

Sarah Bibb

Advanced Higher Computing Project

Hyndland Secondary School

Scottish Candidate Number: 061389148

Date of Birth: 09/11/2000

Contents

# Project Plan…………………………………………………………………………………………………....3

# Requirements Specification………………………………………………………………….……...10

# Test Plan…………………………………………………………………........................................15

# Interface Design……………………………………………………………………………………..….…37

# Structure Design…………………………………………………………………………………..………48

# Implementation………………………………………………………………………………..………….61

# Testing………………………………………………………………………………………………………....82

# Evaluation……………………………………………………………………………………….………….154

# Record of Progress………………………………………………………………………….………….159

Project Plan

# Project Proposal

For my project I am going to create a program to help people revise by using flashcards which they can design.

The user can create a deck of cards and save it to use it again. The values for the cards will be stored in a linked list, which will be written to an external file so that it can be accessed after the program is closed. They can then test themselves on how well they remember the cards by beginning the quiz, where they will be asked the question by the computer and have to remember the corresponding answer. The user can then reveal the correct answer, and indicate to the program whether they answered the question correctly or not. They can also test themselves on how many cards they can revise in a certain amount of time, to try and improve their speed.

The amount of time the user takes to complete the full deck will be timed and displayed at the end of the revision, along with how many questions they got right or wrong. The best scores will be saved in an external file, as this allows the user to try and beat the old scores and improve their knowledge. The user can view the high scores for each deck in a separate window.

# Feasibility Study

## Legal Feasibility

The program should not have any legal issues, as the solution will not involve anything which is owned by another organisation, so copyright will not be an issue. If I use any pictures, I will create them myself to ensure that I am not breaching copyright on an image, by using it without permission, to make sure I comply with the Copyright Designs and Patents Act. If I use any work which is not my own in my program, I will make sure to credit the creator so that I am not claiming credit for it. I will also make sure to get any resources I find from other people from places where they are meant to be used by other people, and are intended to be taken. The Data Protection Act also does not apply to the program, as it only takes the user's name, which is not personal data. Nothing the user inputs is data which needs to be stored carefully or is information which could be used to identify them, so I am not required to store it with security, and I am allowed to keep the information without breaching the Data Protection Act, as it is not identifying data.

## Economic Feasibility

My project does not require any money to create it, as it using software that is already owned and downloaded on the systems I am using, and it will not make any money, so there are no economic factors to be considered.

I will be completing all of the work involved in the project, so there are no salaries which need to be paid and the final solution is not being sold so there are no profits which need to be considered.

## Technical Feasibility

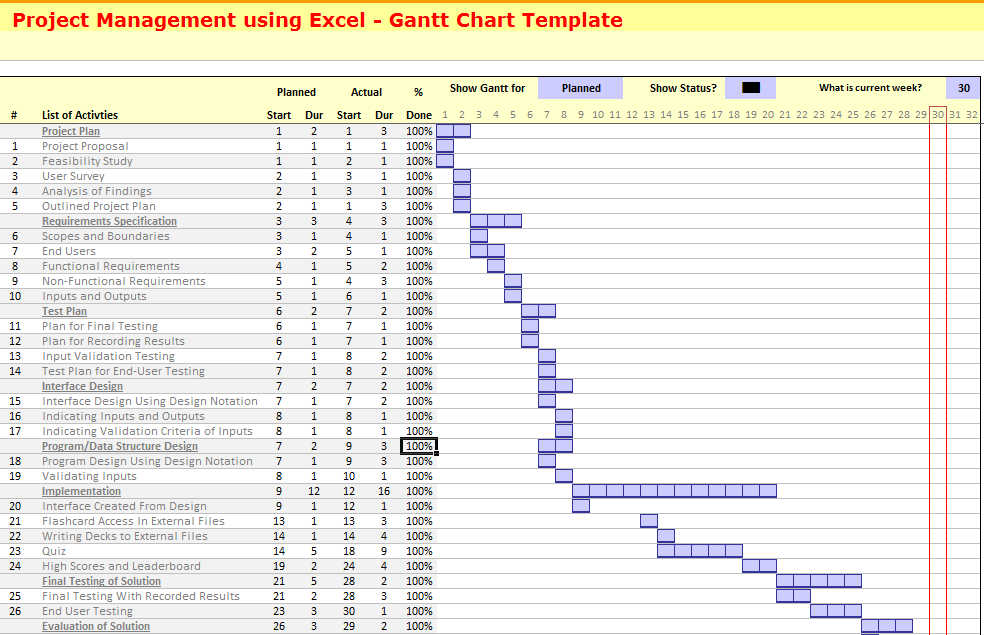
To create my project, I am using Visual Basic, which makes my project more feasible, because I am already familiar with this programming language, which means I don't have to spend time becoming familiar with a new language, I simply have to improve my knowledge of one I have been using for a few years. As well as this it makes the creation of my solution easier as the systems I am using at school already have Visual Studio downloaded on them, which means I have an IDE I am familiar with to use to create my solution, without having to acquire any extra specialist technology. If I used another language it would be likely that extra software would have to be downloaded, and this may require storage space and authorization, but as everything required to use Visual Basic is already available to me, this is not the case. Overall my project is feasible as it only requires software already on the computers I will be using to create my program, and for me to use a language I have experience in.

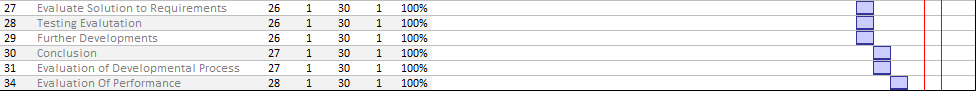
## Schedule Feasibility

While doing my project, I will make sure to plan my activities and manage my time carefully, to ensure to the best of my ability that I complete each task within my time frame and finish the project before the deadline. If I create small deadlines for smaller tasks throughout the project, I can easily make sure I am still working at the speed I planned to, and keeping up with the work required to get the project completed by the end. If I make sure to plan my schedule, the project can be completed within the allowed time.

By creating a Gantt chart detailing the main tasks and sub tasks which must be completed for my project, I will ensure that I can see if I am on track with my schedule, and if I fall behind schedule I can see how far behind I am and what I need to do to get back on track. This will help me as it will allow me to monitor my progress very easily by using a clear display of tasks and completion progress.

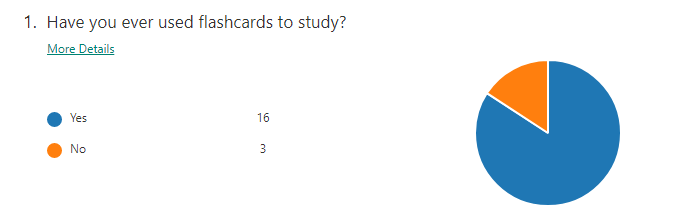
# Gantt Chart



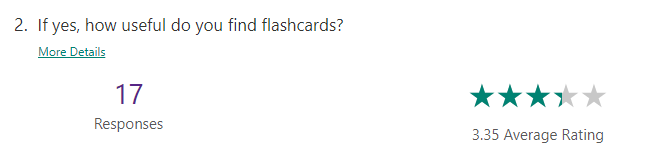


# User Survey

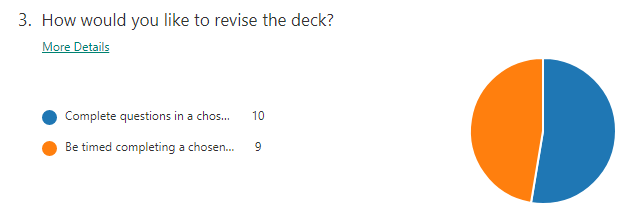
To help ensure that my solution was as successful as possible, I conducted a survey to see what features and specifications people would like for my program. It asked a few simple questions to allow me to see what people would find the most useful. I used Microsoft Forms to do this, as it allowed me to quickly create a simple survey which I could share with people immediately, so that it could be completed quickly and easily.



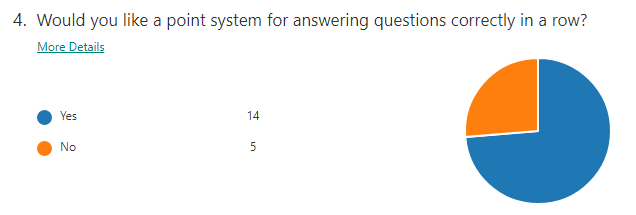
This allowed me to see that a lot of people have tried using flash cards. This means that many people will know how to use flash cards already and may use them in their own studying. Since this is likely, this means that my program will be beneficial to them as it will allow them to study more efficiently and effectively.



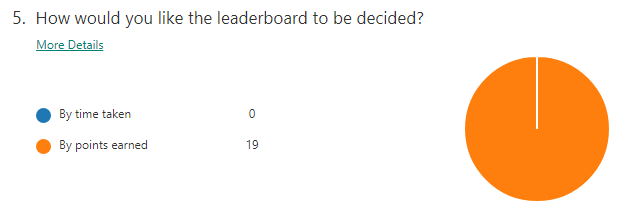
This question let me see whether people enjoyed using flash cards, or whether they found them less useful for studying. By the 3.35 average rating from the people who responded, I can see that people generally find flashcards useful; however there are parts of flash cards which they find less convenient. I can aim to get rid of these inconveniences, so that people can study using flash cards without having to deal with the more time-consuming aspects, like creating the physical cards.



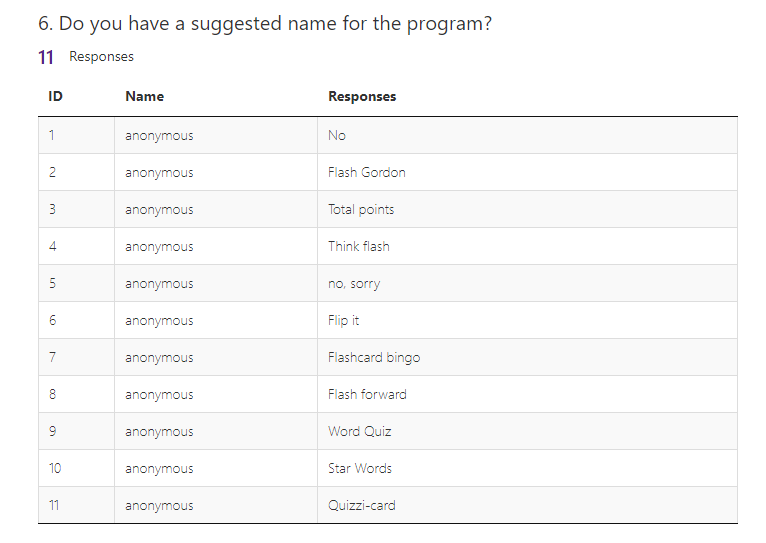
While I will definitely offer the option to revise the whole deck at once, I was also interested in whether people would also like to study a smaller proportion of the deck. The results show that both choosing a time and choosing an amount of questions are fairly popular, but completing questions in a chosen amount of time is slightly more popular. I also think this is the more advantageous method to use for revision, as it forces the user to recall information quickly, so it will be the method I am implementing in my final program.



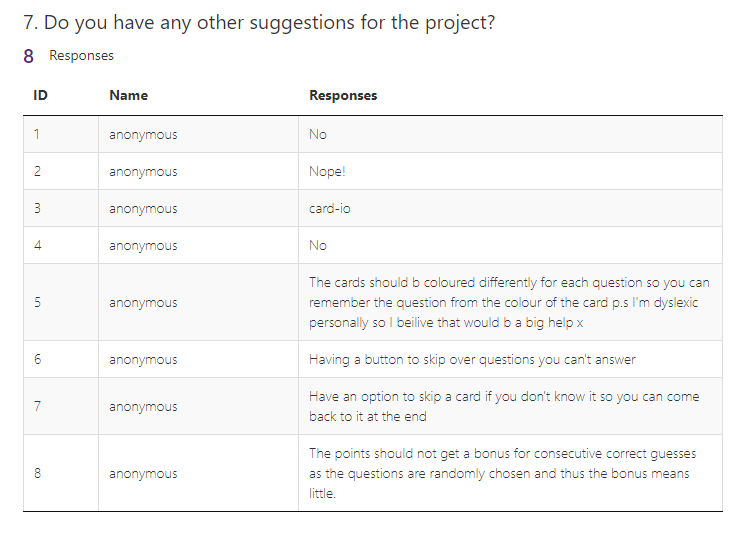
Using points can be a useful way to measure improvement, so I decided to see whether potential users would like this feature, as it seems valuable to their learning. Overall, the majority of people would like a points system. However, during the development of the project I decided it was more advantageous not to use points, because they would be redundant, since I would not assign more points for answering questions correctly in a row, due to feedback I received later in the survey. Instead of using points, I can compare scores using the amount of questions they got right.



Although the leader board no longer includes points, and only includes the amount of correct answers, the decision to organize it using correct answers instead of time is beneficial for the user as it encourages them to take their time to get questions right, which will improve their learning.



Here, I asked the people taking the survey whether they had a suggested name for the program. Although many names were submitted by the users here, I have decided to use a name which was submitted under the extra suggestions section of the survey, as I prefer that name to the ones submitted here.



This is where my preferred name for my program, "Card-io" was submitted. I have chosen this name because it incorporates the card feature of the program with the word cardio, making it memorable and entertaining for the user.

Another suggestion I found useful was the suggestion to use colour to help people with their revision. This was not a possibility I had considered but finding out that it would improve people's memory made me think that this would be a useful feature to add to my program if it is possible.

As well as this, I have also decided to include the option to skip a question in the quiz. Originally, I was simply going to have the option for a correct and incorrect answer, but the suggestions made me realise that it would be useful to have the option to skip a question if the user doesn't know the answer, so that they can return to the question at the end of the quiz, after they have answered the questions they know.

As mentioned early in the survey analysis, I realised that the best way to carry out the points system is to award points for answering questions regardless of whether they were answered correctly consecutively. This is because, as outlined in the suggestion, seeing as the questions are random, whether or not the user knows the questions that come up in a row is purely chance and doesn't reflect their skill or any possible improvement they have made since the last time they took the quiz. Because of this decision, later in the development I decided to remove the points entirely.

Requirements Specification

# Scope

The scope of the project is to create a program which allows the user to test their knowledge using the revision method of flashcards. The project's objective is to help people revise by allowing them to input information they want to test themselves on, but making it more convenient for them as all the information is saved by the program, and it will randomise the cards and time the user so that they don't have to.

# Boundaries

As the project needs to be completed within a certain timeframe, there will be limits to how much can be achieved in the time given. Due to this, the solution must stay within the specified requirements, to ensure that it meets the requirements it must meet and that they are completed within the time that is available. As well as this, the program must be within Advanced Higher level, so that it is realistically achievable. For this reason, some features must be simplified, for instance letting the user tell the program whether they got the answer correct or incorrect instead of the program checking the answer, as this will ensure the program remains feasible.

# End Users

My program is aimed at people who want to study a certain subject by looking over their information quickly and efficiently. Although it is available to older people, it will likely be mainly used by students, who will be younger and more computer literate. This means that they will likely know slightly more advanced computer functions, but to make sure it is available to people studying who are not as computer literate I will try to ensure the program is easy to understand for novice level users as well. By using buttons and radio buttons to allow the user to make their selections, I ensure that the choices they have are simple to see and intuitive to select.

# Inputs

My project will have decks of cards, names of the decks, and high scores from external files stored on the computer in the form of .csv files, and this is one input the program will require.

The program must also take the user's name, so it can store high scores in external files. The name must only include letters, not numbers or special characters.

When creating a card in the Deck Editor, the user will need to input a question and answer, which will need to have a limited length, as well as validation to ensure that something is entered. The editor also allows the user to select a colour, but these will chosen through radio buttons so they are already validated.

The user must input a time if they select to only revise a portion of the deck. This will need to be validated to ensure it is not too big.

The main form of input used in my program is through buttons pressed on the menu and during the quiz, for instance to open the Deck Editor, to begin the quiz or to select an answer on a question.

# Outputs

My project will output the high scores of the deck being viewed on the High Score window.

It will also output .csv files saving the high scores and decks of flash cards when these need to be saved.

While the user does the quiz, the program will output a randomly selected question which will be read in from an external file, and answer once the button is pressed to reveal it.

