Brain Games - Final Documentation

By Arnav Badrish

Documentation Included:

* Project Management Report
* Project procedure report
* Miscellaneous Notes

Contents

# Part 1 – Project Management Report…………………………3

* Definition of the problem……………………………………………3
* Original Ideas and Extra Features………………………………..4
* Intended Audience/Demographic………………………………..5
* Screen Design Sketches and notes on screen design……5
* Project development timeline……………………………………10
* Gantt Chart of project timeframe………………………………12

# Part 2 – Project Procedure Report…………………………….13

* Project development and approach…………………………..13
* Final Solution and reasoning………………………………………13
* Problems encountered……………………………………………….16

# Part 3 – Miscellaneous Notes……………………………………17

* Usage of Intrinsic and Internal Documentation………….17

Part 1

Project Management report

# Definition of the Problem

The problem presented to us, in the exact terms, were:

*You are to design your own programme using Visual Studio. The Programme is to be a game of Boggle (or Yahtzee, Memory Puzzle)*

Using this problem statement, the problem was broken down into a list of requirements that are to be fulfilled by the final product, which are stated below:

* The program needs to be a game of Boggle, Yahtzee or Memory Puzzle
* The program needs to run on Visual Studio
* The program must use some combination of
  + calculations
  + case statements
  + random number generation
  + timer
  + simple and effective graphical design
* The program must have adequate documentation to go with it

However, upon further analysis of the time frame and the capabilities of the language as well as my capabilities, small amendments were made to improve the effectiveness of the final solutions and to improve user experience:

* Must make use of sound effects (SFX)
* Must have customizable backgrounds.
* Must be able to store and load user preferences, as well as user high scores

After further analysis, I concluded that I had come up with a suitable list of requirements, as well as the steps needed to satisfy the problem. The final version of Brain Games has satisfied all the proposed requirements.

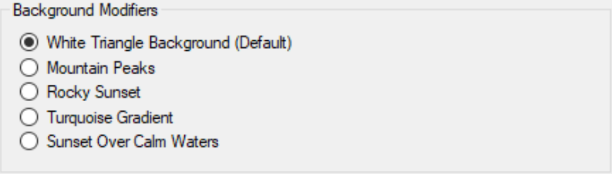
# Original Ideas and Extra Features

There were a few extra features that were implemented into my version of Brain Games to improve user experience and make my version of the game stand apart from the other versions of the game. These features are not explicitly stated in the requirements statement given to us.

* **New Game Mode : Word Blitz**

This is a completely unique game that I had devised and coded and put into my version of Brain Games. The concept was intended to be simple: a sequence of words taken from a word bank of 80 words are flashed onto the screen in a random sequence. Then, a series of textboxes are displayed on the screen. The user’s job is to then type the presented words into the textboxes given in the correct order that they were flashed on the screen as quick as they can. Then, the user is to click a button after they have finished. If the user did not input the words in the correct order, the correct words will be displayed, along with ticks and cross next to them so the user can see what words they got right, and what they got wrong.

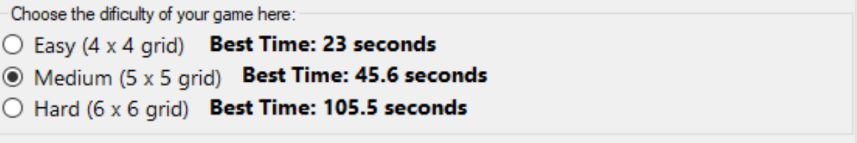
* **Customizable Backgrounds**

The game also allows the user to change the background of the main forms within the application. The user can choose from five different backgrounds. This element allows users to customise their experience

* **Best time functionality**

While not stated explicitly within the problem definition, I have implemented a best time feature within the game to keep a best time to add a sense of competition to the game, while also contributing to the replay value of the game

* **Difficulty Modifier**

Furthermore, I also implemented a difficulty feature for both Word Blitz and Image match to further increase replay value and also to allow the user to further customise their experience.

* **Sound Effects and soundtracks**

While this feature is not exceedingly special, it is a vital tool in creating the ideal user experience, and in creating an immersive game environment. Furthermore, it was not explicitly stated in the problem definition, so it is technically an extra feature.

