Page:  Sign Up Button,  Login Button,  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Unsuccessful SignUp or Login Page contains a quit button which will close the unsuccessfulSignUpOrLogin form if clicked * The Unsuccessful SignUp or Login Page contains a Sign Up button which will close the unsuccessfulSignUpOrLogin form and open the SignUp form if clicked * The Unsuccessful SignUp or Login Page contains a Login button which will close the unsuccessfulSignUpOrLogin form and open the Login form if clicked   Expected Output:   * The elements listed above match the Unsuccessful SignUp or Login form wireframe correctly | Passed: no issues identified |
| 7 | Account Existing Page:  Quit Button,  Login Linklabel | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Account Existing Page contains a quit button that will close the accountExisting form and open the Login form if clicked * The Account Existing Page contains a Login linklabel that will close the accountExisting form and open the Login form if clicked   Expected Output:   * The elements listed above match the Account Existing form wireframe correctly | Passed: no issues identified |
| 8 | Successful Sign-Up Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Successful Sign-Up Page contains a quit button which will close the successfulSignUp form and open the Login form if clicked   Expected Output:   * The elements listed above match the Successful SignUp form wireframe correctly | Passed: no issues identified |
| 9 | No Account Found Page:  Quit Button,  ‘Sign up for an account’ Linklabel | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The No Account Found Page contains a quit button which will close the noAccountFound form and open the Login form if clicked * The No Account Found Page contains a ‘Sign up for an account’ linklabel which will open the SignUp form if clicked   Expected Output:   * The elements listed above match the No Account Found form wireframe correctly | Passed: no issues identified |
| 10 | Instructions Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Instructions Page contains a quit button which will close the Instructions form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the Instructions form wireframe correctly | Passed: no issues identified |
| 11 | Paused Page:  Continue Game button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Paused Page contains a Continue Game button which will close the paused form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the Paused form wireframe correctly | Passed: no issues identified |
| 11 | Losing Message Page:  Restart Game Button,  View Leaderboard button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Losing Message Page contains a RESTART GAME button which will close the losingMessage form and open the GameGrid form, where all form controls will be reset, the number of rubbish pieces discarded will be initialised, and the number of guesses will be reverted to 70 if clicked * The Losing Message Page contains a VIEW LEADERBOARD button which will close the losingMessage form and open the speedySearchersBoard form if clicked   Expected Output:   * The elements listed above match the Losing Message form wireframe correctly | Passed: no issues identified |
| 12 | Winning Message Page:  Restart Game Button,  View Leaderboard button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Winning Message Page contains a RESTART GAME button which will close the winningMessage form and open the GameGrid form, where all form controls will be reset, the number of rubbish pieces discarded will be initialised, and the number of guesses will be reverted to 70 if clicked * The Winning Message Page contains a VIEW LEADERBOARD button which will close the and open the speedySearchersBoard form if clicked   Expected Output:   * The elements listed above match the Winning Message form wireframe correctly | Passed: no issues identified |
| 13 | Sweet Wrapper Infographic Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sweet Wrapper Infographic Page contains a quit button which will close the sweetWrapper form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the sweetWrapper form wireframe correctly | Passed: no issues identified |
| 14 | Plastic Straw Infographic Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sweet Wrapper Infographic Page contains a quit button which will close the plasticStraw form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the plasticStraw form wireframe correctly | Passed: no issues identified |
| 15 | Crisp Packet Infographic Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sweet Wrapper Infographic Page contains a quit button which will close the crispPacket form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the crispPacket form wireframe correctly | Passed: no issues identified |
| 16 | Plastic Bottle Infographic Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The clicking of the quit button will close the plasticBottle form while the GameGrid form remains open   Expected Output:   * The elements listed above match the plasticBottle form wireframe correctly | Passed: no issues identified |
| 17 | Large Car Infographic Page:  Quit Button | Test that:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sweet Wrapper Infographic Page contains a quit button which will close the largeCar form while the GameGrid form remains open if clicked   Expected Output:   * The elements listed above match the largeCar form wireframe correctly | Passed: no issues identified |