# User Requirements

Requirements for the Main Menu:

* Allow the user to enter their name when the Main Menu opens
* Allow the user to open a previously made deck
* Allow the user to open the Deck Editor to create or edit a deck
* Allow the user to open the High Scores window for the deck they have open
* Allow the user to open the Instructions window to view the Instructions
* Allow the user to select whether they want to review the whole deck or a timed portion of the deck
* Allow the user to select a time to revise the deck for the timed option of the quiz
* Allow the user to close the program

Requirements for the High Score window:

* Allow the user to view the highest 3 scores, along with the names and times, of the deck they have opened in the Main Menu.
* Allow the user to return to the Main Menu

Requirements for the Instructions window:

* Has separate instructions for each section of the program
* Can display the instructions to the user when each section is selected
* Has instructions for opening a deck, making or editing a deck, viewing high scores and starting the quiz
* Allow the user to view the instructions, then return to select a different set of instructions
* Allow the user to return to the Main Menu

Requirements for Deck Editor window:

* Allow the user to enter a question and answer for each card
* Allow the user to move backwards and forwards through the cards in the deck being edited
* Allow the user to select a colour for each card
* Allow the user to save their cards into a deck
* Allow the user to enter a name for a new deck
* Display the number of card being viewed
* Allow the user to return to the Main Menu

Requirements for the Quiz window:

* Can randomly select a question to display to the user
* Display the question clearly for each card
* Display the selected colour for each card
* Allow the user to reveal the answer
* Allow the user to select whether they got the question correct, incorrect or if they want to skip it
* Not allow the user to skip a question once they have revealed the answer
* Count the amount of questions the user answers correct or incorrect
* If a card is skipped, display it again in the quiz
* Allow the user to quit to the Main Menu
* Time the quiz

Requirements for Timed Quiz window (in addition to requirements for Quiz window)

* Quiz ends once the specified time has run out

Requirements for the Completed Quiz window:

* Displays the number of questions in the quiz the user answered correctly or incorrectly
* If the user achieves a high score, add their score and details to the list of high scores
* Save the list of high scores for the deck
* If the user achieves a high score, display this at the end of the quiz
* Allow the user to return to the Main Menu

# Operational Requirements

Requirements for the Main Menu:

* Use a form to display the Main Menu
* Use an input box to get the user's name and store it in a string variable
* Validate the user's name to confirm it only contains letters
* Have buttons on the Main Menu to open a deck
* Have a button allowing the user to open the window which lets them edit a deck or create a new deck
* Have a button allowing the user to open the window which displays the high scores.
* Have a button allowing the user to open the window which displays the instructions.
* Confirm if there are high scores for the currently open deck if the high scores button is selected and display an error message if there are not.
* Allow the user to select whether they want to revise the whole deck or a portion of it using radio buttons on the Main Menu
* Have a button for the user to begin the quiz using the deck they have selected
* When a deck is opened, read the questions and answers into a linked list to store them while it is open
* Have a label which displays the name of the deck which is open
* Retrieve the selected time for the timed option using an input box
* Validate the selected time to confirm that the input only contains numbers, and that is between 1 and 300 inclusive.
* Use an integer variable to store the time entered if the user selects the timed option
* Have a button which closes the program

Requirements for the High Score window:

* Read in the scores file for the deck which is open.
* Display the 3 names in a list box
* Display the 3 amounts of correct answers in a list box
* Display the 3 amounts of incorrect answers in a list box
* Display the 3 times in a list box
* Have a button which closes this window

Requirements for the Instructions window:

* Has four buttons for the different instructions that are visible when the form is loaded
* Has four labels containing the instructions for each section
* When a button is selected it hides the buttons and other labels and displays the correct label with the instructions for that section
* Has a back button which will hide the instructions and show the buttons again, so the user can choose a different section
* If the back button is selected while the buttons are visible, the window will close

Requirements for the Deck Editor window:

* Use Textboxes for the user to enter the questions and answers for each card
* Have radio buttons for colour selection for each card
* Have a button which creates a new card from the information which has been entered the text boxes and the radio buttons
* Use buttons to allow the user to move 1 back or 1 forward in the list of cards in the deck
* Use a label to display the current card the user is viewing by displaying the number in the linked list that card is stored at
* Have a button which allows the user to close this window and return to the Main Menu
* When the quit button is selected ask the user if they want to save the deck
* If the user chooses to save the deck, write the deck to an external .csv file which stores the question, answer and colour of each card

Requirements for the Quiz window:

* Use a list box to display the question and answer for each card
* Use a random number generator to select the question to ask the user next
* Display the colour of the card on the quiz form by changing the form's colour to the selected colour
* Use an array to store the numbers which have been used already to choose questions and check the randomly selected number against these, if it is already used generate a new one
* Use a label to display how many cards are in the deck
* Have a button to allow the user to select answered correctly, incorrectly or skip the question to come back to at the end
* Use a Boolean variable to check if the answer is revealed, once the answer is revealed the user should not be able to skip the question
* If a question is skipped, it is not added to the array of used numbers so that it will be chosen again
* Use two integer variables to count the amount of cards answered correctly or incorrectly
* Use the timer function to time the user
* Use an integer to store the time taken

Requirements for Timed Quiz window (in addition to requirements for Quiz window)

* Compare the time passed which is stored in the integer variable to the time selected at the start of the quiz, when it reaches that number end the quiz

Requirements for the Completed Quiz window:

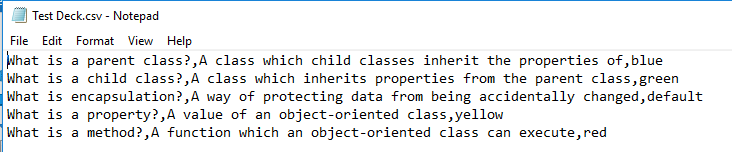
* Displays the numbers stored in the correct and incorrect variables in labels to show the user how many questions they answered correctly or incorrectly
* Use a Boolean variable to check if the user has achieved a high score, and display in a label that they have achieved a high score
* Compare the correct questions answered to the correct answers in the external csv file which stores the high scores for the selected deck, if their score is higher than any of them use a bubble sort to put them in the correct order, and write the updated arrays to file
* Use a button to allow them to return to the Main Menu

Test Plan

# Comprehensive Test Plan

## Main Menu

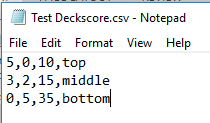
To test the Main Menu's ability to open a deck containing cards, I used a file I made externally from the program, so that I could ensure that the Main Menu was functional before testing the Deck Editor. By doing this instead of using a deck I had made in the Deck Editor, it allowed me to test the Main Menu even if the Deck Editor was not functional.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Entering the user's name. | Type "Sarah" into Input box asking for the user's name. | The program will set the string userName to "Sarah" | To ensure that the program can take the name of the user to use for scoring. |
| **2** | Opening files to select a premade deck. | Click the button "Open Deck". | A window will open containing the files on the computer allowing the user to select a deck file. The window will open automatically on the "decks" folder. | To ensure that the user is able to select a premade deck from the files saved on the computer, and that the program can open the window on the correct directory. |
| **3** | Selecting a valid deck file. | Select a .csv file entitled "Test Deck.csv". | The current.question, current.answer and current.colour variables will be equal each of the values from the file. The length will equal 5 at the end of the procedure. | To ensure that the program can successfully get the values from the file, and that it can count the length of the deck. |
| **4** | Selecting a valid deck file. | Select a .csv file entitled "Test Deck.csv". | The label openLbl will display "Test Deck" | To ensure that the program can display the name of the deck to the user. |
| **5** | Selecting an invalid deck file. | Select a .txt file entitled "test.txt" | The program will display a message box saying "Invalid, please select another file." | To ensure that the program can successfully deal with an invalid file type and continue running without errors. |
| **6** | Creating a new deck while there is no deck open. | Select the button "Create New Deck" before a deck has been opened. | The Main Menu will be replaced with the Deck Creation window. | To ensure that the user can get to the Deck Creation window. |
| **7** | Creating a new deck while there is no deck open. | Select the button "Create New Deck" before a deck has been opened. | The new\_deck variable will be set to true. | To ensure that the program knows that this is a new deck. |
| **8** | Creating a new deck when a deck is already open. | Select the button "Create New Deck" after opening "Test Deck.csv". | The program will display a message box asking the user if they want to close the deck that is currently open. | To ensure that the user can either close the current deck or cancel this choice. |
| **9** | Closing the deck which is currently open. | Select the New button after opening "Test Deck" and select "Yes" when asked if I would like to close the deck. | The program will open the Deck Editor, and the new\_deck variable will be set to True. | To ensure that the user can open a new deck if there is one already open, and that the Deck Editor will recognise that it is a new deck. |
| **10** | Not closing the deck which is currently open. | Select the New button after opening "Test Deck" and select "No" when asked if I would like to close the deck. | The program will not do anything. | To ensure that the user can decide not to close the current deck. |
| **11** | Opening a deck when a deck is already open. | Select the button "Open Deck" after opening "Test Deck.csv". | The program will display a message box asking the user if they want to close the deck that is currently open. | To ensure that the user can either close the current deck or cancel this choice. |
| **12** | Closing the deck which is currently open. | Select the Open button after opening "Test Deck", and select "Yes" when asked if I would like to close the deck. | The file selection window will be opened. | To ensure that the user can open a new deck if there is one already open. |
| **13** | Not closing the deck which is currently open. | Select the Open button after opening "Test Deck" and select "No" when asked if I would like to close the deck. | The program will not do anything. | To ensure that the user can decide not to close the current deck. |
| **14** | Selecting edit currently open deck. | Select the button "Edit Deck" after opening "Test Deck.csv" | The program will open the Deck Creation window with the selected deck available in it. | To ensure that the user can edit an already made deck to add questions to it when it is open. |
| **15** | Selecting edit when no deck is selected. | Select the button "Edit Deck" without having a deck open. | The program will display a message saying "Error: There is no deck open". | To ensure that the program can continue to run successfully after the user tries to edit a deck with no deck open. |
| **16** | Viewing the high scores for the currently open deck. | Select the button "High Scores" after opening the file "Test Deck.csv". | The High Scores window will be shown as a dialog. | To ensure the user can view the high scores for the deck they have open. |
| **17** | Trying to open the High Scores window when there is no deck open. | Select the button "High Scores" without having a deck open. | The program will display a message box saying "Error: No deck open." | To ensure that the program can continue to run successfully after the user tries to view the high scores for a deck with no deck open. |
| **18** | Opening the High Scores window for a deck which does not have any high scores. | Select the button "High Scores" after opening the file "Test Deck.csv" and ensure that there is no "Test Deckscores.csv" file in the scores folder. | The program will display a message box saying "Error: No high scores for this deck." | To ensure that the program can continue to run successfully after the user tries to view the high scores for a deck which does not have any high scores. |
| **19** | Opening the Instructions window. | Selecting the Instructions button. | The Instructions window will be opened as a dialog. | To ensure that the program can display the window which contains the instructions. |
| **20** | Opening the quiz without a deck selected. | Select the button "Start" without having a deck selected. | The program will display a message saying "Error: No deck open." | To ensure that the program can successfully continue running after the user tries to open the quiz without a deck selected. |
| **21** | Choosing to study the whole deck. | Select the radio button "Whole Deck" and then click the button "Start", with the test deck open. | The Main Menu will be replaced with the Quiz window. | To ensure that the user can revise the whole deck of cards. |
| **22** | Choosing to study cards in a certain amount of time. | Select the radio button "Timed Revision" and then click the button "Start" with the test deck open. | The program will display a message box asking the user how long they would like to revise for. | To ensure that the user can revise the deck for a set amount of time. |
| **23** | Selecting a time to revise the cards for. | Select the radio button "Timed Revision" and then click the button "Start", then enter 10. | The Main Menu window will be replaced with the Quiz window.  The limit variable will equal 10. | To ensure that the user can enter a time to revise the deck for. |
| **24** | Using the quit button to exit the program. | Click the button "Quit". | The user will be asked through a message box whether they would like to quit or not. | To ensure that the user can decide to quit the program from the Main Menu. |
| **25** | Choosing to quit using the Quit button on the Main Menu. | Select "Yes" after selecting the Quit button and being asked if I would like to quit. | The program will close. | To ensure that the user can exit from the Main Menu if they wish to. |
| **26** | Choosing not to quit after selecting the Quit button. | Selecting "No" after selecting the Quit button and being asked if I would like to quit. | The program will do nothing. | To ensure that the user can decide not to quit the program if they change their mind. |

## High Score Window

To test that the High Scores window could display the information correctly, I created a high scores file for the test deck which would allow me to clearly see that they were being displayed in the correct order. This was saved in the scores folder.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Displaying the names of each high score. | Opening the High Scores window with the test deck open. | The names will read "top" then "middle" then "bottom" in that order. | To ensure that the program can correctly display the high scores. |
| **2** | Displaying the correct answers of each high score. | Opening the High Scores window with the test deck open. | The numbers will read "5" then"3" then "0" in that order. | To ensure that the program can correctly display the high scores. |
| **3** | Displaying the incorrect answers of each high score. | Opening the High Scores window with the test deck open. | The numbers will read "0" then "2" then "5" in that order. | To ensure that the program can correctly display the high scores. |
| **4** | Displaying the times of each high score. | Opening the High Scores window with the test deck open. | The numbers will read "10 seconds" then "15 seconds" then "35 seconds" in that order. | To ensure that the program can correctly display the high scores. |
| **5** | Quitting the high score window to return to the Main Menu. | Clicking the Close button. | The High Score Window will close. | To ensure that the user can return to the Main Menu after viewing the high scores. |

## Instructions Window

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reasons for Testing** |
| **1** | Displaying the four buttons when the form is shown. | Opening the Instructions window using the "Instructions" button on the Main Menu. | The Instructions form will open as a dialog, with only the four buttons visible. | To ensure that the program can display the buttons for the user to select instructions with when the instructions are opened. |
| **2** | Displaying the instructions for opening a deck. | Selecting the "Opening a Deck" button. | The four buttons will be hidden, and it will show a label which contains the instructions on opening a deck. | To ensure that the user can view the instructions on how to open a deck. |
| **3** | Using the back button after viewing the instructions on opening a deck. | After selecting the "Opening a Deck" button, selecting the Back button. | The label containing the instructions will be hidden, and the four buttons will be displayed again. | To ensure that the user can return to viewing the buttons again after viewing the instructions on opening a deck. |
| **4** | Displaying the instructions for making or editing a deck. | Selecting the "Making or Editing a Deck" button. | The four buttons will be hidden, and it will show a label which contains the instructions on making or editing a deck. | To ensure that the user can view the instructions on how to make or edit a deck. |
| **5** | Using the back button after viewing the instructions on making or editing a deck. | After selecting the "Making or Editing a Deck" button, selecting the Back button. | The label containing the instructions will be hidden, and the four buttons will be displayed again. | To ensure that the user can return to viewing the buttons again after viewing the instructions on how to make or edit a deck. |
| **6** | Displaying the instructions for viewing high scores. | Selecting the "Viewing High Scores" button. | The four buttons will be hidden, and it will show a label which contains the instructions on viewing high scores. | To ensure that the user can view the instructions on how to view high scores. |
| **7** | Using the back button after viewing the instructions on viewing high scores. | After selecting the "Viewing High Scores" button, selecting the Back button. | The label containing the instructions will be hidden, and the four buttons will be displayed again. | To ensure that the user can return to viewing the buttons again after viewing the instructions on viewing high scores. |
| **8** | Displaying the instructions for starting the quiz. | Selecting the "Starting the Quiz" button. | The four buttons will be hidden, and it will show a label which contains the instructions on starting the quiz. | To ensure that the user can view the instructions on how to start the quiz. |
| **9** | Using the back button after viewing the instructions on starting the quiz. | After selecting the "Starting the Quiz" button, selecting the Back button. | The label containing the instructions will be hidden, and the four buttons will be displayed again. | To ensure that the user can return to viewing the buttons again after viewing the instructions on starting the quiz. |
| **10** | Using the back button to return to the Main Menu. | While the four buttons are visible, selecting the Back button. | The Instructions window will close. | To ensure that the user can use the back button to return to the Main Menu. |