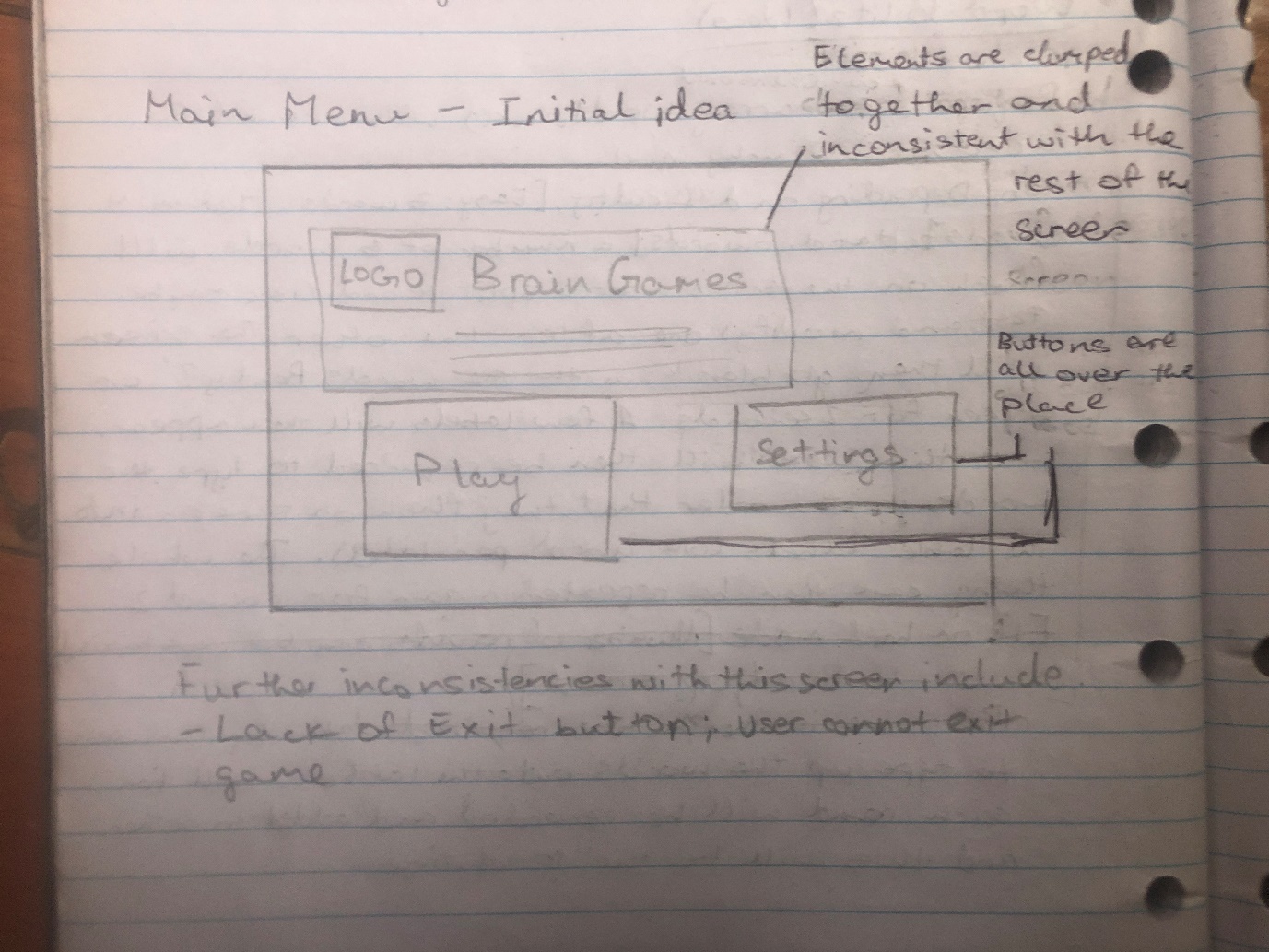
# Target Audience

The final version of Brain Games was not designed with a specific target audience in mind. Instead, it was designed to be played and enjoyed by everyone. As such, the user interface was designed so that it may be operated with ease by everyone, even users who may not excel at computing. Elements such as the music and the user interface were designed so that all users may enjoy it. The user instructions are not targeted to any demographic in particular. Likewise, my program is designed to be simple, but not exceedingly simple so as to only be targeted towards very young users.

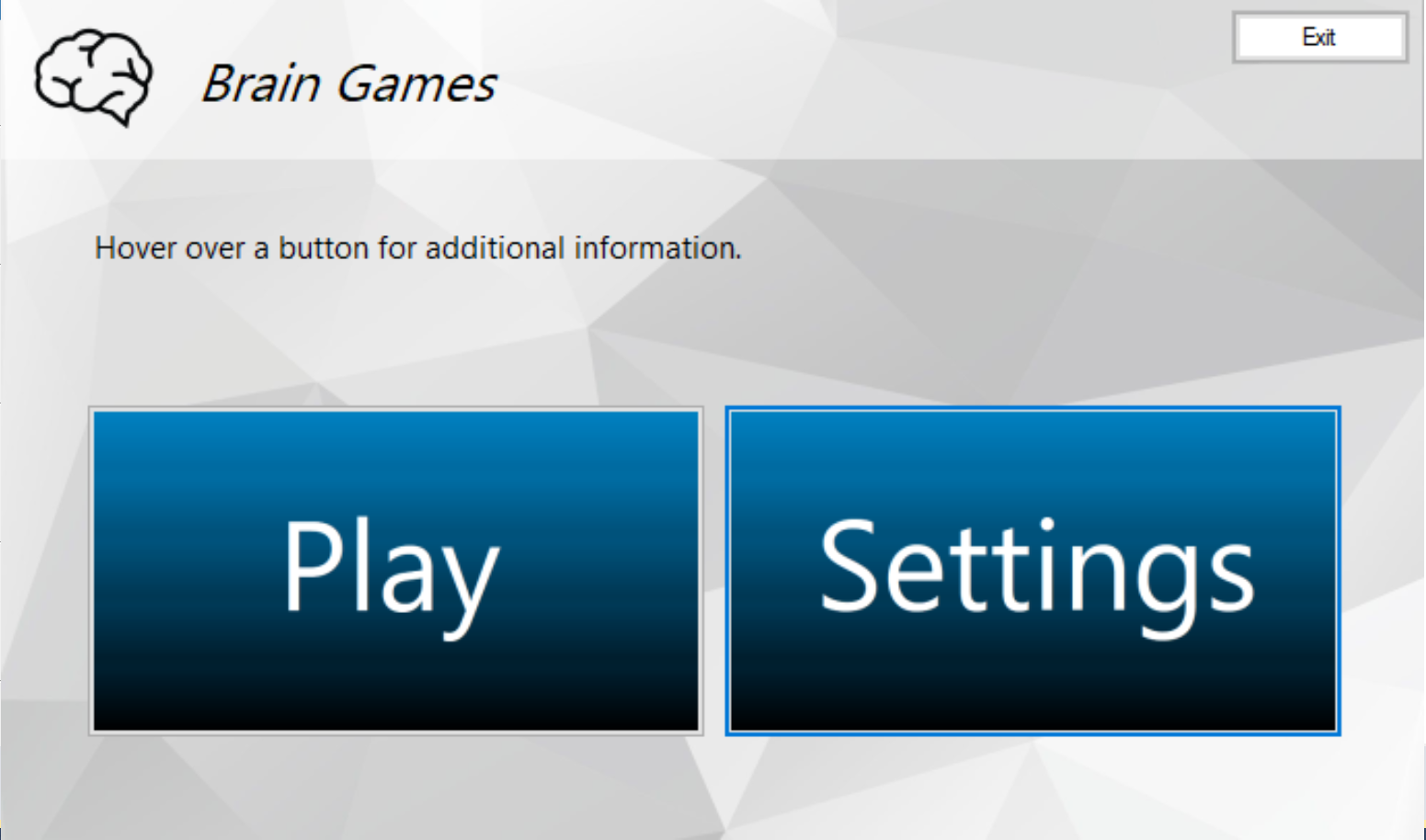
# Screen design sketches

Below are some sketches of the original screen design of the important elements of the game, as well as the final version of said elements. They are annotated to explain changes and improvements.