### **Final and Requirements Testing Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Functional Requirements | | | |
| 18.0 | Sign Up System | Test that:   * The user can enter their details into the following fields: Username, Password – and both are sent to the Garbage Grabberoos Users Database if all validation is passed (length check on Username of 8 characters, length check on Password of 6 characters, presence check on both textboxes, a check for an existing account in Garbage Grabberoos Users Database linked to the entered details)   Do this by:   * Entering a correct set of test credentials into the Sign Up form (‘*validuse*’ for username and ‘*valpas’* for password) and using the successful sign up message opening as well as a new Garbage Grabberoos Users database entry as an indication that the system is performing accurately   Expected Output:   * When the credentials are submitted, a successful sign up message should open in order to prompt the user to login. | Passed: no issues identified |
| 18.1 | Sign Up System | Test that:   * In the case of the user not entering their details into a required field (one or both of Username, Password), make sure the system correctly detects an empty textbox and displays the correct response modal   Do this by:   * Consecutively missing out each of the required (Username and Password) fields in turn, and then both fields at once and validating that the negative response modal is shown upon each submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form. | Passed: no issues identified |
| 18.2 | Sign Up System | Test that:   * In the case of the user entering an invalid username (not 8 characters), make sure the system correctly detects an invalid input and displays the correct response modal   Do this by:   * Entering all valid credentials other than the username (enter “invalidusername” instead) and validate that the negative response modal is shown upon submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form. | Passed: no issues identified |
| 18.3 | Sign Up System | Test that:   * In the case of the user entering an invalid password (not 6 characters), make sure the system correctly detects an invalid input and displays the correct response modal   Do this by:   * Entering all valid credentials other than the password (enter “invalidpassword” instead) and validate that the negative response modal is shown upon submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form. | Passed: no issues identified |
| 18.4 | Sign Up System | Test that:   * In the case of the user entering a username and password that correspond to an account that already exists in the Garbage Grabberoos Users Database, make sure the system detects this and displays the correct response modal   Do this by:   * Entering all valid credentials that correspond to an account that is already present in the database (“ecoluvr8” for username and “iloveg” for password) to validate that the negative response modal is shown upon submission, with the error message “The username and password you have entered are associated with an existing account”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form. | Passed: no issues identified |
| 18.5 | Login System | Test that:   * The user can enter their details into the following fields: Username, Password – and both are used to search the Garbage Grabberoos Users Database to login if all validation is passed (length check on Username of 8 characters, length check on Password of 6 characters, presence check on both textboxes, a check for an existing account in Garbage Grabberoos Users Database linked to the entered details)   Do this by:   * Entering a correct set of test credentials (‘*validuse*’ for username and ‘*valpas’* for password) into the Login form and using the welcome message being displayed with the entered username as an indication that the system is performing accurately   Expected Output:   * When the credentials are submitted, a welcome message should open in order to inform the user of a successful login. | Passed: no issues identified |
| 18.6 | Login System | Test that:   * In the case of the user not entering their details into a required field (one or both of Username, Password), make sure the system correctly detects an empty textbox and displays the correct response modal   Do this by:   * Consecutively missing out each of the required (Username and Password) fields in turn and validating that the negative response modal is shown upon submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form (the user will click on the login linklabel if they want to attempt to login again). | Passed: no issues identified |
| 18.7 | Login System | Test that:   * In the case of the user entering an invalid username (not 8 characters), make sure the system correctly detects an invalid input and displays the correct response modal   Do this by:   * Entering all valid credentials other than the username (enter “invalidusername” instead) and validate that the negative response modal is shown upon submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form (the user will click on the login linklabel if they want to attempt to login again). | Passed: no issues identified |
| 18.8 | Login System | Test that:   * In the case of the user entering an invalid password (not 6 characters), make sure the system correctly detects an invalid input and displays the correct response modal   Do this by:   * Entering all valid credentials other than the password (enter “invalidpassword” instead) and validate that the negative response modal is shown upon submission, with the error message “Please enter another username and/or password that meets the given criteria”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form (the user will click on the login linklabel if they want to attempt to login again). | Passed: no issues identified |
| 18.9 | Login System | Test that:   * In the case of the user entering a username and password that does not correspond to an account that already exists in the Garbage Grabberoos Users Database, make sure the system detects this and displays the correct response modal   Do this by:   * Entering all valid credentials that correspond to an account that is not present in the database (“notexist” for username and “noaccs” for password) to validate that the negative response modal is shown upon submission, with the error message “There is no account that matches the username and password that you have entered”   Expected Output:   * Once submitted the error message should be displayed, containing a quit button that reloads the Sign Up form (the user will click on the login linklabel if they want to attempt to login again). | Passed: no issues identified |
| 19.0 | Game Grid System | Test that:   * The user can click the grid to make a successful guess at the location of a sweet wrapper, and this successful guess is signified by a red placeholder and the opening of the sweet wrapper infographic form   Do this by:   * Clicking on the DataGridView until the grid square representing the sweet wrapper is found   Expected Output:   * Once the grid square representing a sweet wrapper is clicked, the grid square should change colour from white to red and the sweet wrapper infographic form should open | Passed: no issues identified |
| 19.1 | Game Grid System | Test that:   * The user can click the grid to make both successful guesses at the locations of the grid squares representing a plastic bottle, and both of these successful guesses will be signified by a blue placeholder and the opening of the plastic bottle infographic form upon both successful guesses   Do this by:   * Clicking on the DataGridView until both of the grid squares representing a plastic bottle are found   Expected Output:   * Once each grid square representing a plastic bottle is clicked, both grid squares should change colour from white to blue and the plastic bottle infographic form should open upon each click | Passed: no issues identified |
| 19.2 | Game Grid System | Test that:   * The user can click the grid to make all three successful guesses at the locations of the grid squares representing a crisp wrapper, and all of these successful guesses will be signified by a green placeholder and the opening of the crisp wrapper infographic form upon each successful guess   Do this by:   * Clicking on the DataGridView until all three of the grid squares representing a crisp wrapper are found   Expected Output:   * Once each grid square representing a crisp wrapper is clicked, all three grid squares should change colour from white to green and the crisp wrapper infographic form should open upon each click | Passed: no issues identified |
| 19.3 | Game Grid System | Test that:   * The user can click the grid to make all four successful guesses at the locations of the grid squares representing a plastic bottle, and all of these successful guesses will be signified by a pink placeholder and the opening of the plastic bottle infographic form upon each successful guess   Do this by:   * Clicking on the DataGridView until all four of the grid squares representing a crisp wrapper are found   Expected Output:   * Once each grid square representing a plastic bottle is clicked, all four grid squares should change colour from white to pink and the plastic bottle infographic form should open upon each click | Passed: no issues identified |
| 19.4 | Game Grid System | Test that:   * The user can click the grid to make all five successful guesses at the locations of the grid squares representing a large car, and all of these successful guesses will be signified by a purple placeholder and the opening of the large car infographic form upon each successful guess   Do this by:   * Clicking on the DataGridView until all five of the grid squares representing a large car are found   Expected Output:   * Once each grid square representing a large car is clicked, all five grid squares should change colour from white to purple and the large car infographic form should open upon each click | Passed: no issues identified |
| 19.5 | Game Grid System | Test that:   * The user can click the grid to successfully find and discard all 15 pieces of rubbish, which will result in information about the user’s winning time (date of win and time taken to win) being written to a file for display in the Speedy Searchers Leaderboard, and the winning message being shown   Do this by:   * Clicking on the DataGridView until all 15 of the grid squares representing a piece of rubbish are found   Expected Output:   * Once all of the grid squares representing a piece of rubbish are successfully found, the winning message form will be displayed to inform the user that they have won. The value in the stopwatch timer that starts at the first click and ends when all 15 pieces of rubbish are found is written to a file called *Winningtimes.txt* along with the date contained in the DateTimePicker on the form. | Passed: no issues identified |
| 19.6 | Game Grid System | Test that:   * The user can click the grid and fail to find all of the pieces of rubbish before their number of guesses reaches 0, causing the losing message to be shown.   Do this by:   * Clicking on the DataGridView until all of the guesses have run out, but not all squares representing a piece of rubbish have been found   Expected Output:   * Once all 70 of the guesses allocated to the user have been used, the losing message form will be displayed to inform the user that they have lost. |  |
| 19.7 | Game Grid System | Test that:   * In the case of the user clicking a grid square that they have already clicked previously, make sure the system correctly detects a repeat click and provides the appropriate modal response   Do this by:   * Clicking again on a DataGridView grid square that has already been clicked   Expected Output:   * An error message should be displayed of “This grid has already been clicked” and the number of guesses made will not be decreased by 1 | Passed: no issues identified |
| 19.8 | Game Grid  System | Test that:   * In the case of the user clicking on the Paused button on the GameGrid form, the game grid will freeze and the paused form will be opened   Do this by:   * Clicking on the Paused button on the GameGrid form   Expected Output:   * The GameGrid form will be disabled so that the user cannot interact with it, and the paused form will be opened while the GameGrid form also remains open |  |
| 19.9 | Game Grid  System | Test that:   * In the case of the user clicking on the Restart button on the GameGrid form, the game grid and all of its controls and labels will reset   Do this by:   * Clicking on the Restart button on the GameGrid form   Expected Output:   * The GameGrid form will be refreshed, and all GameGrid form controls will be reset. The the number of rubbish pieces discarded will be initialised, and the number of guesses made will be reverted to 70 |  |
| 20 | Speedy Searchers Leaderboard System | Test that   * Upon the loading of the speedySearchersBoard form, the data contained in the file *Winningtimes.txt* is written into two arrays of records called date and time. The information contained in the record is displayed in a list box after being bubble sorted by the time array of record from low to high.   Do this by:   * Winning a game of Garbage Grabberoos (see table entry 19.5) then opening the Speedy Searchers Leaderboard page   Expected Output:   * The speedySearchersBoard form when opened should have a listbox displayed that contains the bubble sorted (from lowest to highest time) times of the user’s previous wins alongside the dates that they occurred on. The details of the most recent win should be the most recent entry in the listbox. | Passed: no issues identified |