## Deck Editor Window

To test the editing portion of the Deck Editor, I used the same Test Deck I used to test the Main Menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Opening a new deck. | Selecting the "New" button on the Main Menu while there is no deck open. | The Deck Editor window will open and both text boxes will be blank. | To ensure that the program can let the user create a new deck. |
| **2** | Opening a new deck. | Selecting the "New" button on the Main Menu while there is no deck open. | The position will equal 1. | To ensure that the program can track the position of a new deck correctly. |
| **3** | Quitting the Deck Editor after making a new deck. | Select the "Quit" button on the Deck Editor. | The program will ask the user if they want to save the deck. | To ensure that the program will ask the user to save a new deck. |
| **4** | Saving a new deck before quitting the Deck Creation window. | Selecting yes when asked if I would like to save the deck before I quit the Deck Creation window. | The program will ask for the name of the deck. | To ensure that the user can save their deck on the deck creation window before they return to the main menu. |
| **5** | Entering a name for a new deck. | When prompted for a name to save the deck with, entering "Created Deck" | The program will write this deck to an external file with this name. | To ensure that the user can save a new deck they have made. |
| **6** | Not saving a deck before quitting the Deck Creation window. | Selecting no when asked if I would like to save the deck before I quit the Deck Creation window. | The program will return to the main menu without writing the deck to an external file, and openLbl will display "No Open Deck". | To ensure that the user can return to the main menu without saving the deck they are working on. |
| **7** | Opening a deck to be edited. | Open "Test Deck" in the Main Menu, then selecting the "Edit" button. | The Deck Editor will be opened, and both text boxes will be blank. | To ensure that the program can display a new card to be added to the deck. |
| **8** | Displaying the number of the card being viewed in the deck that is being edited. | Open the deck editor to edit the deck "Test deck" | At the top of the Deck Editor window the position label will display "Card 6". | To ensure that the program can correctly display which card is being edited. |
| **9** | Using the back arrow to go to the previous card. | Open the Deck Editor to edit the deck "Test Deck", and then press the back arrow to move from the sixth card to the fifth card. | The question text box will display "What is a method?" And the answer text box will display "A function which an object-oriented class can execute". The red radio button will be displayed. | To ensure that the program can display the previous cards so that the user can edit them. |
| **10** | Using the back arrow to go to the previous card. | Open the Deck Editor to edit the deck "Test Deck", and then press the back arrow to move from the sixth card to the fifth card. | The card\_no Label will display "Card No. 5". | To ensure that the program can display the position correctly. |
| **11** | Using the back arrow to go the previous card. | Open the Deck Editor to edit the deck "Test Deck", and then press the back arrow to move from the sixth card to the fifth card. | The position variable will equal 5. | To ensure that the position variable is changed correctly while moving through the deck. |
| **12** | Using the forward arrow to go to the next card. | Press the forward arrow while the program is displaying the first card to move from the first card to the second. | The question text box will display "What is a parent class?" And the answer text box will display "A class which child classes inherit the properties of". The blue radio button will be displayed. | To ensure that the program can display the next cards so that the user can edit them. |
| **13** | Using the forward arrow to go to the next card. | Press the forward arrow while the program is displaying the first card to move from the first card to the second. | The card\_no label will display "Card No. 2" | To ensure that the program can display the position correctly. |
| **14** | Using the forward arrow to go to the next card. | Press the back arrow while the program is displaying the first card to move from the first card to the second. | The position variable will equal 2. | To ensure that the position variable is changed correctly while moving through the deck. |
| **15** | Using the forward arrow to go to the next card. | Use the radio button corresponding to the colour red to select this colour for the first card. | The program will change the stored colour for this card to red. | To ensure that the user can select the colour of each card to red. |
| **16** | Changing the colour for a card to default. | Use the radio button corresponding to the colour default to select this colour for the second card. | The program will change the stored colour for this card to default. | To ensure that the user can select the colour of each card to default. |
| **17** | Changing the colour for a card to green. | Use the radio button corresponding to the colour green to select this colour for the third card. | The program will change the stored colour for this card to green. | To ensure that the user can select the colour of each card to green. |
| **18** | Changing the colour for a card to blue. | Use the radio button corresponding to the colour blue to select this colour for the fourth card. | The program will change the stored colour for this card to blue. | To ensure that the user can select the colour of each card to blue. |
| **19** | Changing the colour for a card to yellow. | Use the radio button corresponding to the colour yellow to select this colour for the fifth card. | The program will change the stored colour for this card to yellow. | To ensure that the user can select the colour of each card to yellow. |
| **20** | Using the quit button to quit the deck creation. | Click on the button "Quit". | The program will display a message box asking if the user wants to save the deck they are currently working on. | To ensure that the user can quit the deck creation. |
| **21** | Saving an edited deck before quitting the Deck Creation window. | Selecting yes when asked if I would like to save the deck before I quit the Deck Editor window. | The window will then return to the main menu, where the deck which was edited will be displayed as the open deck. | To ensure that the user can save their deck on the deck creation window before they return to the main menu. |
| **22** | Not saving a deck before quitting the Deck Creation window. | Selecting no when asked if I would like to save the deck before I quit the Deck Editor window. | The program will return to the main menu without saving the deck, where the deck which was edited will be displayed as the open deck. | To ensure that the user can return to the main menu without saving the deck they are working on. |

## Whole Deck Revision Quiz Window

To test the Quiz, I used the same Test Deck which I used to test the Main Menu and Deck Editor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Displaying a question when the quiz is begun. | Begin the quiz by clicking the button "Start" on the Main Menu. | The question will be displayed with an empty textbox for the answer. | To ensure that the program displays the question correctly. |
| **2** | Displaying the amount of questions in the deck and the deck name. | Begin the quiz by clicking the button "Start" on the Main Menu after opening "Test Deck.csv". | The cardNoLbl will display "No. Of Cards: 5", and the deckNameLbl will display “Deck: Test Deck” | To ensure that the program can display how many cards there are in the deck, and which deck it is. |
| **3** | Revealing the answer to the question. | Pressing the button "Reveal Answer". | The program will display the corresponding answer to the question in the answer text box. | To ensure that the program can then reveal the corresponding answer to the question when requested. |
| **4** | Selecting Incorrect Answer | Clicking on the button "Incorrect Answer". | The question box will display a different question, and the answer text box will be blank. | To ensure that the program can continue to the next question after the user selects an incorrect answer. |
| **5** | Counting the amount of questions answered incorrectly. | Press the button "Incorrect" on the first question. | The variable incorrect should increase to 1. | To ensure the program can correctly count the amount of questions answered incorrectly. |
| **6** | Selecting Correct Answer | Clicking on the button "Correct Answer". | The question box will display a different question, and the answer text box will be blank. | To ensure that the program can continue to the next question after the user selects a correct answer. |
| **7** | Counting the amount of questions answered correctly. | Press the button "Correct" on the second question. | The variable incorrect should increase to 1. | To ensure the program can correctly count the amount of questions answered incorrectly. |
| **8** | Selecting Skip Question. | Clicking on the button "Skip Question". | The program should display a new question, and the question which was skipped should not be added to the answered array. | To ensure that the program can continue to the next question after the user selects skip, and that this question will be revisited as it is not stored among the questions which have been answered. |
| **9** | Not allowing the user to skip a question if they have revealed the answer. | Press the button "Reveal Answer" and then press the button "Skip". | The program should not do anything, and the question displayed should remain the same. | To ensure that the user cannot skip a question once the answer has been revealed. |
| **10** | Displaying a red card. | Begin the quiz and display card 1 in the Test deck. | The quiz window should be red, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **11** | Displaying a default card. | Begin the quiz and display card 2 in the Test deck. | The quiz window should be default, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **12** | Displaying a green card. | Begin the quiz and display card 3 in the Test deck. | The quiz window should be green, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **13** | Displaying a blue card. | Begin the quiz and display card 4 in the Test deck. | The quiz window should be blue, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **14** | Displaying a yellow card. | Begin the quiz and display card 5 in the Test deck. | The quiz window should be yellow, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **15** | Using the quit button to return to the Main Menu. | Click the button "Quit". | The user will be asked through a message box whether they would like to quit or not. | To ensure that the user can decide to quit the program from the Main Menu. |
| **16** | Choosing to quit using the Quit button on the Main Menu. | Select "Yes" after selecting the Quit button and being asked if I would like to quit. | The Quiz window will close, and the Main Menu will be shown. | To ensure that the user can exit from the quiz at any time. |
| **17** | Choosing not to quit after selecting the Quit button. | Selecting "No" after selecting the Quit button and being asked if I would like to quit. | The program will do nothing. | To ensure that the user can decide not to quit the quiz if they change their mind. |

## Timed Deck Revision Quiz Window

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Displaying a question when the quiz is begun. | Begin the quiz by clicking the button "Start" on the Main Menu. | The question will be displayed with an empty textbox for the answer. | To ensure that the program displays the question correctly. |
| **2** | Displaying the amount of questions in the deck. | Begin the quiz by clicking the button "Start" on the Main Menu after opening "Test Deck.csv". | The cardNoLbl will display "No. Of Cards: 5" | To ensure that the program can display how many cards there are in the deck. |
| **3** | Revealing the answer to the question. | Pressing the button "Reveal Answer". | The program will display the corresponding answer to the question in the answer text box. | To ensure that the program can then reveal the corresponding answer to the question when requested. |
| **4** | Selecting Incorrect Answer | Clicking on the button "Incorrect Answer". | The question box will display a different question, and the answer text box will be blank. | To ensure that the program can continue to the next question after the user selects an incorrect answer. |
| **5** | Counting the amount of questions answered incorrectly. | Press the button "Incorrect" on the first question. | The variable incorrect should increase to 1. | To ensure the program can correctly count the amount of questions answered incorrectly. |
| **6** | Selecting Correct Answer | Clicking on the button "Correct Answer". | The question box will display a different question, and the answer text box will be blank. | To ensure that the program can continue to the next question after the user selects a correct answer. |
| **7** | Counting the amount of questions answered correctly. | Press the button "Correct" on the second question. | The variable incorrect should increase to 1. | To ensure the program can correctly count the amount of questions answered incorrectly. |
| **8** | Selecting Skip Question. | Clicking on the button "Skip Question". | The program should display a new question, and the question which was skipped should not be added to the answered array. | To ensure that the program can continue to the next question after the user selects skip, and that this question will be revisited as it is not stored among the questions which have been answered. |
| **9** | Not allowing the user to skip a question if they have revealed the answer. | Press the button "Reveal Answer" and then press the button "Skip". | The program should not do anything, and the question displayed should remain the same. | To ensure that the user cannot skip a question once the answer has been revealed. |
| **10** | Displaying a red card. | Begin the quiz and display card 1 in the Test deck. | The quiz window should be red, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **11** | Displaying a default card. | Begin the quiz and display card 2 in the Test deck. | The quiz window should be default, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **12** | Displaying a green card. | Begin the quiz and display card 3 in the Test deck. | The quiz window should be green, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **13** | Displaying a blue card. | Begin the quiz and display card 4 in the Test deck. | The quiz window should be blue, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **14** | Displaying a yellow card. | Begin the quiz and display card 5 in the Test deck. | The quiz window should be yellow, the colour selected when the card was created. | To ensure that coloured cards are displayed properly. |
| **15** | Will end revision after the selected time has ended. | Enter 10 seconds as the selected time when beginning the quiz, then waiting 10 seconds. | The program should replace the quiz window with the completed quiz window. | To ensure that the user can specify a time to revise cards for, and that the program will end after this time. |
| **16** | Using the quit button to return to the Main Menu. | Click the button "Quit". | The user will be asked through a message box whether they would like to quit or not. | To ensure that the user can decide to quit the program from the Main Menu. |
| **17** | Choosing to quit using the Quit button on the Main Menu. | Select "Yes" after selecting the Quit button and being asked if I would like to quit. | The Quiz window will close, and the Main Menu will be shown. | To ensure that the user can exit from the quiz at any time. |
| **18** | Choosing not to quit after selecting the Quit button. | Selecting "No" after selecting the Quit button and being asked if I would like to quit. | The program will do nothing. | To ensure that the user can decide not to quit the quiz if they change their mind. |

## Completed Quiz Window

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Activity** | **Input** | **Expected Result** | **Reason for Testing** |
| **1** | Can display correct amount of right answers. | Take the quiz using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The completed window will appear at the end of the quiz displaying Correct: 3  Incorrect: 2 in 2 separate labels. | To ensure that the program can correctly count the amount of correct and incorrect answers. |
| **2** | Confirming that the user has achieved the high score if there are already high scores for this deck. | Remove the file "Test Deckscores.csv" from the scores file. Take the quiz using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The variable high\_score will be set to true. | To ensure that the program can save a high score and display to the user that they have a high score, if there are no high scores for this deck. |
| **3** | Writing the new high score to an external file if there are no other high scores. | Remove the file "Test Deckscores.csv" from the scores file. Take the quiz using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The program will create a file called "Test Deck.csv" in the scores folder which will contain the values "3, 2, 10, Sarah" on the first line and "0, 0, 0, blank" on the next two. | To ensure that the program can correctly write the user's high score to an external file if there are not already high scores for this deck. |
| **4** | Confirming that the user has achieved the high score if there are already high scores for this deck. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The variable high\_score will be set to true. | To ensure that the program can save a high score and display to the user that they have a high score if there are high scores for this deck. |
| **5** | Sorting the player's score with the other high scores. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The correct array will contain the numbers 5, 3, 3. | To ensure that the program can correctly sort a new high score into the correct place in the list. |
| **6** | Sorting the player's score with the other high scores. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The incorrect array will contain the numbers 0, 2, 2. | To ensure that the program can correctly sort a new high score into the correct place in the list. |
| **7** | Sorting the player's score with the other high scores. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The time array will contain the numbers 10, 15, 10. | To ensure that the program can correctly sort a new high score into the correct place in the list. |
| **8** | Sorting the player's score with the other high scores. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The names array will contain "top", "middle", "Sarah". | To ensure that the program can correctly sort a new high score into the correct place in the list. |
| **9** | Writing the newly sorted high scores to the external file. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | The file "Test Deckscores.csv" will have the arrays listed above written to file with each value separated by commas. | To ensure that the program can write the new high scores to a file. |
| **10** | Can display to the user that they have achieved a high score. | Add the Testscore.csv file used in the testing for High Scores. Take the quiz with whole deck revision, using the deck "Test Deck.csv" and answer 3 questions correctly and 2 incorrectly. | It will display on the Completed Quiz window in the highscoreLbl "Well done, you achieved a high score for this deck!" | To ensure that the program can display to the user that they have a high score. |
| **11** | Can recognise if the user has not achieved a high score. | While the file "Test Deckscores.csv" is in the scores folder and contains 3 high scores, take the quiz with whole deck revision, using Test Deck and answer all questions incorrectly. | The high\_score variable will be set to false. | To ensure that the program can recognise if the user has not achieved a high score. |
| **12** | Can display to the user that they have not achieved a high score. | While the file "Test Deckscores.csv" is in the scores folder and contains 3 high scores, take the quiz with whole deck revision, using Test Deck and answer all questions incorrectly. | The highscoreLbl will display "Sorry, you did not achieve a high score this time." | To ensure that the program can display to the user that they have not achieved a high score. |
| **13** | Using the Finish button to return to the Main Menu. | Click the Finish button. | The user will be asked through a message box whether they would like to return to the Main Menu. | To ensure that the user can return to the Main Menu after completing the quiz. |
| **14** | Choosing to return to the Main Menu. | Select "Yes" after selecting the Finish button and being asked if I would like to return to the Main Menu. | The Completed Quiz window will close, and the Main Menu window will be shown. | To ensure that the user can return to the Main Menu after completing the quiz. |
| **15** | Choosing not to quit after selecting the Finish button. | Selecting "No" after selecting the Quit button and being asked if I would like to quit. | The program will do nothing. | To ensure that the user can decide not to leave the Completed Quiz window if they change their mind. |