**Main Menu Initial sketch (Hand-drawn) – On Next Page**



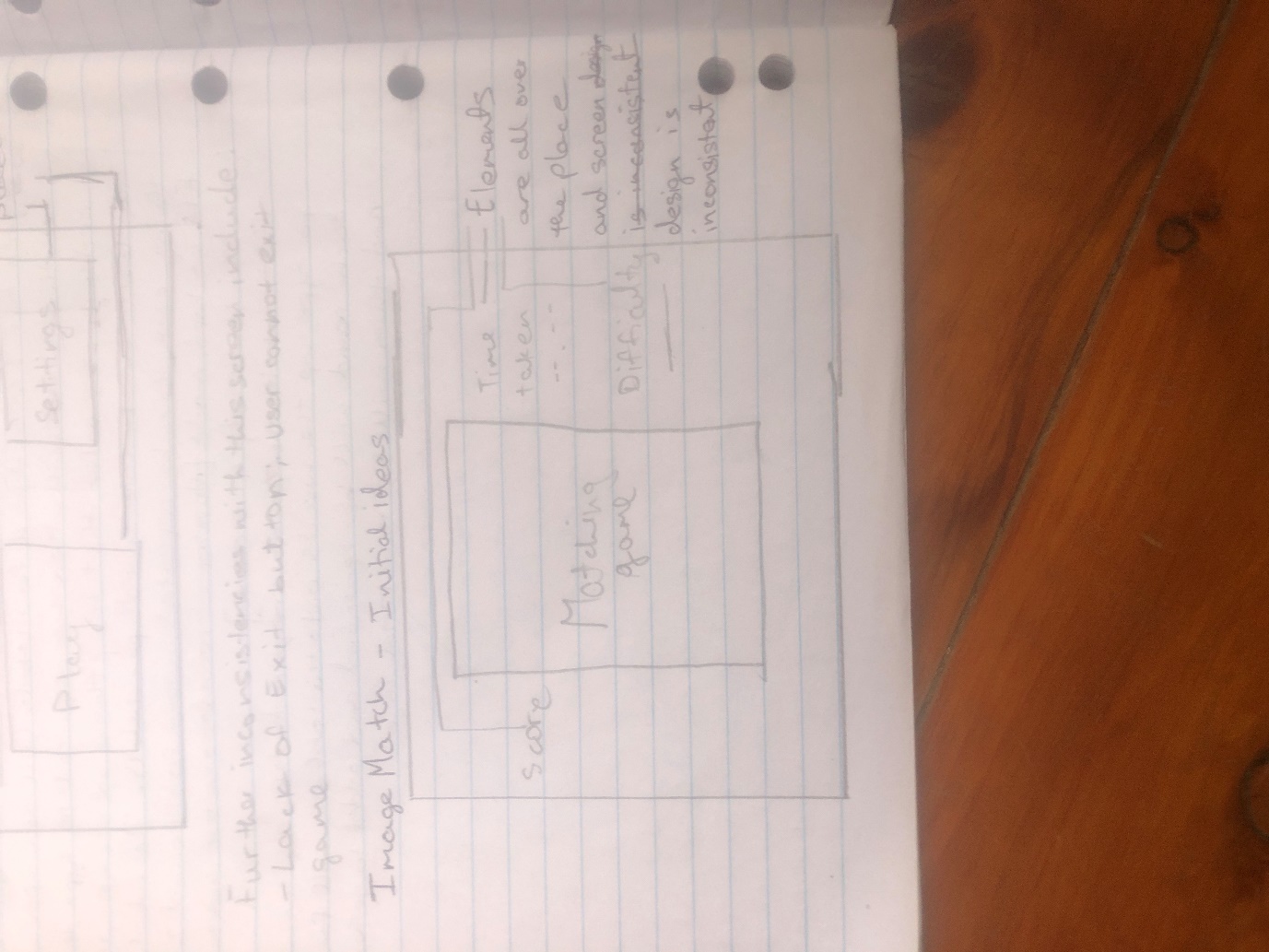
**Main Menu Final Design**



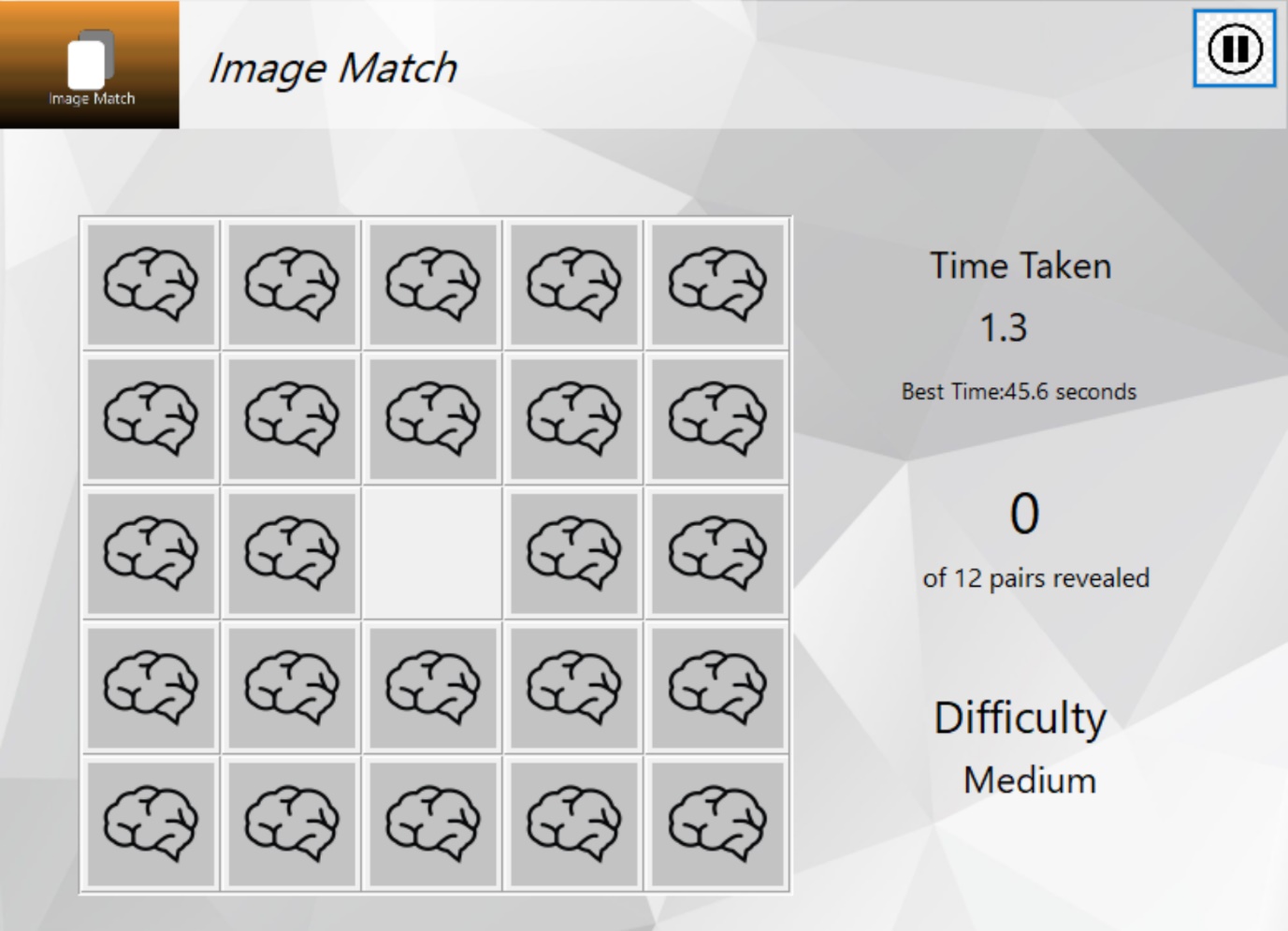
Buttons are grouped together and are also decently sized.