### **Testing with Persona and Test Cases Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| User Stories | | | |
| 21 | Game Grid Page | *As a young girl who is busy with homework and other extracurriculars, I want to be able to learn more about actions that I can take to prevent climate change in a format that doesn’t feel as rigid and academic as a paper-based task like reading a newspaper, so that I am not overwhelmed by information after a school day, and take it in with a brain that is relaxed and ready to be informed.*  Test that:   * Upon playing the Garbage Grabberoos game, every successful act of discarding a piece of rubbish results in an infographic being displayed about that piece of rubbish. The infographics should contain useful information about that piece of rubbish, and an explanation about why to reduce use of that particular type of rubbish   Do this by:   * Successfully discarding any type of rubbish | Passed: no issues identified |
| 22 | Game Grid Page | *As an avid environmentalist, I want to be able to play a slightly different version of the game each time, (different sequence of trash on the grid) which will allow me to become so accustomed to the act of throwing away different types of trash in different locations by virtual means that it becomes second nature in real life too.*  Test that:   * Upon playing the Garbage Grabberoos game, each different instance of the game has randomised positions of the different types of rubbish on the grid.   Do this by:   * Playing the Garbage Grabberoos game twice, and inspecting whether the positions of all rubbish pieces on the grid are different between the two games | Passed: no issues identified |
| 23 | Speedy Searchers Leaderboard Page | *As a very passionate video game user, I want to be able to view my best times whenever I want, in order to be able to beat them with more practice.*  Test that:   * Upon winning a game of Garbage Grabberoos, the user’s time will be read to a file that can be split into arrays of records which will be displayed in a listbox (bubble sorted from low to high time) in the Speedy Searchers Board form that can be viewed at all times during gameplay.   Do this by:   * Winning a game of Garbage Grabberoos, and ensuring that the achieved time is sorted in its correct place in the list of times from low to high | Passed: no issues identified |
| 24 | Sign Up Page | *As a primary school student who plays several video games, I want to be able to make a fun and creative username to sign up with so I can be identified by it whenever I login.*  Test that:   * On submission of details (username and password) that meet the criteria specified on the form during sign up, these details will be used to create a new entry in the Garbage Grabberoos Users Database, which can be searched during login for user identification.   Do this by:   * Submitting details for username and password that pass all validation during sign up (*funuser1* for username and *‘funpas’* for password), then logging in with those same details. | Passed: no issues identified |

|  |
| --- |
| User Scenario |

|  |  |
| --- | --- |
| 25 | *Sara is an athlete in P7 who plays basketball, hockey and badminton. Playing these sports to a high level has given her something of a competitive personality, which translates to her preference in video game features. Recently, Sara’s sports have finished running as they are all winter games and summer is closely approaching. This has left her without an outlet for her competitive spirit, meaning that she has turned to video games. She opens Garbage Grabberoos because she is feeling particularly bored, and knows that it will allow her to compete against someone else. The random rubbish positions prove to be a challenge to beat, but she manages to win. She feels that the game provided her with an incentive to score highly and finish quickly, just like the sports she is so fond of. She clicks off the game and onto the Speedy Searchers leader board, and relishes in the fact that her success has been acknowledged by the ranking system of times, and feels that this is a good substitute for the feeling of coming first in real life.*  Test that:   * The user can successfully create a new account and login with the created details, play Garbage Grabberoos and view the speedy searchers leaderboard   Do this by:   * Creating a new account through the Sign Up page by entering the following credentials “*Sara1234*” for username and “*hockey*” for password. Then login by entering “*Sara1234*” for username and “*hockey*” for password. Consequently, click on START GAME when brought to the game menu, read the instructions, then begin the game. Win the game by finding all of the pieces of rubbish and view the new time associated with this win in the Speedy Searchers Leaderboard |