## Input Validation

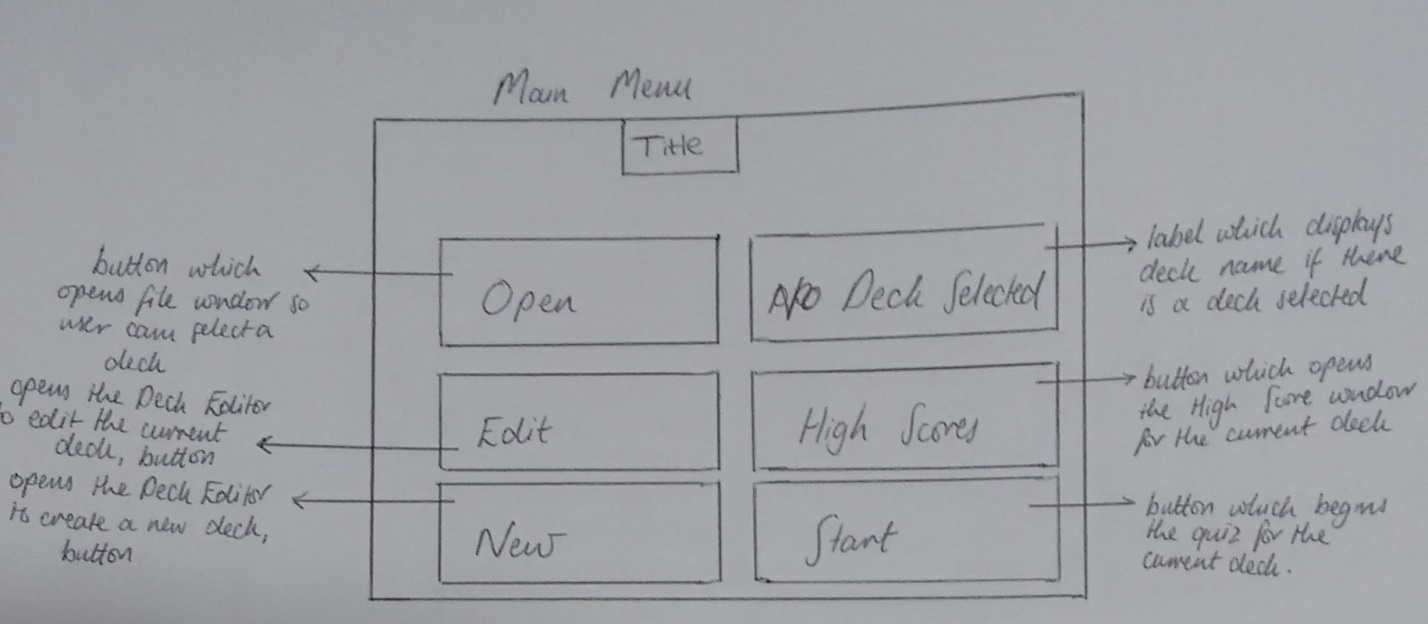
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Area** | **Input** | **Expected Result** | **Type of Data** | **Reason for Testing** | |
| **1.1** | Name Input | Sarah | Program will store this as user name. | Normal | To ensure that a name composed of letters can be put in correctly. |
| **1.2** |  | 123 | Program will display error message and allow the user to enter the name again. | Exceptional | To ensure that a name of numbers cannot be entered. |
| **1.3** |  | !?\* | Program will display error message and allow the user to enter the name again. | Exceptional | To ensure that a name of special characters cannot be entered. |
| **1.4** |  | Leave input blank. | Program will display error message and allow the user to enter the name again. | Exceptional | To ensure that the user must enter a name. |
| **2.1** | Question Input | qwerty | Program will store this as the question for this card and it will move to the next card. | Normal | To ensure that a question between 1 and 200 characters can be entered. |
| **2.2** |  | q | Program will store this as the question for this card and it will move to the next card. | Extreme | To ensure that a question of 1 character can be stored. |
| **2.3** |  | abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqr | Program will store this as the question for this card and it will move to the next card. | Extreme | To ensure that a question of 200 characters can be stored. |
| **2.4** |  | abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz | Program will display error message and allow the user to enter the question again. | Exceptional | To ensure that a question of over 200 characters cannot be stored. |
| **2.5** |  | Leave textbox blank | Program will display error message and allow the user to enter the question again. | Exceptional | To ensure that something must be entered for each question. |
| **3.1** | Answer Input | qwerty | Program will store this as the question for this card and it will move to the next card. | Normal | To ensure that an answer between 1 and 50 characters can be entered. |
| **3.2** |  | a | Program will store this as the question for this card and it will move to the next card. | Extreme | To ensure that an answer of 1 character can be stored. |
| **3.3** |  | abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqr | Program will store this as the question for this card and it will move to the next card. | Extreme | To ensure that an answer of 200 characters can be stored. |
| **3.4** |  | abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz | Program will display error message and allow the user to enter the answer again. | Exceptional | To ensure that an answer of over 200 characters cannot be stored. |
| **3.5** |  | Leave textbox blank | Program will display error message and allow the user to enter the answer again. | Exceptional | To ensure that something must be entered for each answer. |
| **4.1** | Time Selection | 10 | Program will begin a timed revision with the time limit of 10. | Normal | To ensure that the user can select a time between 1 and 300 seconds to revise the deck. |
| **4.2** |  | 1 | Program will begin a timed revision with the time limit of 1. | Extreme | To ensure that the user can select a time of 1 second. |
| **4.3** |  | 300 | Program will begin a timed revision with the time limit of 300. | Extreme | To ensure that the user can select a time of 300 seconds. |
| **4.4** |  | 0 | Program will display error message and allow the user to enter the time again. | Exceptional | To ensure that a time of 0 seconds cannot be entered. |
| **4.5** |  | 305 | Program will display error message and allow the user to enter the time again. | Exceptional | To ensure that a time of over 300 cannot be entered. |
| **4.6** |  | abc | Program will display error message and allow the user to enter the time again. | Exceptional | To ensure that the user can only enter numbers, not letters. |
| **4.7** |  | 1abc | Program will display error message and allow the user to enter the time again. | Exceptional | To ensure that the user can only enter numbers, not numbers and letters. |

User Interface Design

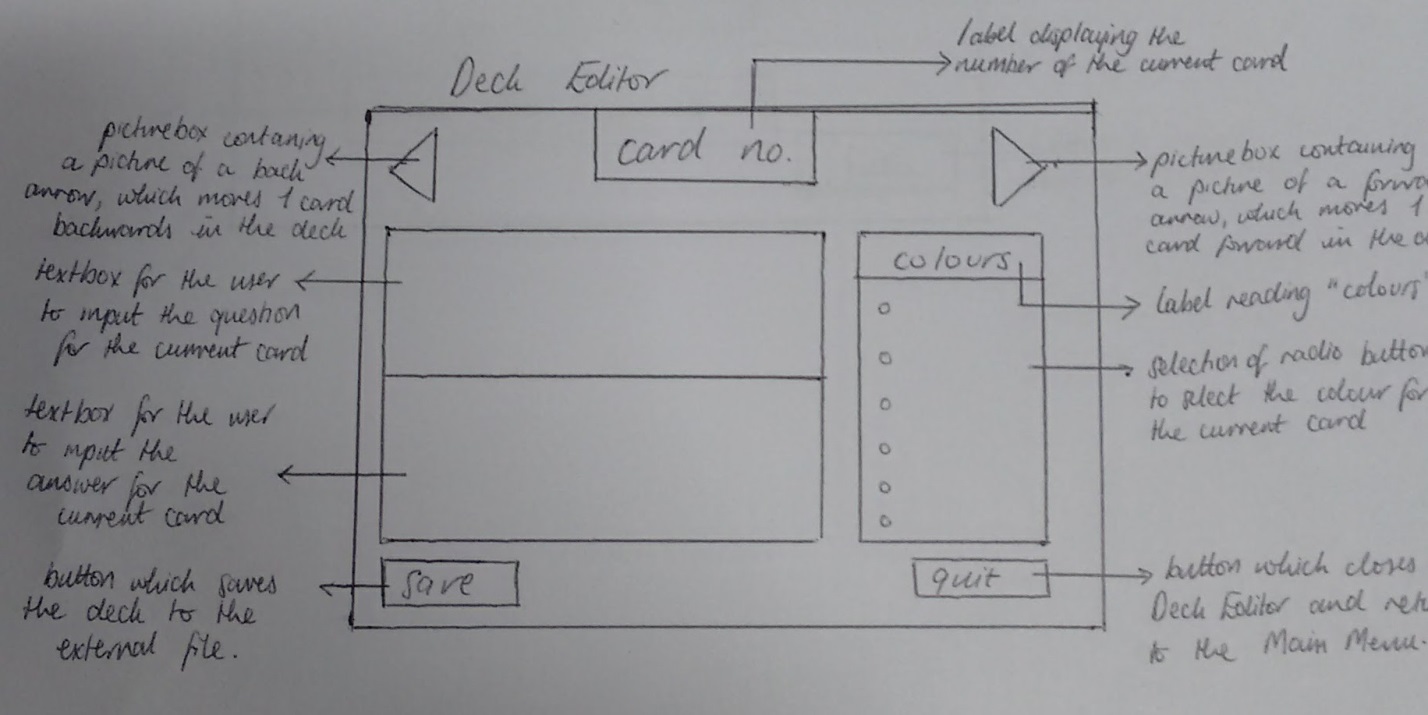
To plan the interface, I would use in my program, I created basic wireframe sketches of my initial ideas for the design. After this, I redrafted the initial designs, making some changes due to possible improvements I had noticed. I then implemented my designs, and made some final changes.