Use of a header to add some consistency to the screen design. Exit button is now also included

**Image Match Initial Sketch (Hand-Drawn)**



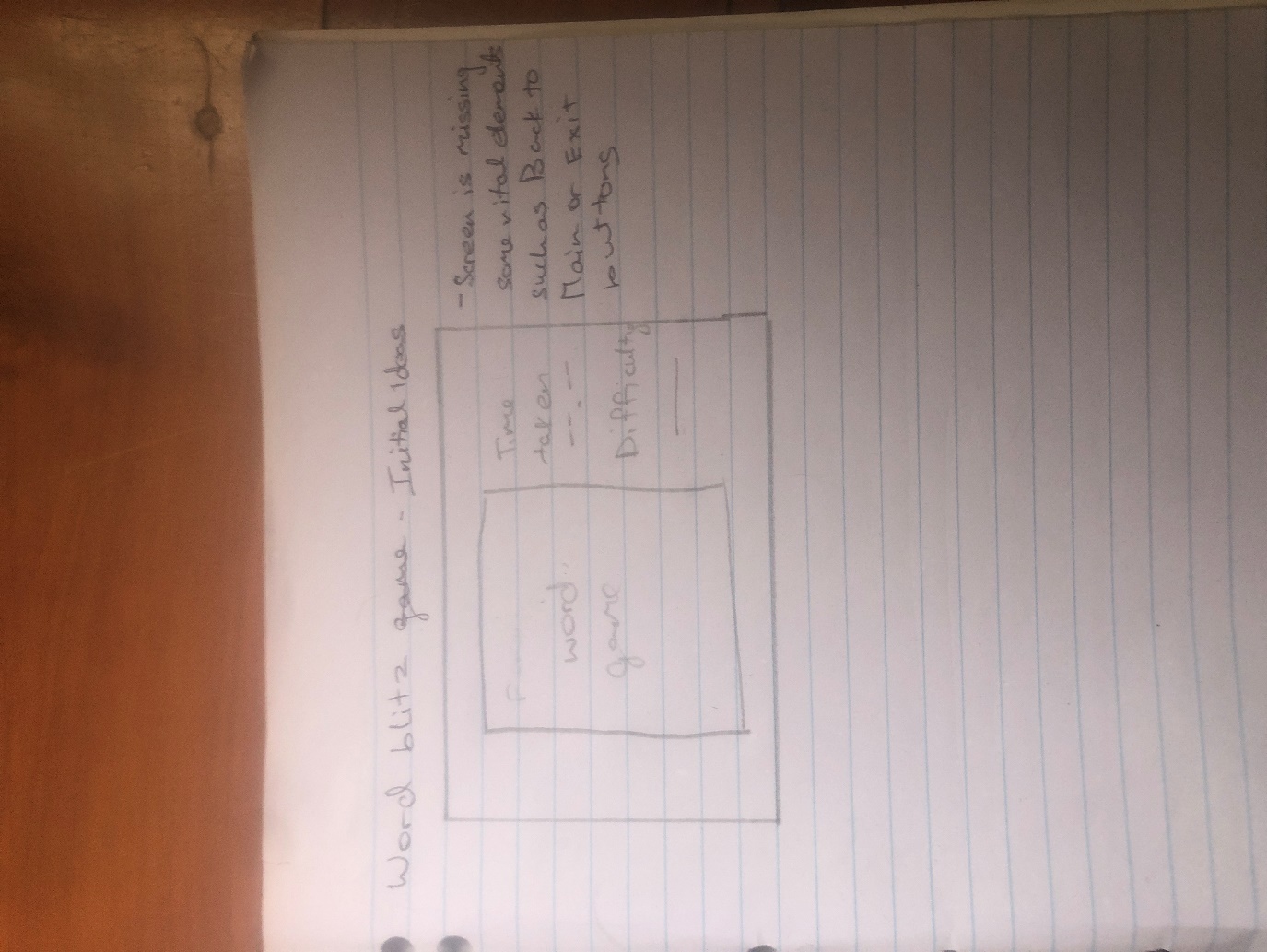
**Image Match Final Design – On Next Page**



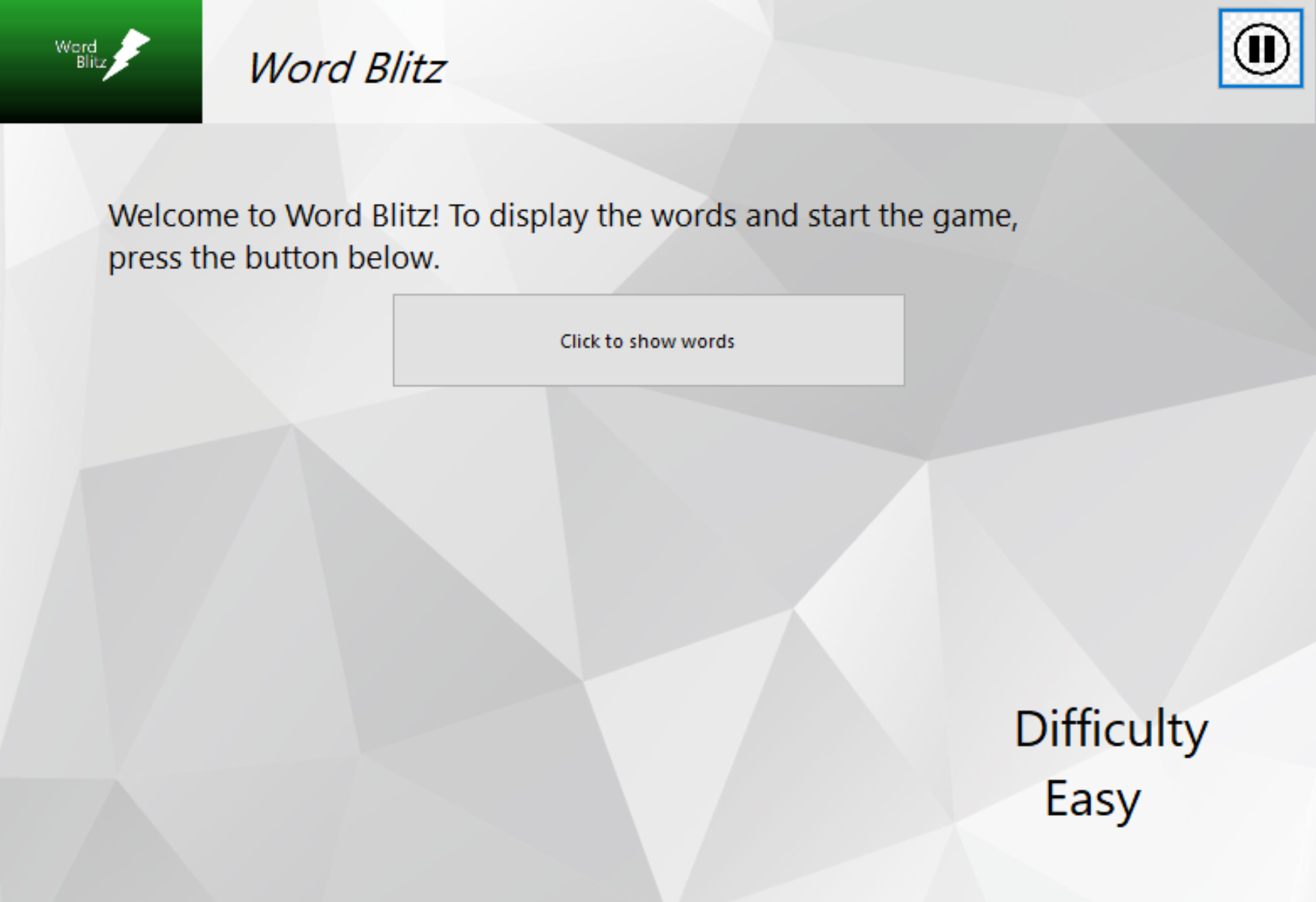
Use of a header to add some consistency to the screen design. The Exit button is now replaced by a pause Button