### **End User Testing Plan**

|  |
| --- |
| Descriptions of several scenarios for other end-users, who will carry them out before answering a survey |

|  |  |  |  |
| --- | --- | --- | --- |
| 26 | User 1 | Scenario:  User 1 will be tasked with creating a new account from scratch and then logging in – with the details chosen by the user. | Completed |
| 27 | User 2 | Scenario:  User 2 will be tasked with winning the Garbage Grabberoos game. They must then view their winning time in Speedy Searchers Leaderboard form. | Completed |
| 28 | User 3 | Scenario:  User 3 will be tasked with losing the Garbage Grabberoos game. They must then repeat the game until they win. | Completed |

## Completed Final Testing

### Interface Testing Completed

|  |  |
| --- | --- |
| Sign Up Page | |
| The Sign Up should display all wireframe specified graphics, including the user interactive controls of:  Textboxes  Sign Up button,  Login linklabel,  Quit button | A screenshot showing the correct placement and content of each element of the Sign Up page according to the wireframes:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sign Up form contains two textboxes – one for username and one for password * The Sign Up form contains a SIGN UP button which opens the Successful Sign Up form when clicked if validation is passed and opens one of 2 error messages when clicked if validation is not is passed * The Sign Up form contains a login linklabel which closes the Sign Up form and opens the Login form when clicked * The Sign Up form contains a Quit button that ends the entire program if clicked * The elements listed above match the Sign Up form wireframes correctly |
| Login Page | |
| The Login Page should display all wireframe specified graphics, including the user interactive controls of:  Textboxes  Login button,  Sign Up linklabel,  Quit button | A screenshot showing the correct placement and content of each element of the Login page according to the wireframes:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Login form contains two textboxes – one for username and one for password * The Login form contains a LOG IN button which opens the welcome message when clicked if validation is passed and opens one of 2 error messages when clicked if validation is not is passed * The Login form contains a Sign up linklabel which closes the Login form and opens the Sign Up form when clicked * The Login form contains a Quit button that ends the entire program if clicked * The elements listed above match the Login form wireframes correctly |
| Game Menu Page | |
| The Game Menu should display all wireframe specified graphics, including the user interactive controls of:  Start Game Button,  View Leaderboard Button,  Quit Game Button | A screenshot showing the correct placement and content of each element of the Game Menu page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The game menu page contains a START GAME button that closes the gameMenu form and opens the howToPlay form if clicked * The game menu page contains a VIEW LEADERBOARD button that closes the gameMenu form and opens the speedySearchersBoard form if clicked * The game menu page contains a QUIT GAME button that ends the entire program if clicked * The elements listed above match the Game Menu form wireframe correctly |