# Draft 1

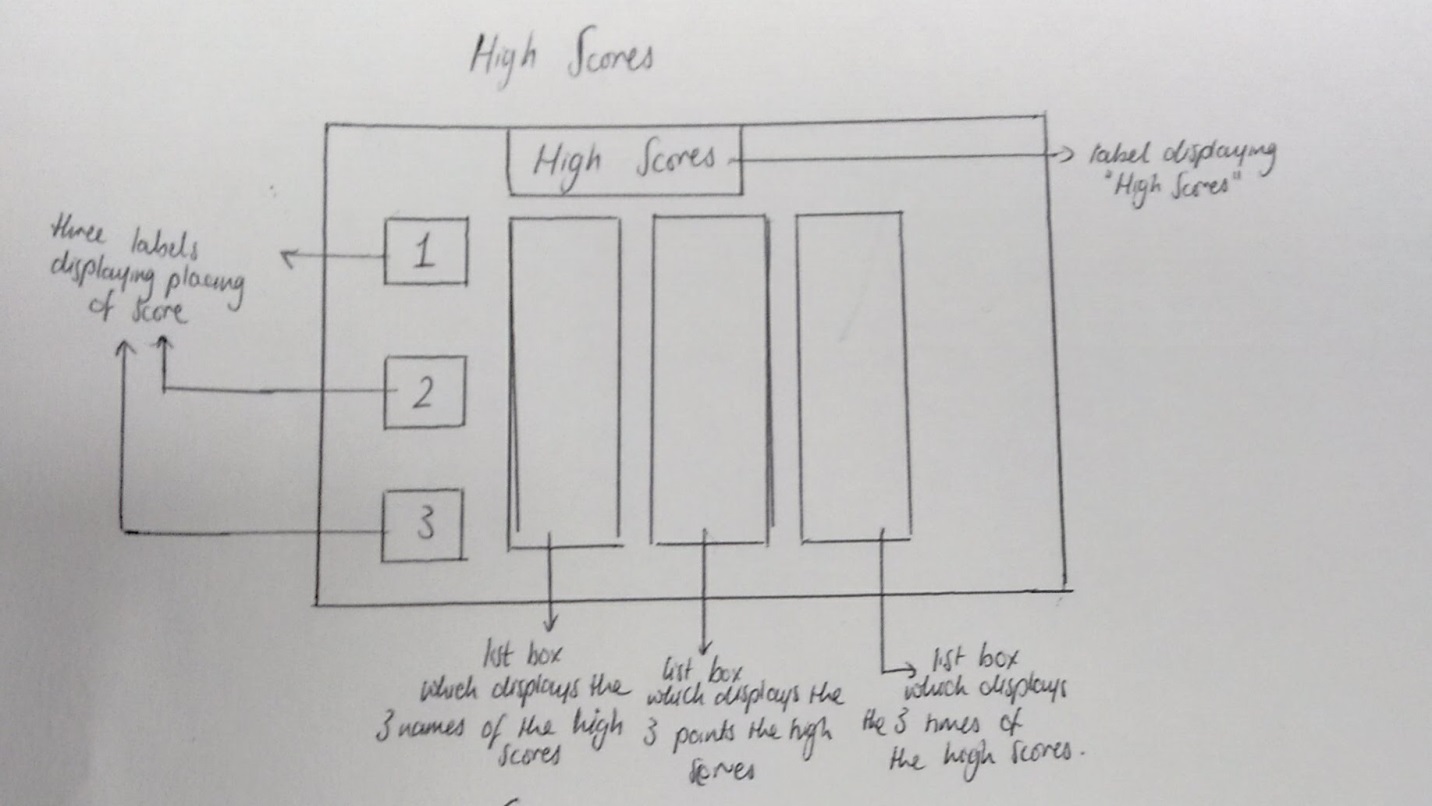
## Main Menu



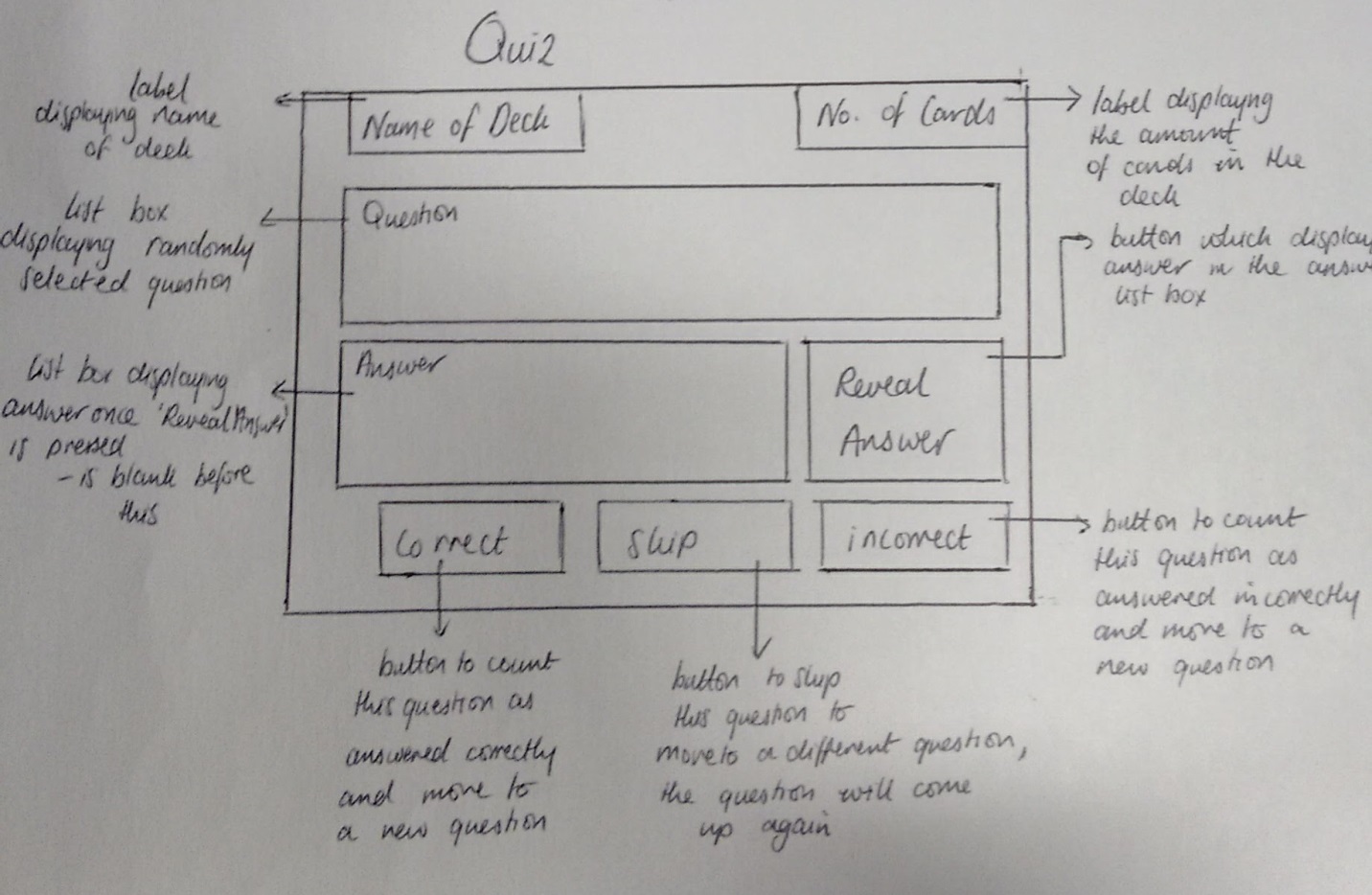
## Deck Editor



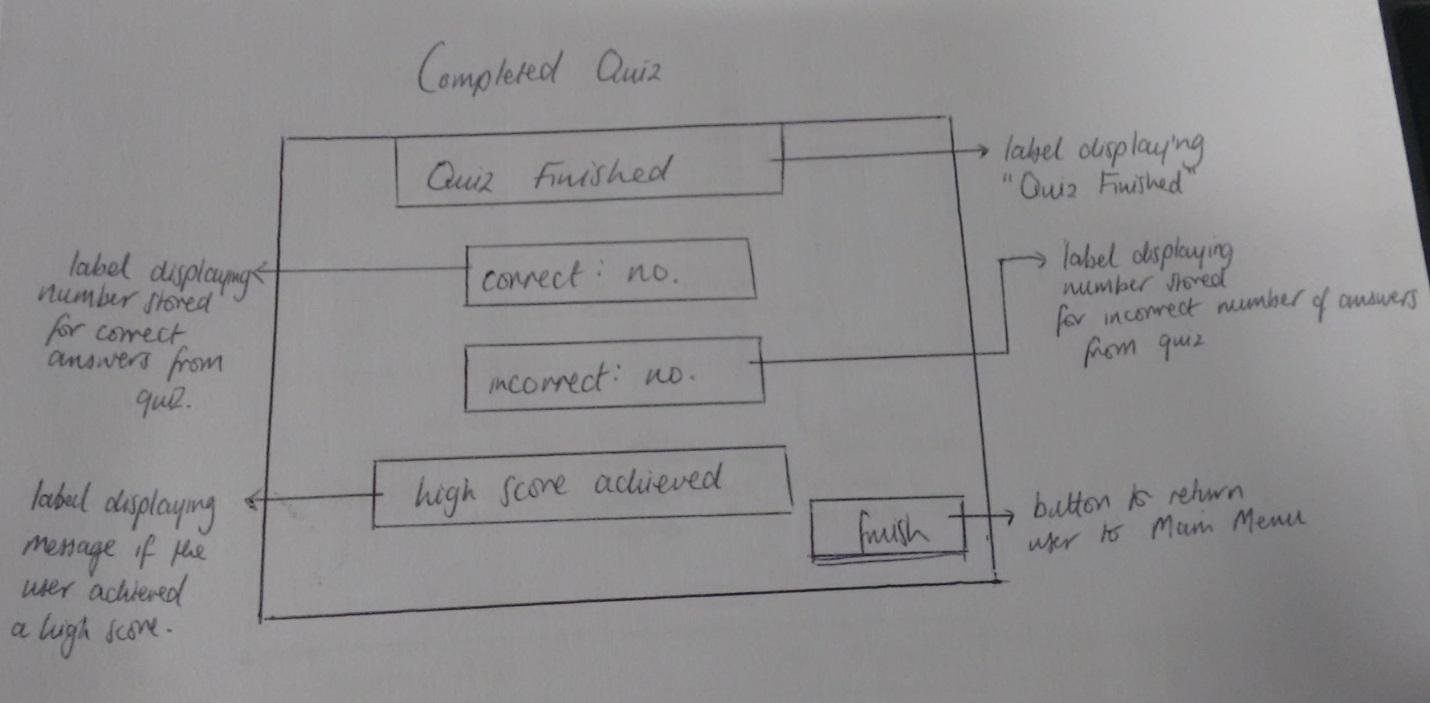
## High Scores



## Quiz

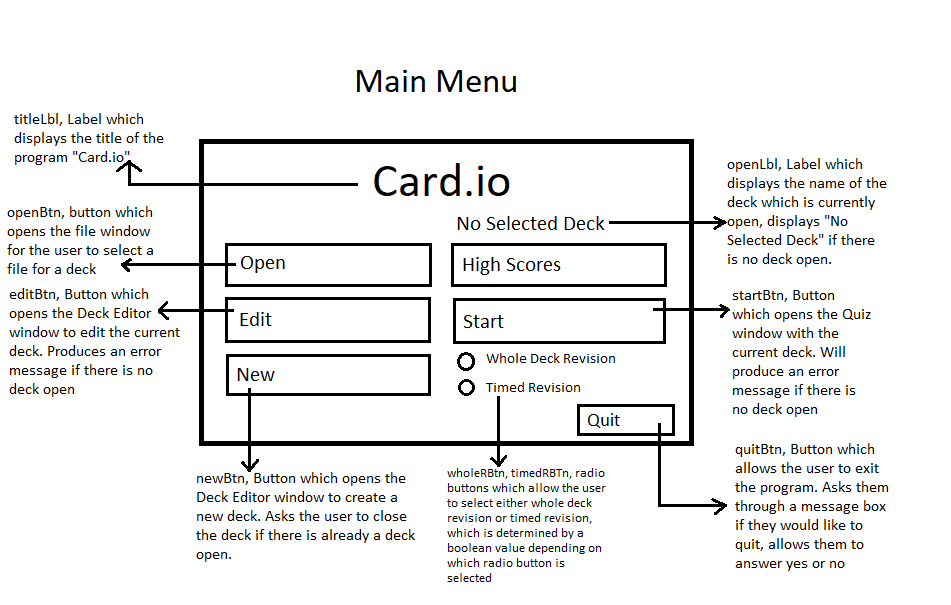


## Completed Quiz



# Draft 2

## Main Menu

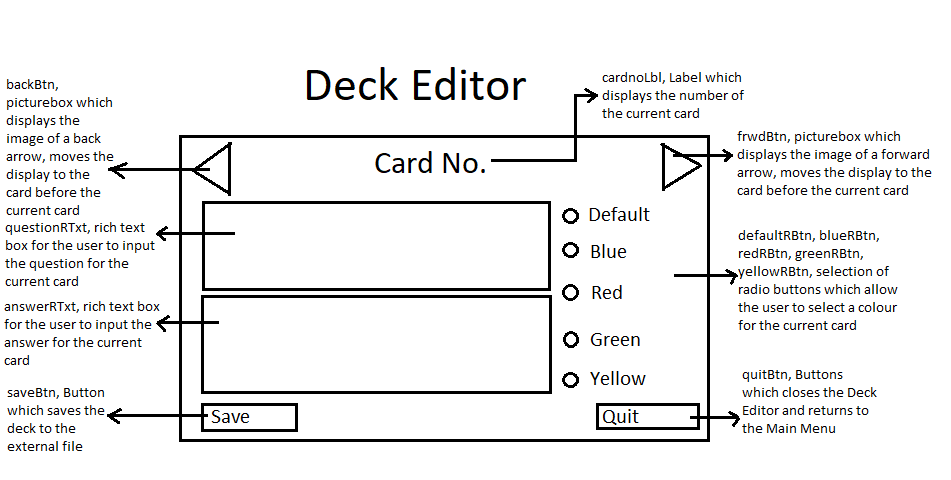


Since the first draft, openLbl has been moved up in the window to make space for the radio buttons which have been added. These allow the user to select the style of quiz they want. Radio buttons were used for this because it does not require any text input from the user, and it is more simple and intuitive for them to decide before they begin the quiz. Other than the radio buttons, buttons have been used for the other inputs as they are easy for users to understand and use and it is clear what each one does. A quit button has also been added to allow the user to exit the program easily.

The name will be entered on this window, and it will need to be validated to ensure it only consists of letters.

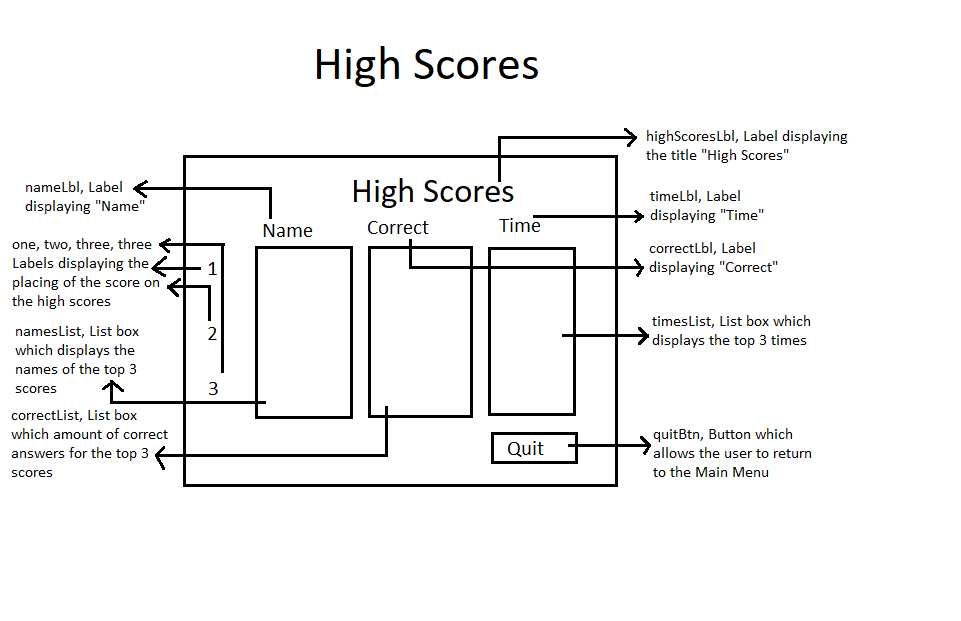
The time which is entered as the time limit for the quiz must be validated as well, to ensure that it is number, to avoid the program running into an error. It also must be between 1 and 300 so that the quiz is not too long.

## Deck Editor

The label displaying colours has been removed from the Deck Editor, as I decided it was not necessary, seeing as it is clear to users that the radio buttons are for selecting colours for the card, and the label is not needed to explain this. The label is used at the top of the form to make it clear to the user what point they are at in the deck, so that they can move through it easily, as they will know whether they are at the beginning or the end. Picture boxes are used for the forward and back buttons because this means arrow images can be used to make it clear to the user what the buttons are for. This makes the design intuitive. The radio buttons are used to input the colour of the card as this is easy for the user to input, and more simply than for instance text input. This means there is no validation needed, as the user can only choose from the options available. As well as this, there are many options for colours, so by using radio buttons it is clear to the user what colours are available. Buttons are used to save and quit so that the user can save the deck and quit easily.

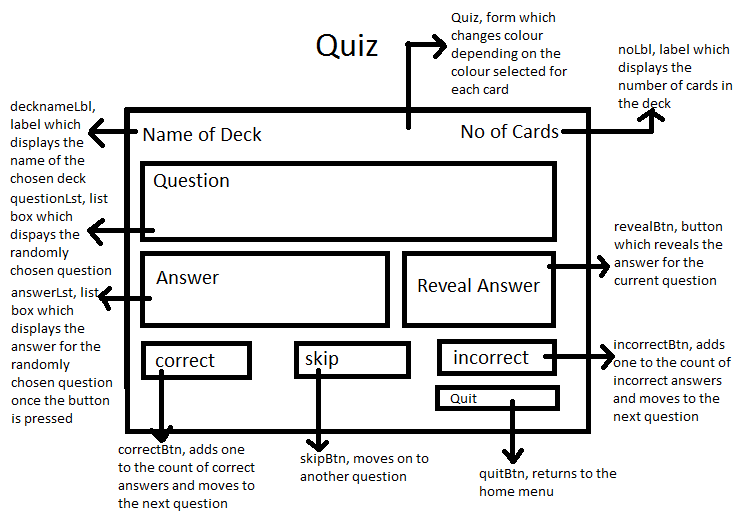
The text boxes’ inputs must be validated to ensure that text has been entered, so that there are values to enter as the questions and answers. The questions and answers must also be a length between 1 and 200 characters, so that they are not too long.

## High Scores



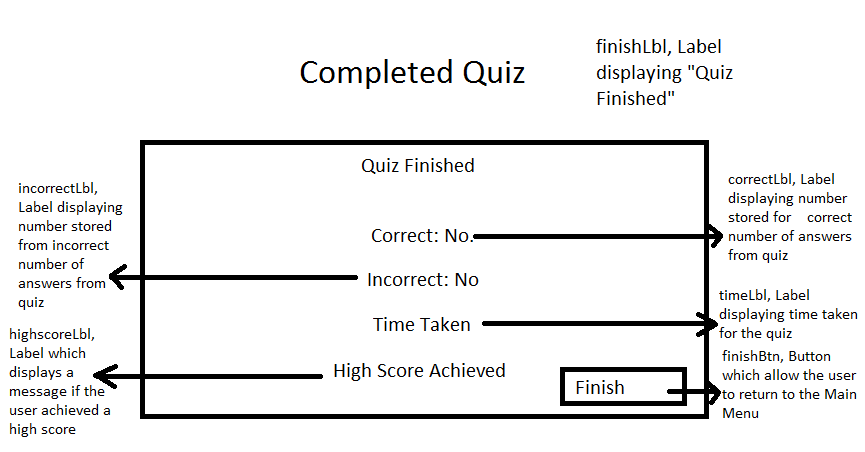
Instead of displaying the amount of points the user had earned, I decided to just display the amount of correct answers, because the amount of points would be the same as the amount of correct answers. Because of this, it is clearer to display the amount of correct answers as it is more obvious to the user what the number represents and how each score compares to the others. Labels were also added above each list box so that the user knows what each value represents. Without these labels it is not clear what information the list boxes contain. A button has also been added so that the user can return to the Main Menu after they have viewed the high scores. List boxes are used to display each score as it is easier to organise each category, and they are convenient for this purpose, as each array of values can be added to the list boxes. The alternatives, for instance using labels to display this information, would be more complicated to arrange and program. The labels at the side containing the numbers are used to display to the user what each place each score is in on the list of high scores.

## Quiz



After the first draft of the quiz window, I included a quit button so that the user can return to the Main Menu without completing the quiz. This makes the program easier to use as they can go back to change the deck or do a different kind of quiz if they change their mind. The questions and answers are displayed in list boxes so that the user cannot edit them. They do not need to type in either the question or the answer; they only need to read it, so list boxes fulfil the requirements needed for this. The labels are used to display the name and number of cards so that the user can easily see which questions they will be answering and how long the quiz will be. These are displayed in labels as this is a good way of displaying a simple piece of information on the form. Buttons have been used to allow the user to reveal the answer, answer the question and skip the question because this is the most intuitive way of inputting this information. It does not require the user to type any information which may then need to be validated.

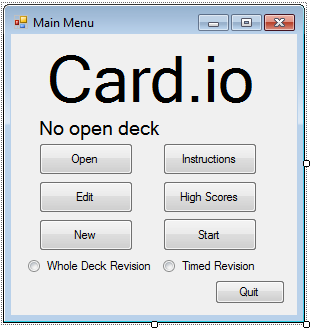
## Completed Quiz



The Completed Quiz window mainly includes labels which are used to display information to the user about how they performed in the quiz. Labels are used as the text does not need to be changed. I also added a button so that the user can move from the Completed Quiz window to the Main Menu, which means they can play through the quiz again, or edit a deck.

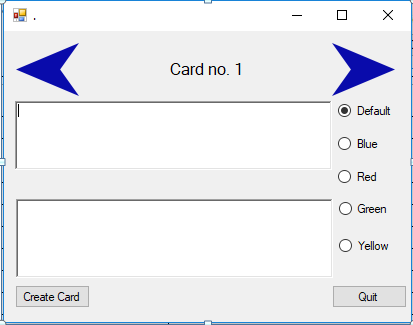
# Draft 3

## Main Menu



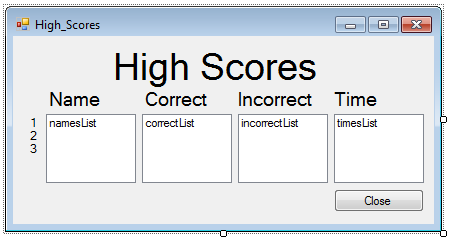
The label displaying the open deck was moved, to give more space for longer deck names. I also added a button for the instructions due to feedback which I received. The radio buttons were moved to make the design more aesthetically pleasing.

## Deck Editor



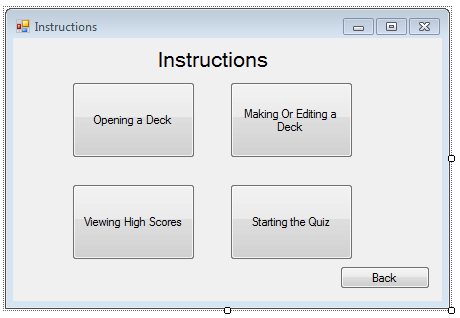
Due to changes that were made to the program between the second draft and the implementation, the save button has been replaced with a create card button.

## High Scores



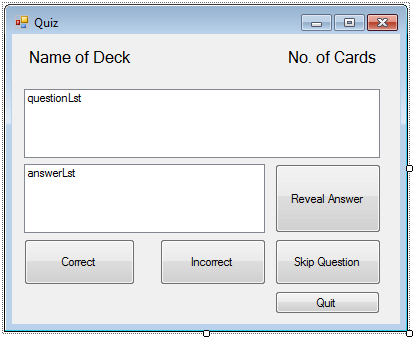
The high score window remained the same, except that I changed the quit button for a close button. The function is the same, but because the high scores window opens in addition to the main menu, this is clearer.

## Instructions



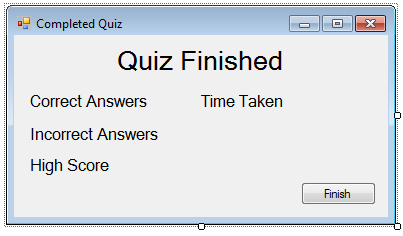
The instructions window is deliberately simple so that it is intuitive for a user to use. The buttons are used so that it is clear what sections of the program there are to learn more about.

## Quiz



The skip and incorrect buttons have been swapped, as it is more intuitive if the buttons to answer the questions are placed next to each other. The skip button now also says "Skip Question" as it is clearer to the user what it does.

## Completed Quiz



The labels for the answers were moved to the side because this means the time can also be moved, which makes the form neater. The high score label was moved to the side to leave more space for the message. The label now displays a message whether the user achieved a high score or not.