This space is now used for all game-related screen elements. They are also spaced out and lined up to provide an easier-to-read and intuitive screen design. It also adds consistency to the screen design

Word Blitz Initial Sketch (Hand-Drawn)

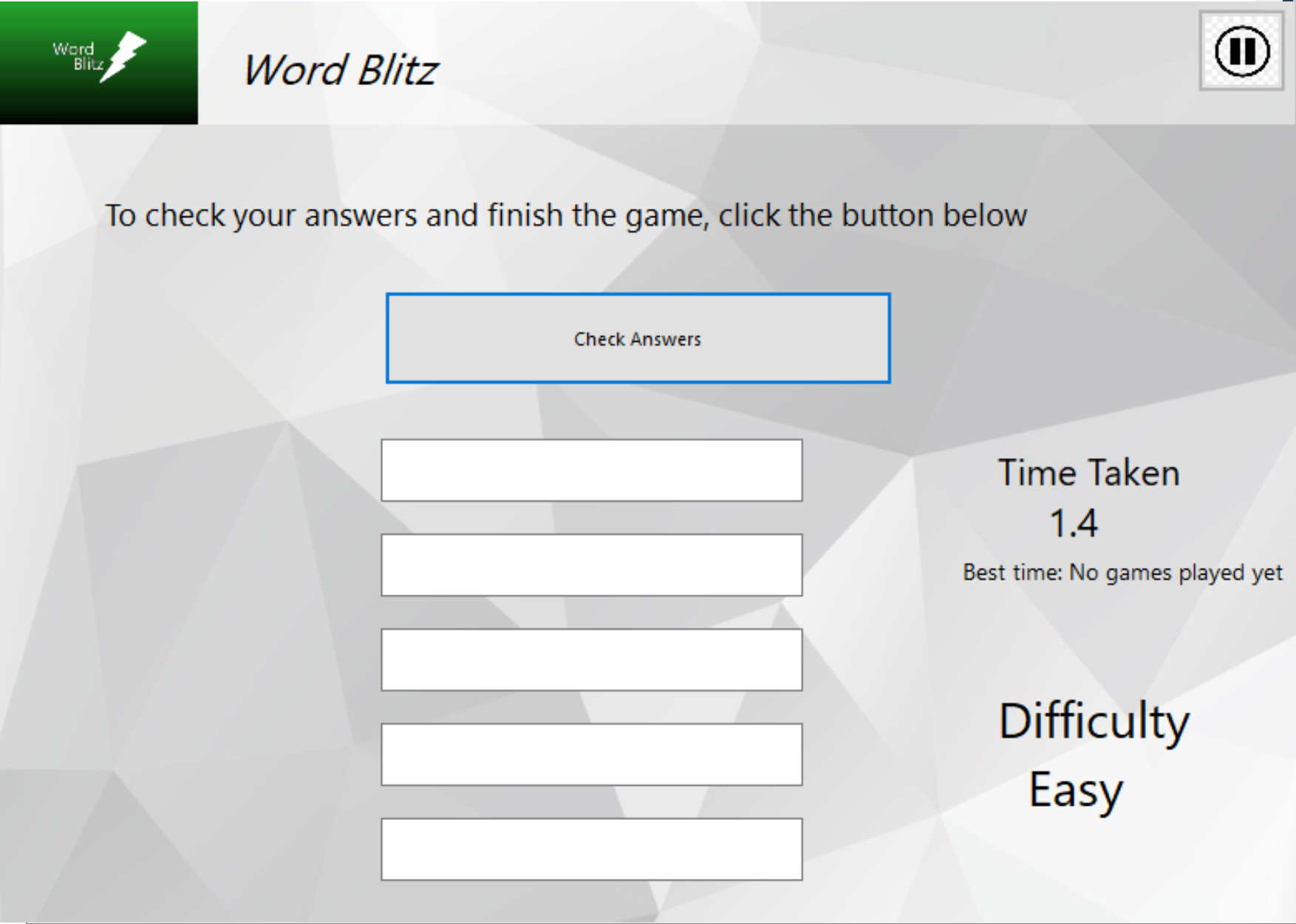


Word Blitz Final Design



Use of a header to add some consistency to the screen design.

The button for spawning in the flashing words, textboxes and later on confirming user inputs and finishing the game is extremely important so it has been given prominence in the screen design



This space is now used for all game-related screen elements. They are also spaced out and lined up to provide an easier-to-read and intuitive screen design. It also adds consistency to the screen design.

# Project Development Timeline

A list of my responsibilities are as follows:

* Defining the Problem
* Planning the game’s features
* Designing the UI of the game
* Implementing the solution
* Testing and refining the solution.
* Creating the documentation and user guide for the project

Below is the outline of what happened in the project timeframe. A gantt chart has also been provided for a visual representation.

**Week 8 (of the school term)**

Started:

* Defining the problem
* Planning the features

Completed:

* Defining the problem
* Planning the features

In this week, I read the assessment notification to determine what the problem required of me, as well as what features I could put into my game. However, due to exams, I had to do a lot of work which did not leave a lot of time to do anything else.

**Week 9 (of School Term)**

Started:

* Designing the UI of the game

In this week, I started to come up with ideas as to the screen design of the forms of the game. However, due to UI design being complex and also due to the huge workload caused by the exams, I was unable to finish this task.

**Week 10 (of School Term)**

Started:

* Implementing the solution
* Testing and refining the solution.

Completed:

* Designing the UI of the game

In this week, I was finally able to finish designing the UI of all the forms in the game. In this week, I also began actually coding the solution, and also testing and refining it while I was coding.

**Week 1 (of Holidays)**

Continued:

* Implementing the solution
* Testing and Refining the solution

In this week, I continued to code and refine my solution, however, due to problems and hiccups encountered, I was unable to finish either task this week

**Week 2 (of Holidays)**

Started:

* Creating the documentation and user guide for the project

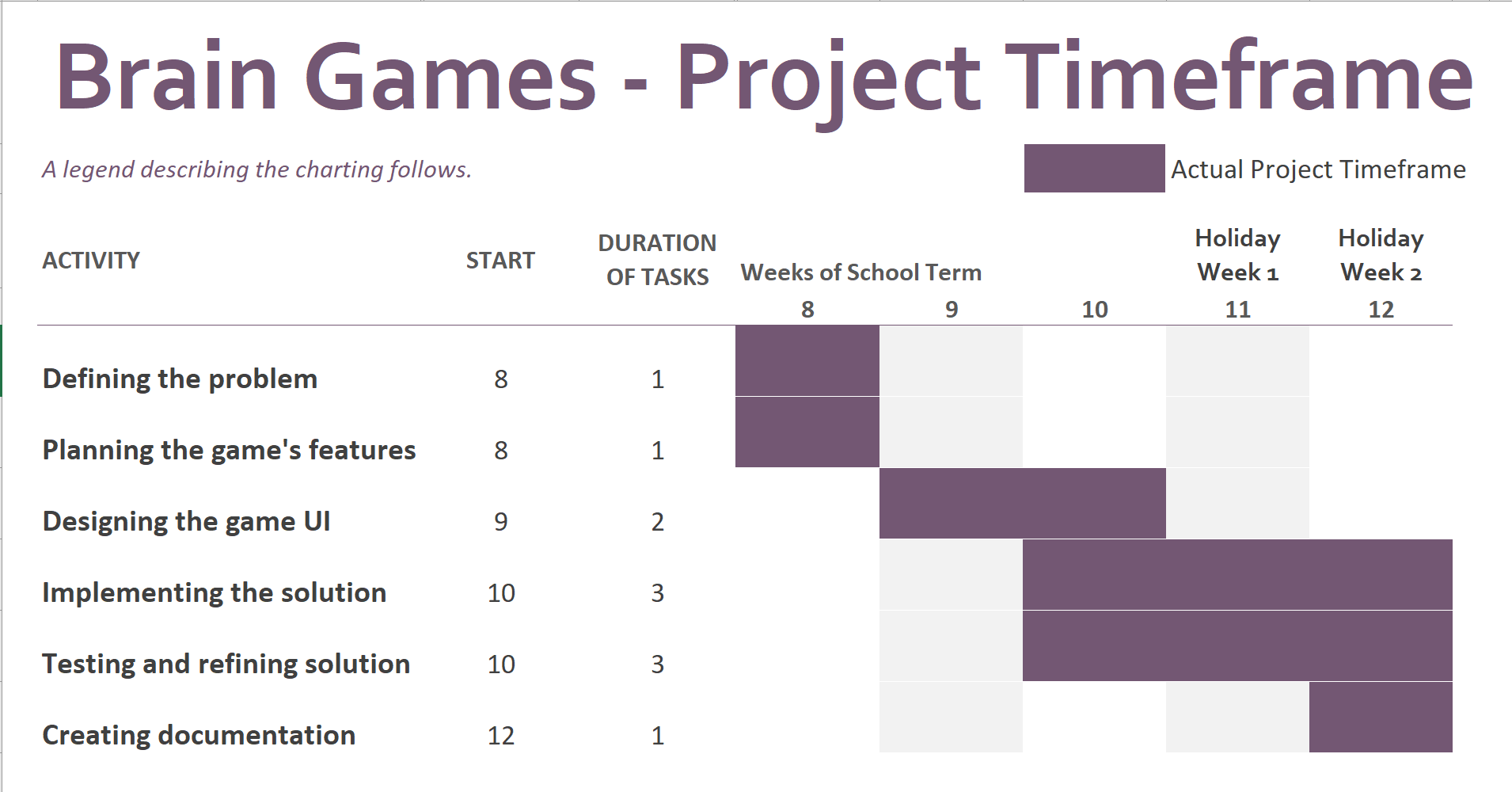
Completed:

* Implementing the solution
* Testing and Refining the solution
* Creating the documentation and user guide for the project

In this week, I finally finish putting the solution into code and refined it to a good product. I also started creating the documentation and user guide for the program this week.

A visual representation of this info in the form of a Gantt Chart is provided below.

# Gantt Chart



Part 2

Project Procedure Report

# Project development and approach

I used the RAD approach to complete my solution as it allowed me to:

* Create reusable modules of code to be used throughout the project,
* Use existing modules of code in my solution,
* Complete the solution in a relatively small project time frame of 5 weeks, albeit increasing the risk of stability issues and bugs in the solution
* Complete the project without any monetary budget,
* Skip the need for excessive documentation, as other approaches such as the structured approach requires a massive amount of documentation which I could not create in the short project timeframe.

Moreover, I did not use tools such as the context diagrams and flowcharts in the development of my game that would be required in the traditional approach, as I utilized a RAD approach.

# Final Solutions and Reasoning

My goal for this game was to create a game that was not only complex in its coding, but also something that would be fun and challenging to play and use, and to create something that would test and exercise my basic coding capabilities. My main focus for this game was:

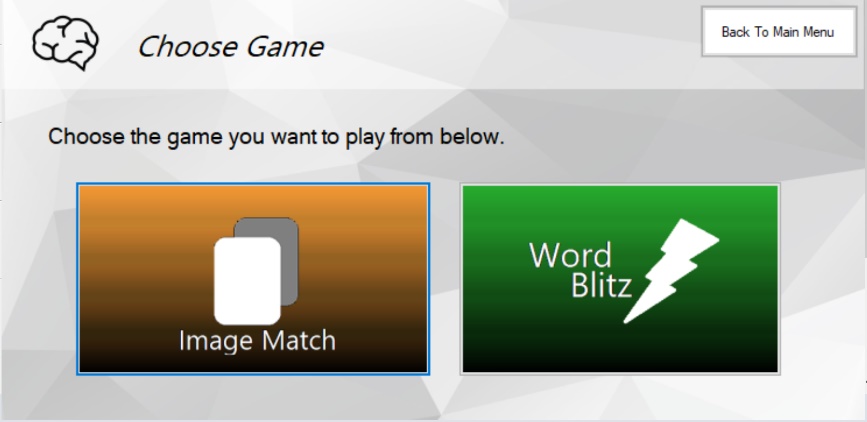
* To make a simple and elegant user interface
* To make gameplay a fun and challenging component for users
* Customizability

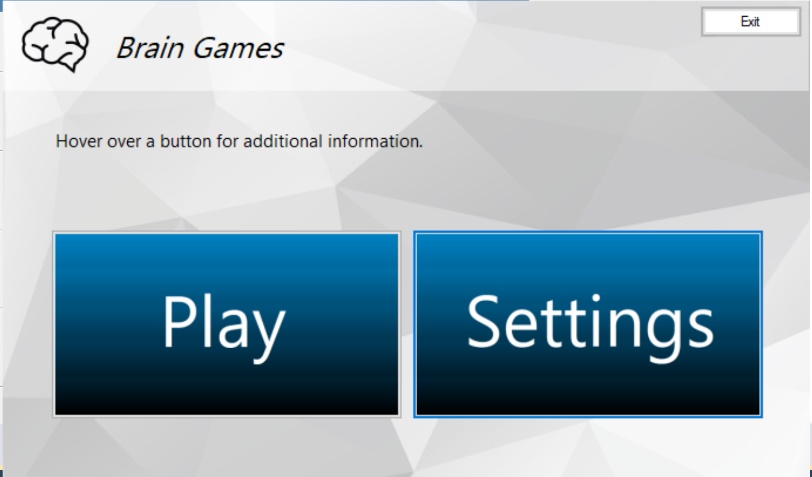
I will go over each of these points briefly with reference to my final solution.

**To make a simple and elegant user interface**

This was an important focus in my creation of Brain Games. This was extremely important to ensure that anyone, even those who may not be great with computers and operating software may be able to use the software.