|  |  |
| --- | --- |
| How to Play Page | |
| The Game Menu Page should display all wireframe specified graphics, including the user interactive controls of:  Start Game Button,  Quit Game Button | Screenshots showing the correct placement and content of each element of the Game Menu page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The game menu page contains a START GAME button that closes the gameMenu form and opens the howToPlay form if clicked * The game menu page contains a VIEW LEADERBOARD button that closes the gameMenu form and opens the speedySearchersBoard form if clicked * The game menu page contains a QUIT GAME button that ends the entire program if clicked * The elements listed above match the Game Menu form wireframe correctly |
| Game Grid Page | |
| The Game Grid Page should display all wireframe specified graphics, including the controls of:  DataGridView (changing colour grid squares in presence of different rubbish types),  Pause Button,  Restart Button,  Quit Button,  Help Button,  Menu Button | A screenshot showing the correct placement and content of each element of the Game Grid page according to its wireframes:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The game grid form contains a 10 by 11 grid of squares formed from a 2D array * The game grid page contains a Pause button which will open the paused form and disable the game grid if clicked * The game grid page contains a Restart button which will reset all form controls, initialise the number of rubbish pieces discarded, and revert number of guesses to 70 if clicked * The game grid page contains a HELP button which will open the Instructions form while keeping the GameGrid form open if clicked * The game grid page contains a Quit button which will close the entire program if clicked * The clicking of a grid square on the grid that contains no rubbish will keep the square colour as white, but notify the user to know that they have just clicked there because of a temporary navy square   A picture containing calendar  Description automatically generated   * The clicking of a grid square on the grid representing a sweet wrapper will change the square colour from white to red   A picture containing timeline  Description automatically generated   * The clicking of a grid square on the grid representing a plastic straw will change the square colour from white to blue   Chart  Description automatically generated with low confidence   * The clicking of a grid square on the grid representing a crisp packet will change the square colour from white to green   Chart  Description automatically generated with medium confidence   * The clicking of a grid square on the grid representing a plastic bottle will change the square colour from white to pink   Chart  Description automatically generated   * The clicking of a grid square on the grid representing a plastic bottle will change the square colour from white to purple   A picture containing scatter chart  Description automatically generated |
| Speedy Searchers Leaderboard Page | |
| The Speedy Searchers Leaderboard Page should display all wireframe specified graphics, including the user interactive controls of:  Menu Button,  Quit Button,  ‘Go back to grid’ Linklabel | A screenshot showing the correct placement and content of each element of the Speedy Searchers Leaderboard page according to its wireframe:  Graphical user interface, table  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Speedy Searchers Leaderboard page contains a Menu button which will close the speedySearchersBoard form and open the gameMenu form if clicked * The Speedy Searchers Leaderboard page contains a Quit button which will close the entire program if clicked * The Speedy Searchers Leaderboard page contains a ‘Go back to grid’ linklabel which will close the speedySearchersBoard form and open the GameGrid form if clicked |
| Unsuccessful SignUp or Login Page | |
| Unsuccessful SignUp or Login Page should display all wireframe specified graphics, including the user interactive controls of:  Sign Up Button  Login Button  Quit Button | A screenshot showing the correct placement and content of each element of the Unsuccessful SignUp or Login page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Unsuccessful SignUp or Login Page contains a quit button which will close the unsuccessfulSignUpOrLogin form if clicked * The Unsuccessful SignUp or Login Page contains a Sign Up button which will close the unsuccessfulSignUpOrLogin form and open the SignUp form if clicked * The Unsuccessful SignUp or Login Page contains a Login button which will close the unsuccessfulSignUpOrLogin form and open the Login form if clicked * The elements listed above match the Unsuccessful SignUp or Login form wireframe correctly |
| Account Existing Page | |
| Account Existing Page should display all wireframe specified graphics, including the user interactive controls of:  Quit Button,  Login linklabel | A screenshot showing the correct placement and content of each element of the Account Existing page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Account Existing Page contains a quit button which will close the noAccountFound form and open the Login form if clicked * The Account Existing Page contains a Login linklabel which will close the noAccountFound form and open the Login form if clicked |
| Successful Sign Up Page | |
| Successful Sign Up Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Successful Sign Up Page according to its wireframe:  Graphical user interface, text, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Successful Sign Up Page contains a quit button which will close the successfulSignUp form and open the Login form if clicked |
| Successful Sign Up Page | |
| Successful Sign Up Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Successful Sign Up Page according to its wireframe:  Graphical user interface, text, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Successful Sign Up Page contains a quit button which will close the successfulSignUp form and open the Login form if clicked |
| No Account Found Page | |
| No Account Found Page should display all wireframe specified graphics, including the user interactive controls of:  Quit Button,  ‘Sign up for an account’ linklabel | A screenshot showing the correct placement and content of each element of the No Account Found Page according to its wireframe:  Graphical user interface, text, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The No Account Found Page contains a quit button which will close the noAccountFound form and open the Login form if clicked * The No Account Found Page contains a ‘Sign up for an account’ linklabel which will close the noAccountFound form and open the SignUp form if clicked |
| Instructions Page | |
| Instructions Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Instructions Page according to its wireframe:  Graphical user interface, text, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Instructions Page contains a quit button which will close the instructions form and keep the GameGrid form open if clicked |
| Paused Page | |
| Paused Page should display all wireframe specified graphics, including the user interactive control of:  Continue Game Button | A screenshot showing the correct placement and content of each element of the Paused Page according to its wireframe:  Graphical user interface, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Instructions Page contains a Continue Game button which will re-enable the GameGrid form |
| Losing Page | |
| Losing Page should display all wireframe specified graphics, including the user interactive control of:  Restart Button,  View Leaderboard Button | A screenshot showing the correct placement and content of each element of the Losing Page according to its wireframe:  Graphical user interface, application, website  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Losing Page contains a Restart Game button which will close the losingForm and open the GameGrid form where all form controls will be reset, the number of rubbish pieces discarded will be initialised, and the number of guesses to 70 will be reverted if clicked * The Losing Page contains a View Leaderboard button which will close the losingForm and open the speedySearchersBoard form |
| Winning Page | |
| Winning Page should display all wireframe specified graphics, including the user interactive control of:  Restart Button,  View Leaderboard Button | A screenshot showing the correct placement and content of each element of the Winning Page according to its wireframe:  Graphical user interface, application  Description automatically generated  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Winning Page contains a Restart Game button which will close the winningForm and open the GameGrid form where all form controls will be reset, the number of rubbish pieces discarded will be initialised, and the number of guesses to 70 will be reverted if clicked * The Winning Page contains a View Leaderboard button which will close the winningForm and open the speedySearchersBoard form |
|  |  |
| Sweet Wrapper Infographic Page | |
| Sweet Wrapper Infographic Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Sweet Wrapper Infographic Page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Sweet Wrapper Infographic Page contains a Quit button which will close the sweetWrapper form while keeping the GameGrid form open |
| Plastic Straw Infographic Page | |
| Plastic Straw Infographic Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Plastic Straw Infographic Page according to its wireframe:    Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Plastic Straw Infographic Page contains a Quit button which will close the plasticStraw form while keeping the GameGrid form open |
| Crisp Packet Infographic Page | |
| Crisp Packet Infographic Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Crisp Packet Infographic Page according to its wireframe:  **Graphical user interface, application  Description automatically generated**  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Crisp Packet Infographic Page contains a Quit button which will close the crispPacket form while keeping the GameGrid form open |
| Plastic Bottle Infographic Page | |
| Plastic Bottle Infographic Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Plastic Bottle Infographic Page according to its wireframe:  **Graphical user interface, application  Description automatically generated**  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Plastic Bottle Infographic Page contains a Quit button which will close the plasticBottle form while keeping the GameGrid form open |
| Large Car Infographic Page | |
| Large Car Infographic Page should display all wireframe specified graphics, including the user interactive control of:  Quit Button | A screenshot showing the correct placement and content of each element of the Large Car Infographic Page according to its wireframe:  **Text  Description automatically generated with medium confidence**  Interface testing plan successfully achieved as:   * All form elements load correctly with no content errors (spelling mistakes, missing buttons etc.) * The Large Car Infographic Page contains a Quit button which will close the largeCar form while keeping the GameGrid form open |