Structure Design

# Moving Between Forms

Deck Editor

Quiz

Main Menu

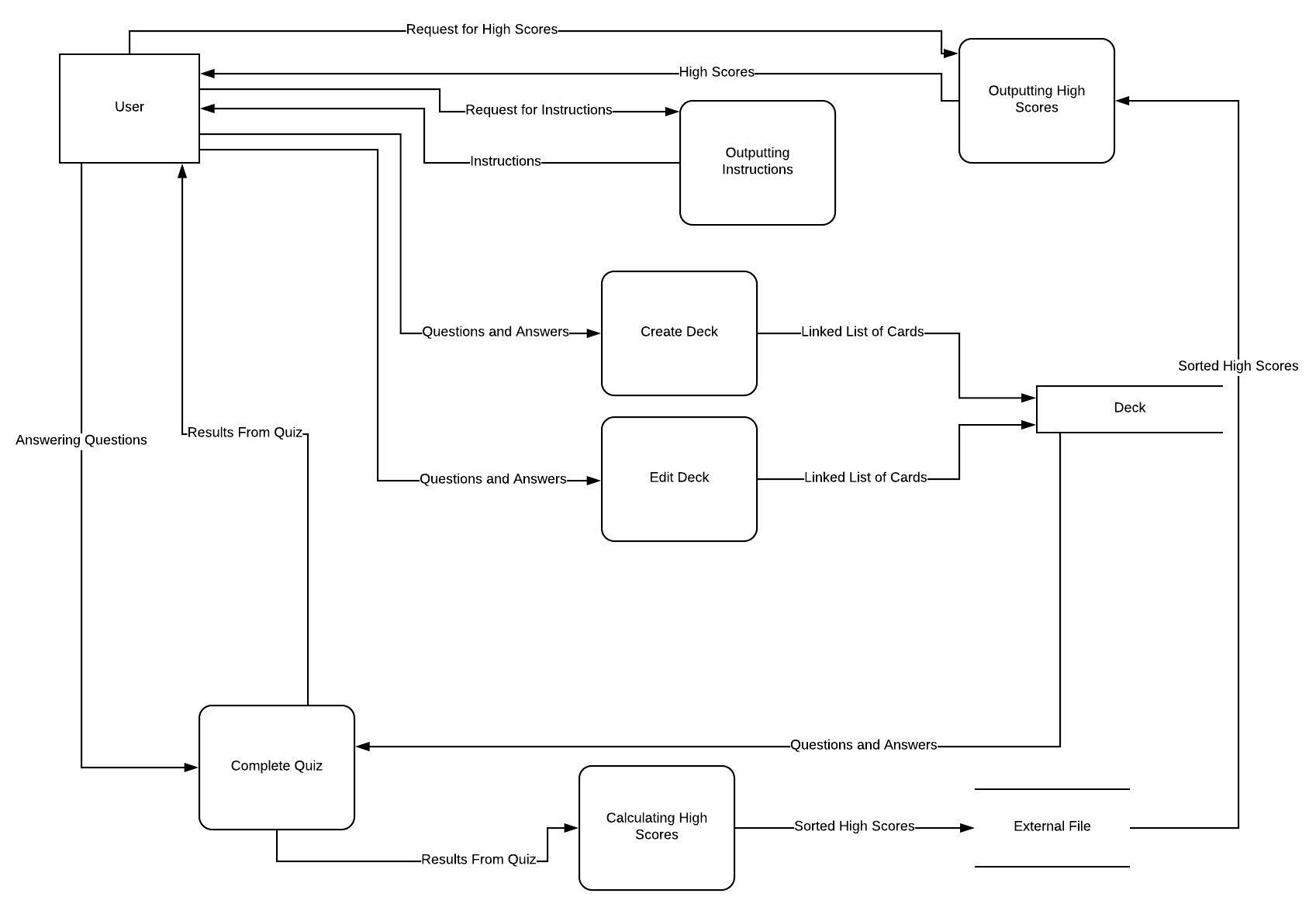
Completed Quiz

High Scores

Instructions

# Program Design

I used a data flow diagram to display the way data is used throughout the program. The basic processes that the program will use are displayed in this diagram.



# Pseudocode

## Main Menu

### Outline

1. Get user name from user
2. When "Open Deck" button is clicked open selected deck from file
3. When "New Deck" button is clicked open blank Deck Editor
4. When "High Scores" is clicked display High Scores window
5. When "Edit" button is clicked open the currently selected deck in the Deck Editor window
6. When the "Quiz" button is clicked open the Quiz window with the selected mode of quiz
7. When "Quit" button is clicked close the program

### On form load

1. GET username from user

### Open Deck button

1. IF deck is open THEN
   1. IF user closes current deck THEN
      1. CALL open deck
   2. END IF
2. ELSE IF deck is not open THEN
   1. CALL open deck
3. END IF

### Open deck procedure

1. OPEN file dialog
2. IF file extension is csv THEN
   1. SET filename to selected file name
   2. OPEN file
   3. FOR each line of file
      1. Read line
      2. SET question to first value
      3. SET answer to second value
      4. SET colour to third value
   4. END FOR
   5. CLOSE file
3. ELSE IF file extension not csv THEN
   1. Send error message to display
   2. Repeat until valid file selected
4. END IF

### New Deck button

1. IF deck is open THEN
   1. IF user closes current deck THEN
      1. HIDE Main Menu
      2. OPEN Deck Editor for new deck
   2. END IF
2. ELSE IF deck is not open THEN
   1. HIDE Main Menu
   2. OPEN Deck Editor window for new deck
3. END IF

### High Scores button

1. IF deck is open THEN
   1. HIDE Main Menu window
   2. OPEN High Scores window
2. ELSE IF deck is not open THEN
   1. DISPLAY error message
3. ELSE IF no high scores file THEN
   1. DISPLAY error message
4. END IF

### Edit button

1. IF deck is open THEN
   1. HIDE Main Menu
   2. OPEN Deck Editor for existing deck
2. ELSE IF deck is not open THEN
   1. DISPLAY error message
3. END IF

### Quiz button

1. IF quiz is timed THEN
   1. IF deck is open THEN
      1. REPEAT
         1. GET time limit from user
         2. IF time is not a number OR time is not between 0 and 300 THEN
            1. DISPLAY error message
            2. GET time limit from user
         3. END IF
      2. UNTIL time limit is valid
      3. OPEN Quiz for timed quiz
   2. END IF
2. ELSE IF quiz is not timed THEN
   1. IF deck is open THEN
      1. OPEN Quiz for whole quiz
   2. END IF
3. END IF

### Quit button

1. IF user wants to exit THEN
   1. CLOSE Main Menu
2. END IF

## Deck Editor

### Outline

1. Uses list of current deck to find attributes of deck
2. Can go back or forward
3. When "Quit" button clicked ask to save deck then return to Main Menu
4. Saves the deck to an external file
5. Enter questions and answers to textboxes
6. Use radio buttons to select colour for each card – default is default colour
7. Displays the number of card being viewed currently

### On load

1. IF deck is new THEN
2. SET position to 1
3. ELSE IF deck is not new THEN
4. SET position to deck length + 1

### Forward Button

1. IF position is not last card THEN
   1. CALL update
   2. SET current card to start
   3. FOR position
      1. SET current card to next card
   4. END FOR
   5. DISPLAY current card
   6. ADD 1 to position
   7. DISPLAY position
2. ELSE IF position is last card THEN
   1. CALL update
   2. CLEAR question text box
   3. CLEAR answer text box
   4. CREATE New Card
   5. SET current card to next card
   6. ADD 1 to position
   7. DISPLAY position

### Back Button

1. If position > 2 THEN
   1. CALL update
   2. FOR (position – 2)
      1. SET current card to next card
   3. END FOR
   4. DISPLAY current card
   5. SUBTRACT 1 from position
   6. DISPLAY position
2. ELSE IF position = 2 THEN
   1. CALL update
   2. SET current card to start
   3. DISPLAY current card
   4. SET position to 1
   5. DISPLAY position
3. END IF

### Quit Button

1. IF user wants to save deck THEN
   1. CALL save
2. END IF
3. CLOSE Deck Editor
4. OPEN Main Menu

### Create Card Button

1. IF question text box and answer text are not blank AND question <= 300 AND answer <= 300 THEN
   1. CALL new card
2. ELSE
   1. DISPLAY error message
3. END IF

### New Card Function

1. SET current question to question textbox
2. SET current answer to answer textbox
3. SET current colour to selected radio button
4. CLEAR question text box
5. CLEAR answer text box
6. CREATE New card
7. SET current card to next card
8. ADD 1 to deck length
9. ADD 1 to position
10. DISPLAY position

### Save Function

1. IF deck is new THEN
   1. GET name from user
2. ELSE
   1. SET name to deck name
3. END IF
4. IF file exists THEN
   1. SET current card to start
   2. OPEN file
   3. REPEAT
      1. WRITE question & answer & colour to file
      2. SET current card to next card
   4. UNTIL end of deck
   5. CLOSE file
5. ELSE
   1. CREATE file
   2. REPEAT
      1. WRITE question & answer & colour to file
      2. SET current card to next card
   3. UNTIL end of deck
   4. CLOSE file
6. END IF

## High Scores

### Outline

1. Reads high scores from external file
2. Displays scores to user
3. When "Quit" button clicked ask to save deck then return to Main Menu

### Quit Button

1. CLOSE High Scores

### On Load

1. CLEAR names list box
2. CLEAR correct list box
3. CLEAR incorrect list box
4. CLEAR times list box
5. OPEN high scores file
   1. FOR each line of file
      1. Read line
      2. SET correct to first value
      3. SET incorrect to second value
      4. SET time to third value
      5. Set names to fourth value
   2. END FOR
6. CLOSE high scores file
7. DISPLAY names
8. DISPLAY correct
9. DISPLAY incorrect
10. DISPLAY times

### Value Selected In List Box

1. Selected index in listbox = selected index in other listboxes

## Instructions Window

### Outline

1. When buttons are pressed hide buttons and displays
2. When back button pressed show buttons
3. When back button pressed again return to Main menu

### Back Button

1. IF buttons shown THEN
   1. CLOSE Instructions
2. ELSE
   1. CALL reset
   2. SHOW open button
   3. SHOW deck button
   4. SHOW scores buttons
   5. SHOW quiz button
3. END IF

### On Load

1. CALL reset
2. SHOW open button
3. SHOW deck button
4. SHOW scores button
5. SHOW high scores button

### Reset Function

1. HIDE open button
2. HIDE deck button
3. HIDE scores button
4. HIDE quiz button
5. HIDE open label
6. HIDE deck label
7. HIDE scores label
8. HIDE quiz label

### Open Button

1. CALL reset
2. SHOW open label

### Deck Button

1. CALL reset
2. SHOW deck label

### Scores Button

1. CALL reset
2. SHOW scores label

### Quiz Button

1. CALL reset
2. SHOW quiz button

## Quiz Window

### Outline

1. Randomly selects question to display
2. Checks that question has not been asked
3. Displays question when reveal button is pressed
4. Counts the amount that incorrect or correct buttons are pressed
5. Cannot skip a question once the answer is revealed
6. Skip button moves to new question
7. Quiz finishes when deck is finished, or time runs out
8. Quit button returns user to the Main Menu

### Quit Button

1. IF user wants to quit THEN
   1. CLOSE Quiz
   2. SHOW Main Menu
2. END IF

### On Load

1. FOR deck length
   1. SET answered array value to -1
2. END FOR
3. SET finished to false
4. SET incorrect to 0
5. SET correct to 0
6. SET time to 0
7. SET number to 0
8. CLEAR questions list box
9. START timer
10. DISPLAYT deck name
11. DISPLAY number of cards

### Skip Button

1. IF answer not revealed THEN
   1. CALL display
2. END IF

### Correct Button

1. ADD 1 to correct
2. ADD question number to answered array
3. ADD 1 to number
4. CALL display

### Incorrect Button

1. ADD 1 to incorrect
2. ADD question number to answered array
3. ADD 1 to number
4. CALL display

### Display Function

1. IF number = deck length THEN
   1. STOP timer
   2. CLOSE Quiz
   3. SHOW Completed Quiz
2. ELSE
   1. CALL select question
   2. SET form colour to current card colour
   3. CLEAR question list box
   4. DISPLAY question
   5. CLEAR answer list box
   6. SET answer to not revealed

### Select Question Function

1. REPEAT
   1. SET current to random number
   2. REPEAT
      1. IF current = answered(index) THEN
         1. Question is invalid
      2. END IF
      3. INCREMENT index by 1
   3. UNTIL question is invalid OR index = deck length
2. UNTIL question is valid

### Timer Tick

1. INCREMENT time by 1
2. IF quiz is timed THEN
   1. IF time > = time limit THEN
      1. STOP timer
      2. HIDE Quiz
      3. OPEN Completed Quiz
   2. END IF
3. END IF

## Completed Quiz Window

### Outline

1. Displays amount of questions correct and incorrect
2. Displays time taken
3. Checks if the user has achieved a high score
4. Displays if the user achieved a high score
5. If user has achieved a high score sort high scores
6. Write high scores to an external file
7. Finish button returns the user to the Main Menu

### Finish Button

1. IF user wants to quit THEN
   1. CLOSE Completed Quiz
   2. CLOSE Quiz
   3. SHOW Main Menu
2. END IF

### On Load

1. CALL high score check
2. CALL display
3. IF user achieved high score THEN
   1. CALL write
4. END IF

### High Score Check Function

1. IF quiz is timed THEN
   1. High score not achieved
2. ELSE IF high score file already exists THEN
   1. OPEN high score file
   2. FOR each line of file
      1. Read line
      2. SET correct to first value
      3. SET incorrect to second value
      4. SET time to third value
      5. SET names to fourth value
   3. END FOR
   4. IF user score > any high score THEN
      1. High score achieved
   5. END IF
   6. CLOSE file
3. ELSE
   1. High score achieved

### Display Function

1. DISPLAY correct
2. DISPLAY incorrect
3. DISPLAY time
4. IF high score achieved THEN
   1. DISPLAY "Well done, you achieved a high score for this deck!"
5. ELSE
   1. DISPLAY "Sorry, you did not achieve a high score this time."

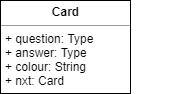
### Write Function

1. IF high score file exists THEN
   1. SET correct(3) to user’s correct value
   2. SET incorrect(3) to user’s incorrect value
   3. SET time(3) to user’s time
   4. SET names(3) to user’s name
   5. REPEAT
      1. SET swapped to false
      2. FOR counter FROM 1 to 3
         1. IF correct(counter – 1) < correct(counter) THEN
            1. swap(correct(counter), correct(counter - 1))
            2. swap(incorrect(counter), incorrect(counter - 1))
            3. swap(time(counter), time(counter - 1))
            4. swap(names(counter), names(counter - 1))
            5. swap(counter, (counter - 1))
            6. SET swapped to true
         2. END IF
   6. OPEN file
   7. FOR amount of high scores
      1. WRITE correct, incorrect, time and name to file
   8. CLOSE file
2. ELSE IF file does not exist THEN
   1. CREATE file
   2. WRITE user’s correct, incorrect, time and name to file
   3. WRITE blank value to file
   4. WRITE blank value to file
   5. CLOSE file
3. END IF

### Swap Function

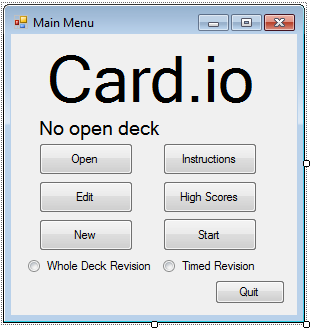
1. SET temp to first value
2. SET first value to second value
3. SET second value to temp

## Card Class



Implementation

# Main Menu



Public Class Main\_Menu

Public current As New Card 'keeps track of the current location in the linked list

Public start As New Card 'stores the head of the linked list

Dim deck\_open As Boolean = False 'stores whether there is a deck currently open

Public new\_deck As Boolean = True 'stores whether the deck is new or edited

Public deck As String 'stores the name of the deck

Public length As String 'stores the length of the deck

Public timed As Boolean = False 'stores whether the quiz is timed

Public limit As Integer 'stores the time limit of a timed quiz

Public userName As String 'stores the user's name

Private Sub Main\_Menu\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

'displays that there is no deck open

openLbl.Text = "No Open Deck"

'sets the default quiz setting to whole deck revision

wholeRBtn.Checked = True

End Sub

Private Sub openBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles openBtn.Click

If deck\_open = True Then

'asks user to confirm they want to close the current deck

If MsgBox("Would you like to close the currently open deck?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

'runs procedure to open deck

Call open\_deck(deck\_open)

'resets the values for the deck name, openLbl and deck\_open

deck = ""

openLbl.Text = "No Open Deck"

deck\_open = False

End If

Else

'runs procedure to open deck

Call open\_deck(deck\_open)

End If

End Sub

Private Sub open\_deck(ByRef deck\_open As Boolean)

Dim extension As String 'stores the extension of the selected file

Dim contents(1) As String 'stores the split contents of the file name

Dim filename As String 'stores the name of the selected file

Dim textline As String 'stores the text on each line being read

Dim temp(2) As String 'stores the split contents of the file

'sets the initial directory to display when the user selects a deck

openDeckdialog.InitialDirectory = "H:\S6\Computing\Project Files\decks"

'allows the user to select a file

openDeckdialog.ShowDialog()

filename = openDeckdialog.FileName

extension = System.IO.Path.GetExtension(filename)

'confirms the file extension is .csv

If extension = ".csv" Then

deck\_open = True

length = 0

deck = Dir(openDeckdialog.FileName)

contents = Split(deck, ".")