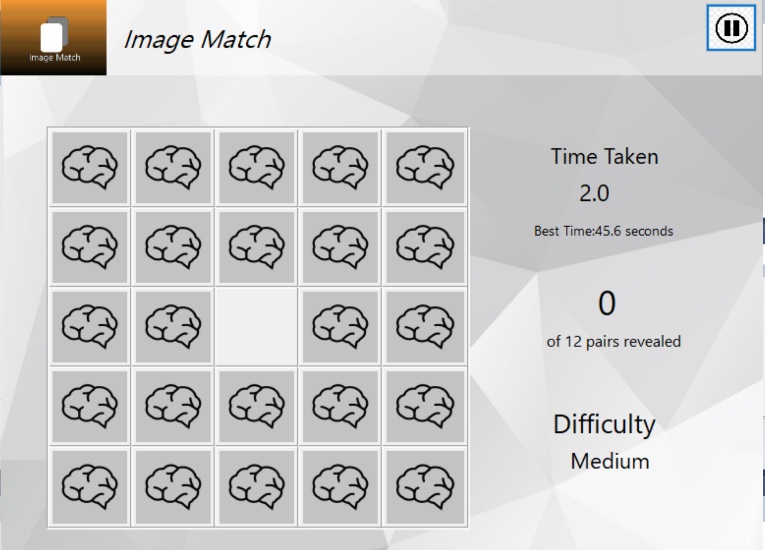
I attempted to achieve this through the use of consistency in screen design. By making the screen design, the user only needs to learn ho to operate one form in the game in order to have a good idea as to how to operate the rest of the game using their previous knowledge, thus saving them some time and frustration and also making the game more convenient to play.

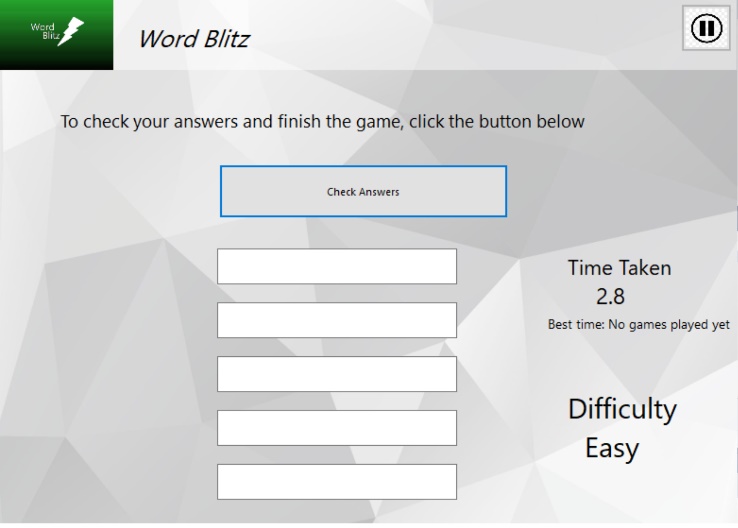
For example, the big buttons on the screens such as Play, Image Match, Settings, and Word Blitz all utilize the same hover-over and resting effects and are also situated in the same places on the forms to add an element of consistency to the game play as can be seen in the images below.



Moreover, the use of a header also adds massively to the consistency of screen design within the game as the navigational buttons such as the Exit and Main Menu buttons can also be found in the headers. They are also of the same formatting and position within the header.

Furthermore, within the game windows, game specific statistics and elements such as best time, difficulty and “Click to show words” in Word Blitz are all placed logically and in the right places within the screen design.





As seen above, the time and difficulty are locate in the same places within the form and the pause button is located within the same positions within the headers, which are also located in the same places.

A good software experience is made up of one that is pleasing to the eyes, but does not sacrifice actual function for aesthetic experience. As such, all my form elements were designed to be utilised to the maximum by the game’s logic while also serving an aesthetic function. I had considered using oblique fonts for my buttons, but concluded that the normal fonts were better for the sake of legibility. The weird colours for the Image match and Word blitz buttons in the Choose Game form, were chosen mainly to help the user differentiate each game in their mind, while aesthetic value was also considered, but to a lesser extent.

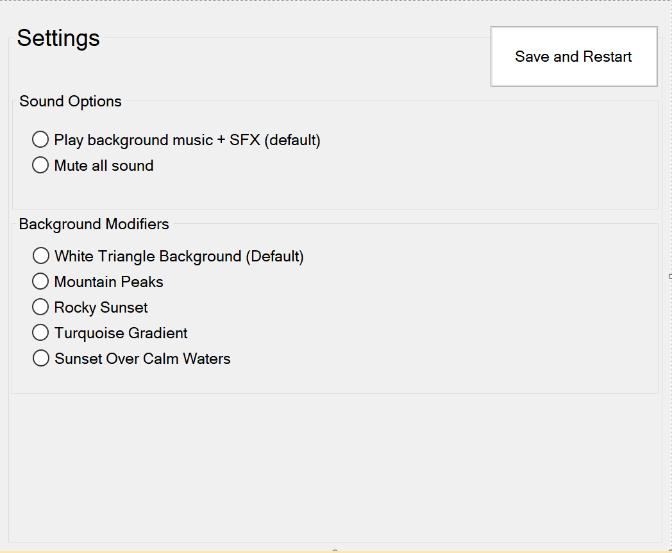
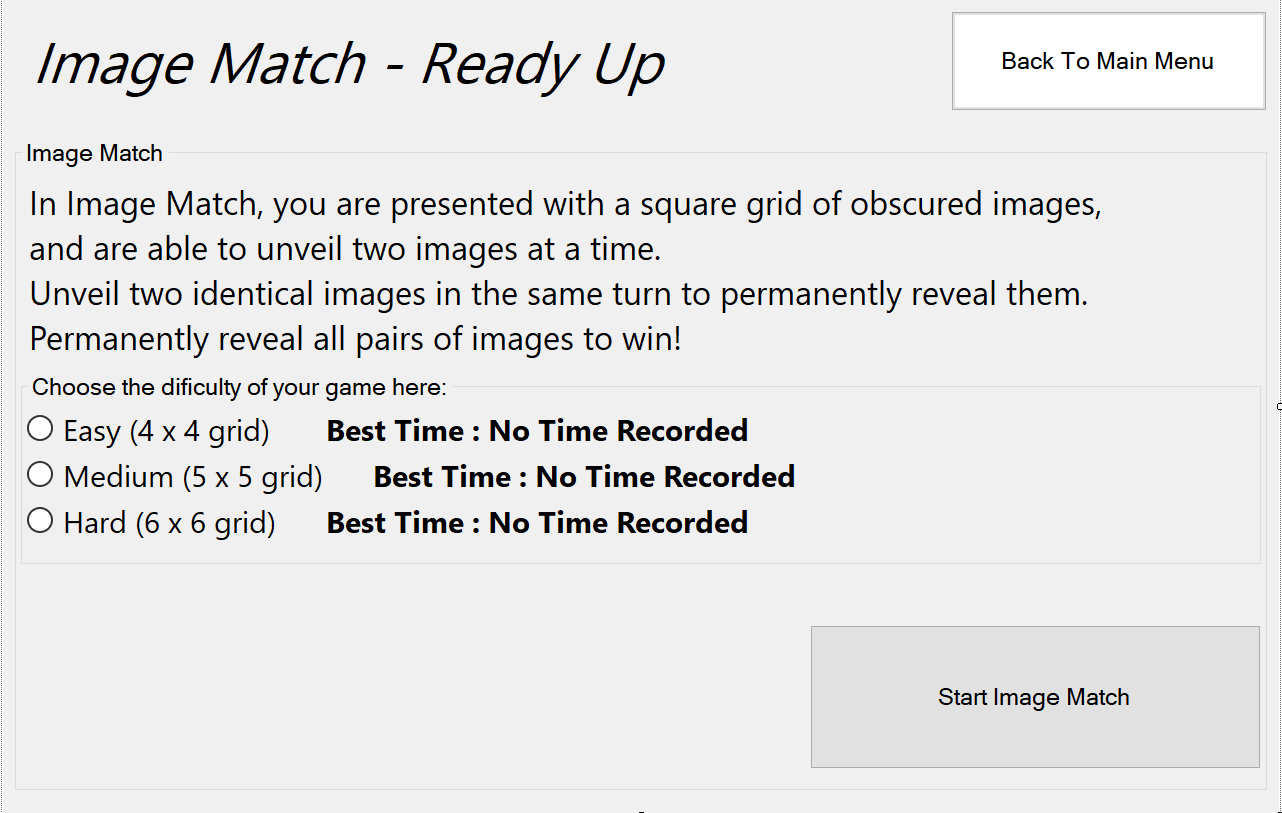
**To make gameplay a fun and challenging component for users**