### Final and Requirements Testing Completed

|  |  |
| --- | --- |
| Test Case | Description and Results |
| 18.0 | Successful: The user can enter their credentials (that pass all validation) into the specified Username and Password fields. The data is used to create a new entry in the Garbage Grabberoos Database correctly, and a successful sign up message is shown.    Evidence that the system works correctly:    A screenshot of a computer  Description automatically generated with low confidence |
| 18.1 | Successful: If the user fails to put information in either or both textboxes, then the unsuccessful sign up or login message is shown. |
| 18.2 | Successful: The username text field input needs to pass the specified validation (8 characters for username) or the unsuccessful sign up or login error message will be shown, even if the password is the correct length of 6 characters. |
| 18.3 | Successful: The password text field input needs to pass the specified validation (6 characters for password) or the unsuccessful sign up or login error message will be shown, even if the username is the correct length of 8 characters.  Graphical user interface  Description automatically generated |
| 18.4 | Successful: In the case of the user signing up by entering a username and password that pass validation but correspond to an account that already exists in the Garbage Grabberoos Users Database, the specific account existing error message will be shown.  Table  Description automatically generated  *(Evidence that the entered details already correspond to an account in the Garbage Grabberoos Users database)*  Graphical user interface  Description automatically generated with medium confidence |
| 18.5 | Successful: The user can enter their credentials (that pass all validation) into the specified Username and Password fields. The data is used to search the Garbage Grabberoos Users Database for the corresponding account, the username of which is used in the welcome message that is shown upon successful login. |
| 18.6 | Successful: If the user fails to put information in either or both textboxes, then the unsuccessful sign up or login message is shown. |
| 18.7 | Successful: The username text field input needs to pass the specified validation (8 characters for username) or the unsuccessful sign up or login error message will be shown, even if the password is the correct length of 6 characters. |
| 18.8 | Successful: The password text field input needs to pass the specified validation (6 characters for password) or the unsuccessful sign up or login error message will be shown, even if the username is the correct length of 8 characters.  Graphical user interface  Description automatically generated |
| 18.9 | Successful: In the case of the user logging in by entering a username and password that pass validation but do not correspond to an account that already exists in the Garbage Grabberoos Users Database, the specific no account found error message will be shown.  *Evidence that an account linked to the entered details (“notexist” for Username and “noaccs” for Password) does not exist in the Garbage Grabberoos Users database:* |
| 19.0 | Successful: The user can click the grid to make a successful guess at the location of a sweet wrapper, and this successful guess is signified by a red placeholder and the opening of the sweet wrapper infographic form |
| 19.1 | Successful: The user can click the grid to make a successful guess at the location of a plastic straw, and this successful guess is signified by a blue placeholder and the opening of the plastic straw infographic form  Chart  Description automatically generated with low confidence |
| 19.2 | Successful: The user can click the grid to make a successful guess at the location of a crisp packet, and this successful guess is signified by a green placeholder and the opening of the crisp packet infographic form  Chart  Description automatically generated with low confidence  Graphical user interface, application  Description automatically generated |
| 19.3 | Successful: The user can click the grid to make a successful guess at the location of a plastic bottle, and this successful guess is signified by a pink placeholder and the opening of the plastic bottle infographic form  Chart  Description automatically generated  Graphical user interface, application  Description automatically generated |
| 19.4 | Successful: The user can click the grid to make a successful guess at the location of a large car, and this successful guess is signified by a purple placeholder and the opening of the large car infographic form  Text  Description automatically generated with medium confidence  A picture containing chart  Description automatically generated |
| 19.5 | Successful: If the user successfully discards all 15 pieces of rubbish before their number of guesses left reaches 0 then the success message will be shown, and the time will be added to the WinningTimes.txt file.  Graphical user interface, application  Description automatically generated |
| 19.6 | Successful: If the user fails to find all of the pieces of rubbish before their number of guesses reaches 0, then the losing message will be shown.  Graphical user interface, application, website  Description automatically generated |
| 19.7 | Successful: If the user clicks on a grid square that they have already previously clicked on, the “You already hit that cell” error message will be shown.  A picture containing timeline  Description automatically generated |
| 19.8 | Successful: In the case of the user clicking on the Paused button on the GameGrid form, the game grid will freeze and the paused form will be opened  Graphical user interface, application  Description automatically generated |
| 19.9 | In the case of the user clicking on the Restart button on the GameGrid form, the game grid and all of its controls and labels will reset  Calendar  Description automatically generated |
| 20 | Successful: Upon winning a game of Garbage Grabberoos (shown in table section 19.5) then loading the speedySearchersBoard form, the data contained in the file *Winningtimes.txt* is written into two arrays of records called date and time. The information contained in the record is displayed in a list box after being bubble sorted by the time array of record from low to high, and the data associated with the most recent win is sorted into the list appropriately.  Graphical user interface, table  Description automatically generated |

### Testing with Persona and Test Cases Completed

|  |  |
| --- | --- |
| 21 | *As a young girl who is busy with homework and other extracurriculars, I want to be able to learn more about actions that I can take to prevent climate change in a format that doesn’t feel as rigid and academic as a paper-based task like reading a newspaper, so that I am not overwhelmed by information after a school day, and take it in with a brain that is relaxed and ready to be informed.*  Successful: Upon playing the Garbage Grabberoos game, every successful act of discarding a piece of rubbish results in an infographic being displayed about that piece of rubbish. The infographics should contain useful information about that piece of rubbish, and an explanation about why to reduce use of that particular type of rubbish  *Example discarding of sweet wrapper shown below, alongside the matching infographic* |
| 22 | *As an avid environmentalist, I want to be able to play a slightly different version of the game each time, (different sequence of trash on the grid) which will allow me to become so accustomed to the act of throwing away different types of trash in different locations by virtual means that it becomes second nature in real life too.*  Successful: Upon playing the Garbage Grabberoos game, each different instance of the game has randomised positions of the different types of rubbish on the grid.  *Instance 1*    *Instance 2* |
| 23 | *As a very passionate video game user, I want to be able to view my best times whenever I want, in order to be able to beat them with more practice.*  Successful: Upon winning a game of Garbage Grabberoos, the user’s time will be read to a file that can be split into arrays of records which will be displayed in a listbox (bubble sorted from low to high time) in the Speedy Searchers Board form that can be viewed at all times during gameplay.  *\*2 second disparity between screenshot of end time and click of final square*  A picture containing calendar  Description automatically generated  Graphical user interface, application, Word  Description automatically generated  Table  Description automatically generated |
| 24 | *As a primary school student who plays several video games, I want to be able to make a fun and creative username to sign up with so I can be identified by it whenever I login.*  Successful: The user is able to make a creative 8 letter username and 6 letter password of their choice at sign up, then log in to be associated with them (for example: referred to by username in the welcome message). |