'sets name of deck to name of file without extension

deck = contents(0)

'displays name of deck

openLbl.Text = deck

current = start

Dim objTextfile As New System.IO.StreamReader(filename)

'reads in values from the file while there are values to read

Do While Not (objTextfile.Peek() = -1)

textline = objTextfile.ReadLine()

temp = Split(textline, ",")

'stores values in correct variables

current.question = temp(0)

current.answer = temp(1)

current.colour = temp(2)

length = length + 1

If Not (objTextfile.Peek() = -1) Then

'creates new card to store values while there are values to read

current.nxt = New Card

current = current.nxt

End If

Loop

objTextfile.Close()

objTextfile.Dispose()

current = start

Else

'error message if it is not a csv file

MsgBox("Invalid, please select another file.")

End If

End Sub

Private Sub quitBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles quitBtn.Click

Dim Response As DialogResult 'stores user's response from dialog box

'confirms user's choice

Response = MessageBox.Show("Do you really want to exit?", "", \_

MessageBoxButtons.YesNo, MessageBoxIcon.Question)

If Response = DialogResult.Yes Then

Me.Close()

End If

End Sub

Private Sub Main\_Menu\_Shown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shown

Dim input As String 'stores the input before it is validated

'confirms that the input only consists of letters

Do

input = InputBox("What is your name?", " ")

If IsLetter(input) = True Then

userName = input

Else

'error message if the name is not valid

MsgBox("Please enter a name which does not include numbers or other characters.")

End If

Loop Until IsLetter(input) = True

End Sub

Private Function IsLetter(ByVal input As String) As Boolean

Dim letter As Integer 'stores the separated value

Dim isString As Boolean 'stores whether the name is valid

'this confirms that each character in the string is a letter

For counter = 1 To Len(input)

letter = Asc(Mid(input, counter, 1))

If (letter >= 65 And letter <= 90) Or (letter >= 97 And letter <= 122) Then

isString = True

Else

isString = False

End If

Next

Return isString

End Function

Private Sub editBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles editBtn.Click

'if there is a deck open, it can be edited

If deck\_open = True Then

new\_deck = False

Hide()

Deck\_Editor.Show()

Else

MsgBox("Error: There is no deck open")

End If

End Sub

Private Sub scoresBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles scoresBtn.Click

'opens high scores window if there is a deck open and a valid scores file

If deck = Nothing Then

MsgBox("Error: No deck open.")

ElseIf Dir("H:\S6\Computing\Project Files\scores\" & deck & "score.csv") <> "" Then

High\_Scores.ShowDialog()

Else

MsgBox("Error: No high scores for this deck.")

End If

End Sub

Private Sub newBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles newBtn.Click

'opens deck editor to make a new deck if there is no deck open

If deck\_open = True Then

'allows the user to close the deck

If MsgBox("Would you like to close the currently open deck?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

deck = ""

openLbl.Text = "No Open Deck"

Hide()

Deck\_Editor.Show()

deck\_open = False

End If

Else

Hide()

Deck\_Editor.Show()

End If

End Sub

Private Sub startBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles startBtn.Click

Dim input As String 'stores input before it has been validated

'takes a time limit if the quiz is timed

If timed = True Then

If deck\_open = True Then

Do

input = InputBox("How many seconds would you like the quiz to be?")

'checks the time is valid

If IsNumeric(input) = True AndAlso (input > 0 And input <= 300) Then

limit = input

ElseIf IsNumeric(input) AndAlso (input <= 0 Or input > 300) Then

MsgBox("Please enter a number which is between 1 and 300.")

Else

MsgBox("Please enter a number.")

End If

Loop Until IsNumeric(input) = True AndAlso (input > 0 And input <= 300)

Hide()

Quiz.Show()

Else

MsgBox("Error: No deck open.")

End If

End If

'opens quiz without time limit if it is not timed

If timed = False Then

If deck\_open = True Then

Hide()

Quiz.Show()

Else

MsgBox("Error: No deck open.")

End If

End If

End Sub

Private Sub timedRBtn\_CheckedChanged(ByVal sender As Object, ByVal e As EventArgs) Handles timedRBtn.CheckedChanged

'sets timed to true if the time revision radio button is selected

timed = True

End Sub

Private Sub wholeRBtn\_CheckedChanged(ByVal sender As Object, ByVal e As EventArgs) Handles wholeRBtn.CheckedChanged

'sets timed to false if the whole deck revision radio button is selected

timed = False

End Sub

Private Sub instructionBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles instructionBtn.Click

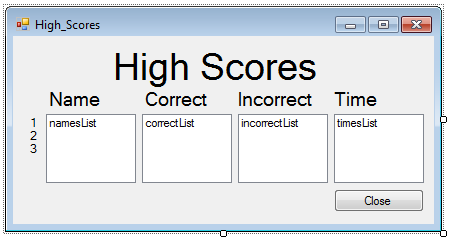
'displays instructions window

Instructions.ShowDialog()

End Sub

End Class

# High Scores



Public Class High\_Scores

Private Sub quitBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles quitBtn.Click

Close()

Dispose()

End Sub

Private Sub High\_Scores\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim textline As String 'stores the value for the line being read in

Dim temp(3) As String 'stores the split values of the contents of the file

Dim correct(2) As Integer 'stores the values of the correct answers

Dim incorrect(2) As Integer 'stores the values of the incorrect answers

Dim time(2) As Integer 'stores the values of the times

Dim names(2) As String 'stores the values of the names

'clears the display

namesList.Items.Clear()

correctList.Items.Clear()

incorrectList.Items.Clear()

timesList.Items.Clear()

'opens the score file for the deck

Dim objTextfile As New System.IO.StreamReader("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv", True)

'reads in the contents of the file

For counter = 0 To 2

textline = objTextfile.ReadLine()

temp = Split(textline, ",")

'stores the values in the variables

correct(counter) = CInt(temp(0))

incorrect(counter) = CInt(temp(1))

time(counter) = CInt(temp(2))

names(counter) = temp(3)

Next

objTextfile.Close()

objTextfile.Dispose()

'displays the values which have been read in

For counter = 0 To 2

namesList.Items.Add(names(counter))

correctList.Items.Add(correct(counter))

incorrectList.Items.Add(incorrect(counter))

timesList.Items.Add((time(counter)) & " seconds")

Next

End Sub

Private Sub namesList\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles namesList.SelectedIndexChanged

'updates the selection across each list box

incorrectList.SelectedIndex = namesList.SelectedIndex

correctList.SelectedIndex = namesList.SelectedIndex

timesList.SelectedIndex = namesList.SelectedIndex

End Sub

Private Sub correctList\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles correctList.SelectedIndexChanged

'updates the selection across each list box

incorrectList.SelectedIndex = correctList.SelectedIndex

namesList.SelectedIndex = correctList.SelectedIndex

timesList.SelectedIndex = correctList.SelectedIndex

End Sub

Private Sub incorrectList\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles incorrectList.SelectedIndexChanged

'updates the selection across each list box

incorrectList.SelectedIndex = incorrectList.SelectedIndex

namesList.SelectedIndex = incorrectList.SelectedIndex

timesList.SelectedIndex = incorrectList.SelectedIndex

End Sub

Private Sub timesList\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles timesList.SelectedIndexChanged

'updates the selection across each list box

incorrectList.SelectedIndex = timesList.SelectedIndex

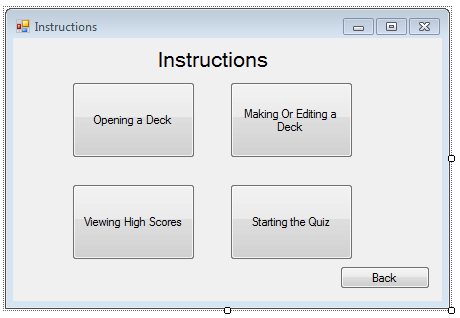
namesList.SelectedIndex = timesList.SelectedIndex

timesList.SelectedIndex = timesList.SelectedIndex

End Sub

End Class

# Instructions



Public Class Instructions

Dim open As Boolean = False 'stores whether there are some instructions open

Private Sub quitBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles backBtn.Click

'if there are no instructions open closes the window

If open = False Then

Close()

'if there are instructions open resets the window

Else

open = False

Call reset()

opnintrBtn.Show()

deckintrBtn.Show()

scoresintrBtn.Show()

quizintrBtn.Show()

End If

End Sub

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

'resets the window

Call reset()

'shows the correct instructions

opnintrLbl.Show()

'changes the title

titleLbl.Text = "Opening a Deck"

open = True

End Sub

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

'resets the window

Call reset()

'shows the correct instructions

deckintrLbl.Show()

'changes the title

titleLbl.Text = "Making or Editing a Deck"

open = True

End Sub

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

'resets the window

Call reset()

'shows the correct instructions

scoresintrLbl.Show()

'changes the title

titleLbl.Text = "High Scores"

open = True

End Sub

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

'resets the window

Call reset()

'shows the correct instructions

quizintrLbl.Show()

'changes the title

titleLbl.Text = "Starting the Quiz"

open = True

End Sub

Private Sub reset()

'resets the title label

titleLbl.Text = "Instructions"

'hides each button and label

opnintrBtn.Hide()

opnintrLbl.Hide()

deckintrBtn.Hide()

deckintrLbl.Hide()

scoresintrBtn.Hide()

scoresintrLbl.Hide()

quizintrBtn.Hide()

quizintrLbl.Hide()

End Sub

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

'resets the window

Call reset()

'shows the buttons

opnintrBtn.Show()

deckintrBtn.Show()

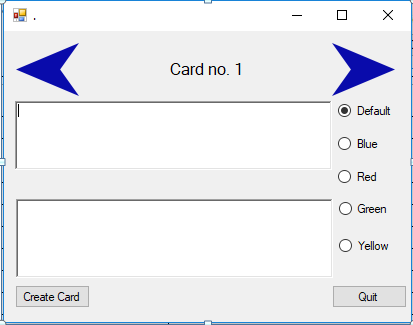
scoresintrBtn.Show()

quizintrBtn.Show()

End Sub

End Class

# Deck Editor



Public Class Deck\_Editor

Public position As Integer 'stores the position of card which is being displayed

Private Sub quitBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles quitBtn.Click

Dim Response As DialogResult 'stores the user's response

'checks if the user wants to save their deck

Response = MessageBox.Show("Do you want to save the deck before exiting?", "",

MessageBoxButtons.YesNo, MessageBoxIcon.Question)

If Response = DialogResult.Yes Then

'runs the procecure to write the deck to an external file

Call save()

End If

Close()

Dispose()

'returns to the main menu

Main\_Menu.Show()

End Sub

Private Sub createBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles createBtn.Click

'checks that the input is valid

If (questionRTxt.Text <> "" And answerRTxt.Text <> "") And (Len(questionRTxt.Text) <= 200 And Len(answerRTxt.Text) <= 200) Then

'calls the procedure to make a new card

Call new\_card()

ElseIf questionRTxt.Text = "" Or answerRTxt.Text = "" Then

'error message if the input is blank

MsgBox("Please enter a value for both the question and the answer.")

'error message if the question or answer is too long

Else

MsgBox("The question and answer must be less than 200 characters.")

End If

End Sub

Private Sub new\_card()

'creates a new card in the linked list

'sets the question and answer for the current card to the values in the textbox

Main\_Menu.current.question = questionRTxt.Text

Main\_Menu.current.answer = answerRTxt.Text

'sets the colour of the card to the selected radio button

If redRBtn.Checked = True Then

Main\_Menu.current.colour = "red"

ElseIf BlueRBtn.Checked = True Then

Main\_Menu.current.colour = "blue"

ElseIf GreenRBtn.Checked = True Then

Main\_Menu.current.colour = "green"

ElseIf yellowRBtn.Checked = True Then

Main\_Menu.current.colour = "yellow"

Else

Main\_Menu.current.colour = "default"

End If

'resets the text boxes

questionRTxt.Clear()

answerRTxt.Clear()

'creates a new card in the linked list to store the next card and moves to this card

Main\_Menu.current.nxt = New Card

Main\_Menu.current = Main\_Menu.current.nxt

'updates the length of the deck

Main\_Menu.length = Main\_Menu.length + 1

'moves the position forward in the deck and updates the display

position = position + 1

cardnoLbl.Text = ("Card no. " & position)

End Sub

Private Sub save()

Dim deck As String 'stores the deck name

'takes the name of the deck if it is a new deck

If Main\_Menu.new\_deck = True Then

deck = InputBox("What would you like to name this deck?", "Deck Name")

Else

'sets the name as the existing name of the deck

deck = Main\_Menu.deck

End If

'if the deck already exists - adds it onto the end of the pre-existing file

If Dir("H:\S6\Computing\Project Files\decks\" & deck & ".csv") <> "" Then

'opens the file

Dim objTextfile As New System.IO.StreamWriter("H:\S6\Computing\Project Files\decks\" & deck & ".csv", False)

Main\_Menu.current = Main\_Menu.start

'while there is another card to write to the file

Do While Not Main\_Menu.current.nxt Is Nothing

'writes each card to the file

objTextfile.WriteLine(Main\_Menu.current.question & "," & Main\_Menu.current.answer & "," & Main\_Menu.current.colour)

'moves through the linked list

Main\_Menu.current = Main\_Menu.current.nxt

Loop

objTextfile.Close()

objTextfile.Dispose()

Else

'if the deck doesn't already exist - makes a new one to save it to

Dim objTextfile As New System.IO.StreamWriter("H:\S6\Computing\Project Files\decks\" & deck & ".csv")

Main\_Menu.current = Main\_Menu.start

'while there is another card to write to the file

Do While Not Main\_Menu.current.nxt Is Nothing

'writes each card to the file

objTextfile.WriteLine(Main\_Menu.current.question & "," & Main\_Menu.current.answer & "," & Main\_Menu.current.colour)

'moves through the linked list

Main\_Menu.current = Main\_Menu.current.nxt

Loop

objTextfile.Close()

objTextfile.Dispose()

End If

End Sub

Private Sub Deck\_Editor\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Main\_Menu.current = Main\_Menu.start

defaultRBtn.Checked = True

'if the deck is being edited

If Main\_Menu.new\_deck = False Then

'displays a new card after the last pre-existing card

position = Main\_Menu.length + 1

Else

'resets position and length

position = 1

Main\_Menu.length = 0

End If

'displays the position

cardnoLbl.Text = ("Card no. " & position)

End Sub

Private Sub display()

'sets the text box values to the current question and answer

questionRTxt.Text = Main\_Menu.current.question

answerRTxt.Text = Main\_Menu.current.answer

'sets the selected radio button to the current card's colour

If Main\_Menu.current.colour = "red" Then

redRBtn.Checked = True

ElseIf Main\_Menu.current.colour = "blue" Then

BlueRBtn.Checked = True

ElseIf Main\_Menu.current.colour = "green" Then

GreenRBtn.Checked = True

ElseIf Main\_Menu.current.colour = "yellow" Then

yellowRBtn.Checked = True

Else

defaultRBtn.Checked = True

End If

End Sub

Private Sub frwdBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles frwdBtn.Click

'if the display is not at the end of the deck

If position < (Main\_Menu.length) Then

'updates any changes made to the card

Call update\_card()

Main\_Menu.current = Main\_Menu.start

'loops to the correct card

For counter = 1 To (position)

Main\_Menu.current = Main\_Menu.current.nxt

Next

'displays new card

Call display()

'updates position and display

position = position + 1

cardnoLbl.Text = ("Card no. " & position)

'if the displays is it at the end of the deck

ElseIf position = (Main\_Menu.length) Then

'updates any changes made to the card

Call update\_card()

'creates a new card

Main\_Menu.current.nxt = New Card

Main\_Menu.current = Main\_Menu.current.nxt

'clears the text boxes for the new card

questionRTxt.Clear()

answerRTxt.Clear()

'updates the position and display

position = position + 1

cardnoLbl.Text = ("Card no. " & position)

End If

End Sub

Private Sub backbtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles backBtn.Click

'if the display is not moving to the first card

If position > 2 Then

'updates any changes made to the card

Call update\_card()

Main\_Menu.current = Main\_Menu.start

'loops to the correct card

For counter = 1 To (position - 2)

Main\_Menu.current = Main\_Menu.current.nxt

Next

'displays the new card

Call display()