A game is not fun if it is not challenging and gives player a sense of overcoming such challenge. To give the user a sense of overcoming such challenge a “best time” feature was also implemented to track the best time of the user for each difficulty of each game. Through this, the user can have a fun time attempting to beat their best scores. Moreover, to make the game more or less challenging, the user can also play the games on different difficulties. The games were also designed with simple ideas in consideration, and to be engaging and elegant.

**Customizability**

Customizability as also one of my major focuses when it came to making the game. The user should be able to change the environment of the game to what they wish to customize their experience of the game to be more aesthetically suited to them. To satisfy this need, I have implemented a feature in settings where the user can choose from one of five different backgrounds for the main forms in the game.

Furthermore, I have also implemented a difficulty feature for both Image Match and Word Blitz in my solution, so that the user can then make either game more or less challenging, depending on their preferences.



Both features can be seen in the images above.

# Problems Encountered

Over the course of the development timeframe, I had encountered many problem, mainly in the implementation of the solutions. I will go over the problems encountered briefly in this section.

* **Having the image match be different sizes for different difficulties:** One of the first and most major problems that I had to face was how to make the image match game be different sizes for each of the 3 difficulties that can be chosen by the user. Initially, I wanted to make a static game, with three different windows, one for each difficulty. However, I didn’t go through with it, as I concluded that that would be a waste of time and that it was the easy way out and not an intuitive solution to the problem at all. So, after searching for a solution, I used a nested loop to create new picture-boxes and have them fill a table layout panel, whose size would be modified beforehand, depending on the size of the game required. The counters for the nested loops would also be used as the ‘x’ and ‘y’ coordinates that can then be used to refer to each picture-box. This was an extremely intuitive solution to the problem that took less time to code and was a lot more dynamic than the static approach of creating separate forms for each of the difficulties. I could then use this module of code to spawn in textboxes and labels for the word blitz game as well.
* **Making the game register the clicks on the picture-boxes:** This was another minor issue that was quickly resolved. I used the AddHandler, AddressOf line to add a click handler to each picture-box as they were being generated in the nested loops. Each picture box would have a default image, and upon being clicked, they would display a different image.
* **Comparing the two clicked images:** This was yet another issue that was a bit harder to resolve, as I couldn’t simply use an If statement to compare the images attributed to each clicked picture-box, as images cannot be compared. However, after extensive thinking, I found that if I found the directory of the image attributed to each picture-box, I could then compare these file paths, as they were simply string values, which can be easily compared and used within an If statement, because if the file paths are equal, then both the picture-boxes were displaying the same image.

# Usage of Intrinsic and Internal Documentation

Internal and intrinsic documentation is important in this solution as it allows for easy comprehension of each line of code and the role it plays in the solution and thus, the maintenance and later modification of said code as well. Intrinsic documentation is used throughout the entire program, and in each and every form.

**Meaningful variable and control names**

Each and every name of each of my variables, elements and controls was given so that the code is easy to comprehend and so that the name of the controls and procedures clarify the purpose of the control itself. The naming convention for variables within the forms also clarify the type of the control in addition to its purpose.

For example:

lblScore

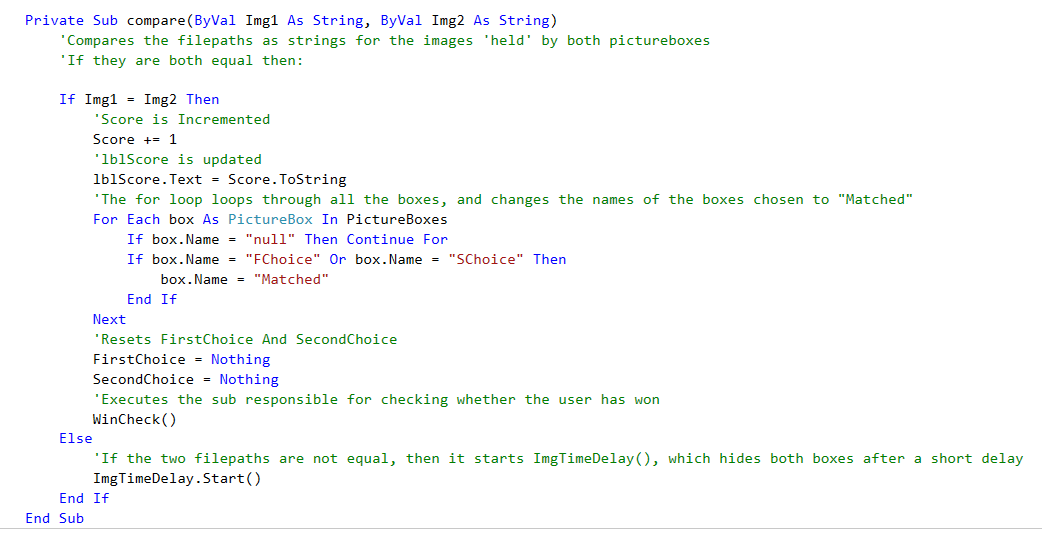
We can see that this is a label as shown by the prefix, and we can see that it displays the score, as shown by the suffix.

ImgGrid

We can see that this is a “grid” – which tells us that it’s a table layout panel and that it is responsible for holding images as suggested by its prefix.

**Use Of Indentations**

Indentations have been used throughout the code to create hierarchy and cohesion within the code. This also makes the code easier on the eyes, and thus easier to understand. For example, take a look at this segment of code:



The use of indentations is extremely important as there are multiple if, case and for statements within the code. The indentations allow the programmer to identify the different levels of such statements in order to make sense of what’s going on in the code.

**Extensive commenting**

The comments also help the programmer understand what’s going on by explaining what each line of code does in detail and explaining the purpose of each declared variables. The comments are in plain documentation. The segment of code above also has been extensively commented on by me, and it is extremely helpful to an outsider looking at this code, as the comments can help clear any doubts about what the program is doing.

This makes the implementation of later bug fixes and updates less frustrating for the programmer.