|  |
| --- |
| User Scenario |

|  |  |
| --- | --- |
| 25 | *Sara is an athlete in P7 who plays basketball, hockey and badminton. Playing these sports to a high level has given her something of a competitive personality, which translates to her preference in video game features. Recently, Sara’s sports have finished running as they are all winter games and summer is closely approaching. This has left her without an outlet for her competitive spirit, meaning that she has turned to video games. She opens Garbage Grabberoos because she is feeling particularly bored, and knows that it will allow her to compete against someone else. The random rubbish positions prove to be a challenge to beat, but she manages to win. She feels that the game provided her with an incentive to score highly and finish quickly, just like the sports she is so fond of. She clicks off the game and onto the Speedy Searchers leader board, and relishes in the fact that her success has been acknowledged by the ranking system of times, and feels that this is a good substitute for the feeling of coming first in real life.*  At sign up, Sara initially doesn’t notice the username and password specifications since she is hurrying to make an account, eager to play Garbage Grabberoos as her friends have told her about it. She attempts to create an account with a username and password that both fail validation as they are too long (username is not 8 characters and password is not 6 characters).    This input is met with an error message, which prompts Sara to re-read the sign up form and notice the username and password criterias.    She clicks on the SIGN UP button on the error message, and tries again. She makes the 8 letter username her name followed by numbers *(“sara1234”),* and chooses something that she loves to be the 6 letter password *(“hockey”)* because she will knows that she will remember it at login.    She is presented with a successful sign up message informing her that her chosen details have passed all validation so the account has been created successfully, and closes the message in order to login.    She logins with her created details, and as expected, was able to remember them with ease.    She is greeted with a welcome message, which she enjoyed as she felt it served as a direct acknowledgement of her playing the game.    After closing the welcome message she is brought to the game menu, where she reads each of the options before clicking on START GAME.    She is met with instructions on how to play the game, which she comprehends relatively quickly as she has played some games with a similar premise (finding a hidden object on a grid) in the past. She clicks on the START GAME button.    An empty grid opens on her screen.    She makes three tentative clicks to get a feel of the game.    On her fifth click, she successfully finds one of the rubbish pieces: a crisp packet. She reads all of the information contained on the infographic and feels that she will now be much more conscious about disposing of her crisp packets after eating them at school lunchtime.    She clicks below then above this grid square and manages to find the second crisp packet grid.    Her next find is the large car grid square, and she finds the other four grid squares that represent a large car by clicking below the first large car grid square.    While searching for more rubbish, Sara mistakenly clicks on a grid square that she has clicked on before. She is met with this error message, and feels glad that this blip does not count as a guess.      She next finds the two squares that represent a plastic straw.    She begins to worry when she notices that she only has three guesses left.    She loses the game once she has used up all of her guesses, and is given this message.    It inspires her to click the RESTART GAME button to try again, while she vows that she will win the next game and be able to insert a winning entry into her currently empty Speedy Searchers Leaderboard. |

### End User Testing Completed

|  |
| --- |
| Results of several scenarios for other end-users, who carried them out before answering a survey |

|  |  |  |
| --- | --- | --- |
| 26 | User 1 | Successful: All elements of the scenario were carried out successfully and no errors found.  *Account creation and subsequent login by User 1:* |
| 27 | User 2 | Successful: All elements of the scenario were carried out successfully and no errors found.  *Win and subsequent Speedy Searchers Leaderboard viewing by User 2:*  *\*2 second disparity between screenshot of end time and click of final square* |
| 28 | User 3 | Successful: All elements of the scenario were carried out successfully and a minor error was found then resolved.  *Loss then subsequent restart of Garbage Grabberoos game:*        From this end user test, I identified an error: that the losing message does not close when the RESTART button is clicked and a new game grid is opened, so I modified the code to close the losing message when the user clicks on the RESTART button.  *Previous Code:*    *Modified code:* |

# EVALUATION

# Fitness for Purpose

The results of testing demonstrate that the final solution successfully meets all of the functional and end user requirements detailed at the start of development during the analysis stage (see Requirements Specification).

The project meets end user requirements because while playing Garbage Grabberoos the user will be:

* Having their winning score times stored within the game
* Interacting with a user interface that has a visually pleasing colour scheme and is easy to navigate
* Interacting with a user interface that is gender neutral and can appeal to everybody
* Learning how to recycle effectively, and know which pieces of rubbish to look out for
* Practicing recycling virtually in order to replicate it more often in real life
* Learning more about climate change and the role that certain products have in catalysing it
* Playing a game with a short duration so that they can fit into their daily lives
* Playing a game that incorporates the topics of recycling, sustainability and vegetarianism
* Playing a game that poses a standard level of difficulty
* Learning more facts about the environment and specific rubbish pieces

The project meets functional requirements because the Garbage Grabberoos System will:

* Be able to connect to the Garbage Grabberoos User Database
* Allow for a user to make an account that will update the Garbage Grabberoos User Database by adding their username and password with an INSERT INTO query
* Check the Garbage Grabberoos User Database for an existing account that corresponds to the user’s username and password during the sign up process with a SELECT query
* Validate the user’s entry for username and password upon both sign up and login
* Allow for a user to log on using the username and password associated with their created account by checking the Garbage Grabberoos User Database using a SELECT query
* Contain a grid (made up of an 11 by 10 2D array) that can be clicked on by the user
* List user winning scores sorted from low to high in a leader board that can be visited throughout gameplay
* Allow for the grid squares that have been clicked on to be remembered, so the user cannot use two guesses on the same square and will instead be displayed an error message if this happens
* Time each game in order to be able to write this to a file and then add this time to the time array of records if the user wins the game
* Contain forms that act as infographics and educate users on the rubbish they pick up during gameplay
* Contain buttons that will allow the user to pause and restart the game

# Results of Testing

### Log of Ongoing Testing

The majority of errors and bugs in my program were identified during implementation through my log of ongoing testing. Firstly, I had minor issues with the user interface of my program, where I discovered that I could not use Visual Basic powerpack shapes as they were not compatible with all versions of VisualBasic, and so stopped me from working on my project from home. I added a panel instead upon research for how implement a border in Visual Basic – consulting websites like StackOverflow and YouTube. I next had issues with the input validation for my sign up and login pages as the error messages were not formatting properly and the form controls of these messages became white and frozen upon display. I resolved this by implementing a Timer function, which took several days of research before I found a viable solution and consideration of functions like Sleep (See Research and Development of New Skills for more information). One of my most major issues was with the Speedy Searchers Leaderboard. I first had problems with the parsing of the record – where only a few values were being successfully shown in the listbox. I had to alter the loop that was displaying values in the listbox to solve this. I then had an error produced by an extra line in the file containing game win information, and implemented a feature in the game to automatically ignore this line once I realised that I would otherwise have to remove it myself upon each new entry into the file. Lastly, I had issues with the passing of variables between the functions that validated username and password in order to achieve all of the necessary types of login and sign up input validation. I had to alter my parameter passing to resolve this. In order to make the formal and actual parameters of various functions match, I reduced the number of variables I had created that were all performing the same function, and used the minimal number of these variables to achieve a successful input validation system, and avoid confusion as to which variables contained the values that I required.

### Interface Testing

I performed interface testing on my program to make sure that there were no visual disparities (in terms of graphics, text or layout) between each form page and its corresponding wireframe. No issues were identified as a result of this testing.

### Final and Requirements Testing – coupled with Integration Testing

For final and requirements testing, I tested the program to ensure that all functional and end user requirements were met. This meant a test of the full program: sign up, login, gameplay and the leaderboard display. This test also served as integration testing since one of the test cases performed was an entry (INSERT INTO) of Username and Password into the Garbage Grabberoos Users Database via the Garbage Grabberoos game sign up system, and a subsequent search (SELECT) of the Garbage Grabberoos Users Database for an account corresponding to these details via the Garbage Grabberoos game login system representing the integration between software design and development and database design and development. All of the test cases passed (including the one performed for integration testing) and there were no errors identified as a result of this testing. The main conclusion of this testing is that the final solution successfully meets all of the functional and end user requirements detailed at the start of development during the analysis stage, and incorporates an integration between a software development based program and a database with at least one table (the users table in the Garbage Grabberoos Users Database).

### Testing with Personas, User Stories and Test cases

For this test, I designed test cases for each of the user stories and the user scenario that I had created at the start of the development process during the analysis stage (See User Stories and User Scenario). All of the test cases passed, so there were no errors identified as a result of this testing.

### End User Testing

For this test, I asked three end users to use the program to act out three scenarios that I created for the purpose of observing the users interacting with the program, and to expose any final discreet errors. All of the end users completed their task successfully, and one end user (User 3) enabled me to uncover an error that had gone unnoticed till then. The losing form was remaining open when the user clicked the RESTART button to access a fresh game grid, so I modified the code to prevent this from happening (See End User Testing Completed)

## Future Maintainability

Overall, my program code is very maintainable. I have included detailed and plentiful comments, in order to explain the purpose of sections of code. I have used meaningful identifiers for variables, arrays, procedures and functions so that their purpose is clear. I have used indentation and white space to make the start and end of control structures and subprograms more easily identifiable. Additionally, the code is modular, as I have split the code into appropriate procedures and functions with relevant names that demonstrate their content, which will make it easier to focus on the relevant part of code when performing future maintenance.

In order to further improve maintenance, I could increase the white space used internally within the sub procedures and functions in the program code as I feel that I only used utilised white space at the start and end of control structures. I also could have used indentation more frequently, as I used it sparingly – mainly within loops to place emphasis on inner and outer parts.

To conclude, I believe my code for Garbage Grabberoos is well-designed for maintainability, which will make it easier to maintain and update in the future.

## Robustness

I believe that my program code is reasonably robust. Firstly, it generates many different types of error message (which cover almost every type of scenario) during sign up and login. If the user enters a username and/or password that are not of the correct length, fails to enter an input into one or both textboxes, enters details that correspond to an account already in the Garbage Grabberoos Database during sign up, or enters details that do not correspond to an account already in the Garbage Grabberoos Database during login, then an appropriate error message will be shown. There is also validation applied to game grid, where the user will be unable to click on the same grid square twice without being shown an error message.

However, my robustness is lacking due to some assumptions that I have made during implementation. I have assumed that users will not click on the grid squares used to represent alphabet and number label tags, so there is no validation applied for if they do this. In order to further improve robustness, I would apply validation that prevents them from clicking on these grid squares without being shown an error message.

## Usability

The Garbage Grabberoos game is very usable. I achieved this in my program code by providing clear user prompts and instructions before gameplay through implementation of a ‘How to Play form’, and by including a help screen that can be accessed throughout gameplay. I also created a clear and uncluttered user interface on every form that can be easily navigated.

To improve usability, I think that I could make the criteria for the username and password during sign up clearer to the user by increasing the size of the label text and making it bold, to prevent users from overlooking it during sign up and struggling to create an account.