'updates the position and the display

position = position - 1

cardnoLbl.Text = ("Card no. " & position)

'if the display is moving to the first card

ElseIf position = 2 Then

'updates any changes made to the card

Call update\_card()

Main\_Menu.current = Main\_Menu.start

'displays the new card

Call display()

'updates the position and the display

position = 1

cardnoLbl.Text = ("Card no. " & position)

End If

End Sub

Private Sub update\_card()

'updates the current card without making a new card

'checks that there are values in the text boxes

If questionRTxt.Text <> "" And answerRTxt.Text <> "" Then

'updates the question and answer

Main\_Menu.current.question = questionRTxt.Text

Main\_Menu.current.answer = answerRTxt.Text

'updates the selected colour

If redRBtn.Checked = True Then

Main\_Menu.current.colour = "red"

ElseIf BlueRBtn.Checked = True Then

Main\_Menu.current.colour = "blue"

ElseIf GreenRBtn.Checked = True Then

Main\_Menu.current.colour = "green"

ElseIf yellowRBtn.Checked = True Then

Main\_Menu.current.colour = "yellow"

Else

Main\_Menu.current.colour = "default"

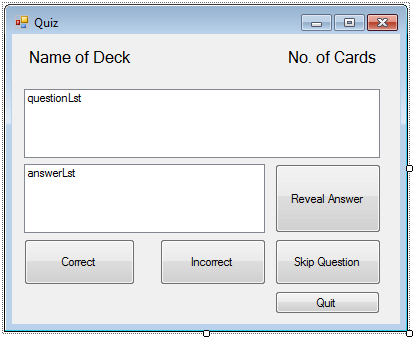
End If

End If

End Sub

End Class

# Quiz



Public Class Quiz

Dim question As Integer 'stores the value of the question in the array

Dim length As Integer = Main\_Menu.length 'takes the length of the deck from the main menu

Dim answered(length) As Integer 'stores which questions have been answered

Public correct As Integer 'stores the amount of correct answers

Public incorrect As Integer 'stores the amount of incorrect answers

Public time As Integer 'stores the time taken to complete the quiz

Dim no\_answered As Integer 'stores the amount of questions answered

Dim finished As Boolean = False 'stores whether the user has finished the quiz

Private Sub skipBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles skipBtn.Click

'checks if all the questions have been answered

If finished = False Then

Call display()

End If

End Sub

Private Sub quitBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles quitBtn.Click

Dim Response As DialogResult 'stores the user's response

'confirms the user's choice to quit

Response = MessageBox.Show("Are you sure you want to quit and return to the main menu? You will lose all of your progress.", "", \_

MessageBoxButtons.YesNo, MessageBoxIcon.Question)

If Response = DialogResult.Yes Then

Me.Close()

Me.Dispose()

'returns the user to the main menu

Main\_Menu.Show()

End If

End Sub

Private Sub Quiz\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

'empties answered questions array

For counter = 0 To length

answered(counter) = -1

Next

'resets variables to begin the quiz again

finished = False

correct = 0

incorrect = 0

time = 0

no\_answered = 0

questionLst.Items.Clear()

'starts the timer

Timer.Start()

'displays the deck name and amount of cards

deckNameLbl.Text = ("Deck: " & Main\_Menu.deck)

cardNoLbl.Text = ("No. of cards: " & length)

'displays the first card

Call display()

End Sub

Private Function select\_q() As Integer

Dim current As Integer 'stores the current chosen question

Dim invalid As Boolean = False 'stores whether the question is a valid question

Dim counter As Integer 'stores the times the loop has executed

'selects a new question

Do

Randomize()

'selects a random number of index for the array of questions

current = (((length - 1) \* Rnd()) + 1)

counter = 0

invalid = False

'checks that this question has not been asked

Do

If current = answered(counter) Then

invalid = True

End If

counter = counter + 1

Loop Until invalid = True Or counter = length

'loops until the question is valid

Loop Until invalid = False

Return current

End Function

Private Sub display()

'checks that the quiz is not finished

If no\_answered = length Then

Timer.Stop()

Me.Hide()

Completed\_Quiz.Show()

Else

'selects a question to display

question = select\_q()

'moves through linked list to the selected question

Main\_Menu.current = Main\_Menu.start

If question <> 1 Then

For counter = 1 To (question - 1)

Main\_Menu.current = Main\_Menu.current.nxt

Next

End If

'changes the form to the selected colour

If Main\_Menu.current.colour = "red" Then

Me.BackColor = Color.Red

ElseIf Main\_Menu.current.colour = "blue" Then

Me.BackColor = Color.Blue

ElseIf Main\_Menu.current.colour = "green" Then

Me.BackColor = Color.Green

ElseIf Main\_Menu.current.colour = "yellow" Then

Me.BackColor = Color.Yellow

Else

Me.BackColor = Color.FromKnownColor(KnownColor.Control)

End If

'displays the question and clears the answer

questionLst.Items.Clear()

questionLst.Items.Add(Main\_Menu.current.question)

answerLst.Items.Clear()

'sets question to unfinished

finished = False

End If

End Sub

Private Sub rightBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles rightBtn.Click

'adds 1 to count of correct answers

correct = correct + 1

'adds question to array of answered questions

answered(no\_answered) = question

'adds 1 to the count of answered questions

no\_answered = no\_answered + 1

'displays new question

Call display()

End Sub

Private Sub wrongBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles wrongBtn.Click

'adds 1 to count of incorrect answers

incorrect = incorrect + 1

'adds question to array of answered questions

answered(no\_answered) = question

'adds 1 to count of answered questions

no\_answered = no\_answered + 1

'displays new question

Call display()

End Sub

Private Sub revealBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles revealBtn.Click

'reveals the answer if it hasn't been revealed

If finished = False Then

answerLst.Items.Clear()

answerLst.Items.Add(Main\_Menu.current.answer)

End If

'sets the question to finished so it can't be skipped

finished = True

End Sub

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

'adds 1 to the time count

time = time + 1

'checks if the time limit has been reached

If Main\_Menu.timed = True Then

'ends the quiz if the limit has been reached

If time >= Main\_Menu.limit Then

Timer.Stop()

Me.Hide()

Completed\_Quiz.Show()

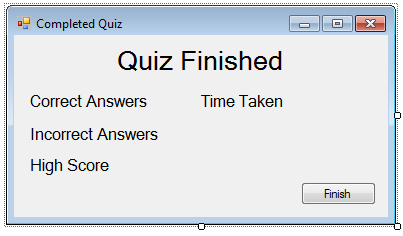
End If

End If

End Sub

End Class

# Completed Quiz



Public Class Completed\_Quiz

Dim correct(3) As Integer 'stores each player's amount of correct answers

Dim incorrect(3) As Integer 'stores each player's amount of incorrect answers

Dim time(3) As Integer 'stores each player's times

Dim names(3) As String 'stores each player's names

Private Sub finishBtn\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles finishBtn.Click

Dim Response As DialogResult 'stores the user's response

'confirms the user's decision to return to the main menu

Response = MessageBox.Show("Return to main menu?", "", \_

MessageBoxButtons.YesNo, MessageBoxIcon.Question)

If Response = DialogResult.Yes Then

'returns to the main menu

Me.Close()

Main\_Menu.Show()

Quiz.Close()

End If

End Sub

Private Sub Completed\_Quiz\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim high\_score As Boolean 'stores whether the user achieved a high score

'checks if the user got a high score

high\_score = score()

'displays the user's results

Call display(high\_score)

'if the user achieved a high score writes the new high scores to the file

If high\_score = True Then

Call write\_score()

End If

End Sub

Private Function score() As Boolean

Dim high\_score As Boolean = False 'stores if the user achieved a high score

Dim textline As String 'stores the text for the line being read

Dim temp(3) As String 'stores the split values of the file being read

'the user does not achieve a high score if the quiz is timed

If Main\_Menu.timed = True Then

high\_score = False

'if the score file exists checks the user's score against the other high scores

ElseIf Dir("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv") <> "" Then

Dim objTextfile As New System.IO.StreamReader("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv") 'opens the score file

Main\_Menu.current = Main\_Menu.start

'reads in the values stored in the high score file

For counter = 0 To 2

textline = objTextfile.ReadLine()

temp = Split(textline, ",")

'stores the values in variables

correct(counter) = CInt(temp(0))

incorrect(counter) = CInt(temp(1))

time(counter) = CInt(temp(2))

names(counter) = temp(3)

Next

'checks if the user's scores is higher than any of the high scores

For counter = 0 To 2

If Quiz.correct > correct(counter) Then

'if their score is higher they have achieved a high score

high\_score = True

End If

Next

objTextfile.Close()

objTextfile.Dispose()

Else

'if the deck doesn't already exist

high\_score = True

End If

Return high\_score

End Function

Private Sub display(ByVal high\_score As Boolean)

'displays the amount of correct and incorrect answers and time taken

correctLbl.Text = ("Correct Answers: " & Quiz.correct)

incorrectLbl.Text = ("Incorrect Answers: " & Quiz.incorrect)

timeLbl.Text = (Quiz.time & " seconds taken")

'displays a message telling the user if they achieved a high score or not

If high\_score = True Then

highscoreLbl.Text = "Well done, you achieved a high score for this deck!"

Else

highscoreLbl.Text = "Sorry, you did not achieve a high score this time."

End If

End Sub

Private Sub write\_score()

Dim swapped As Boolean 'stores whether a swap has been made while sorting the scores

'checks if the file already exists

If Dir("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv", False) <> "" Then

'adds the user's scores to the array to be sorted

correct(3) = Quiz.correct

incorrect(3) = Quiz.incorrect

time(3) = Quiz.time

names(3) = Main\_Menu.userName

'sorts the information which has been entered

Do

swapped = False

For counter = 1 To 3

'swaps the values if they are in the wrong order

If correct(counter - 1) < correct(counter) Then

swap(correct(counter), correct(counter - 1))

swap(incorrect(counter), incorrect(counter - 1))

swap(time(counter), time(counter - 1))

swap(names(counter), names(counter - 1))

swap(counter, (counter - 1))

swapped = True

End If

Next

'loops until there are no swaps made - this means they are in the correct order

Loop Until swapped = False

Dim objTextfile As New System.IO.StreamWriter("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv", False)

'writes the highest 3 scores to the pre-existing file

For counter = 0 To 2

objTextfile.WriteLine(correct(counter) & "," & incorrect(counter) & "," & time(counter) & "," & names(counter))

Next

objTextfile.Close()

objTextfile.Dispose()

'if the scores file does not already exist

Else

'creates a scores file

Dim objTextfile As New System.IO.StreamWriter("H:\S6\Computing\Project Files\scores\" & Main\_Menu.deck & "score.csv")

'writes the user's score

objTextfile.WriteLine(Quiz.correct & "," & Quiz.incorrect & "," & Quiz.time & "," & Main\_Menu.userName)

'writes placeholder scores to the file

objTextfile.WriteLine("0" & "," & "0" & "," & "0" & "," & "blank")

objTextfile.WriteLine("0" & "," & "0" & "," & "0" & "," & "blank")

objTextfile.Close()

objTextfile.Dispose()

End If

End Sub

Private Sub swap(ByRef val1 As Object, ByRef val2 As Object)

'swaps the two values

Dim temp As Object 'stores a temporary value while it is being swapped

temp = val1

val1 = val2

val2 = temp

End Sub

End Class

# Card Class

Public Class Card

Public question As String 'stores the question for the current card

Public answer As String 'stores the answer for the current card

Public colour As String 'stores the colour for the current card

Public nxt As Card 'stores the next card in the linked list

Public Sub New()

Me.question = ""

Me.answer = ""

Me.colour = ""

'Me.nxt = nxt

End Sub

End Class

Testing

# Recording Results

## Main Menu

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshot of Input** | **Screenshots of Actual Results** |
| **1** | Entering the user’s name |  |
|  |  |  |
| **2** | Opening files to select a premade deck. |  |
|  |  |  |
| **3** | Selecting a valid deck file. |  |
|  |  |  |
| **4** | Selecting a valid deck file. |  |
|  |  |  |
| **5** | Selecting an invalid deck file. |  |
|  |  |  |
| **6** | Creating a new deck while there is no deck open. |  |
|  |  |  |
| **7** | Creating a new deck while there is no deck open. |  |
|  |  |  |
| **8** | Creating a new deck when a deck is already open. |  |
| **8** |  |  |
| **9** | Closing the deck which is currently open. |  |
|  |  |  |
| **10** | Not closing the deck which is currently open. |  |
|  |  |  |
| **11** | Opening a deck when a deck is already open. |  |
|  |  |  |
| **12** | Closing the deck which is currently open. |  |
|  |  |  |
| **13** | Not closing the deck which is currently open. |  |
|  |  |  |
| **14** | Selecting edit currently open deck. |  |
| **14** |  |  |
| **15** | Selecting edit when no deck is selected. |  |
|  |  |  |
| **16** | Viewing the high scores for the currently open deck. |  |
|  |  |  |
| **17** | Trying to open the High Scores window when there is no deck open. |  |
|  |  |  |
| **18** | Opening the High Scores window for a deck which does not have any high scores. |  |
|  |  |  |
| **19** | Opening the Instructions window. |  |
|  |  |  |
| **20** | Opening the quiz without a deck selected. |  |
|  |  |  |
| **21** | Choosing to study the whole deck. |  |
|  |  |  |
| **22** | Choosing to study cards in a certain amount of time. |  |
|  |  |  |
| **23** | Selecting a time to revise the cards for. |  |
|  |  |  |
| **24** | Using the quit button to exit the program. |  |
|  |  |  |
| **25** | Choosing to quit using the Quit button on the Main Menu. |  |
|  |  | Program closed. |
| **26** | Choosing not to quit after selecting the Quit button. |  |
| **26** |  |  |

## High Score Window

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Displaying the names of each high score. |  |
|  |  |  |
| **2** | Displaying the correct answers of each high score. |  |
|  |  |  |
| **3** | Displaying the incorrect answers of each high score. |  |
|  |  |  |
| **4** | Displaying the times of each high score. |  |
|  |  |  |
| **5** | Quitting the high score window to return to the Main Menu. |  |
|  |  |  |

## Instructions

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Displaying the four buttons when the form is shown. |  |
|  |  |  |
| **2** | Displaying the instructions for opening a deck. |  |
|  |  |  |
| **3** | Using the back button after viewing the instructions on opening a deck. |  |
|  |  |  |
| **4** | Displaying the instructions for making or editing a deck. |  |
|  |  |  |
| **5** | Using the back button after viewing the instructions on making or editing a deck. |  |
|  |  |  |
| **6** | Displaying the instructions for viewing high scores. |  |
|  |  |  |
| **7** | Using the back button after viewing the instructions on viewing high scores. |  |
|  |  |  |
| **8** | Displaying the instructions for starting the quiz. |  |
| **8** |  |  |
| **9** | Using the back button after viewing the instructions on starting the quiz. |  |
|  |  |  |
| **10** | Using the back button to return to the Main Menu. |  |
|  |  |  |

## Deck Editor Window

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Opening a new deck. |  |
|  |  |  |
| **2** | Opening a new deck. |  |
|  |  |  |
| **3** | Quitting the Deck Editor after making a new deck. |  |
|  |  |  |
| **4** | Saving a new deck before quitting the Deck Creation window |  |
|  |  |  |
| **5** | Entering a name for a new deck. |  |
|  |  |  |
| **6** | Not saving a deck before quitting the Deck Creation window. |  |
| **6** |  |  |
| **7** | Opening a deck to be edited. |  |
|  |  |  |
| **8** | Displaying the number of the card being viewed in the deck that is being edited. |  |
|  |  |  |
| **9** | Using the back arrow to go to the previous card. |  |
|  |  |  |
| **10** | Using the back arrow to go to the previous card. |  |
|  |  |  |
| **11** | Using the back arrow to go the previous card. |  |
|  |  |  |
| **12** | Using the forward arrow to go to the next card. |  |
|  |  |  |
| **13** | Using the forward arrow to go to the next card. |  |
|  |  |  |
| **14** | Using the forward arrow to go to the next card. |  |
|  |  |  |
| **15** | Using the forward arrow to go to the next card. |  |
|  |  |  |
| **16** | Changing the colour for a card to default. |  |
|  |  |  |
| **17** | Changing the colour for a card to green. |  |
|  |  |  |
| **18** | Changing the colour for a card to blue. |  |
|  |  |  |
| **19** | Changing the colour for a card to yellow. |  |
|  |  |  |
| **20** | Using the quit button to quit the deck creation. |  |
|  |  |  |
| **21** | Saving an edited deck before quitting the Deck Creation window. |  |
|  |  |  |
| **22** | Not saving a deck before quitting the Deck Creation window. |  |
|  |  |  |

## Quiz Window

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Displaying a question when the quiz is begun. |  |
| **1** |  |  |
| **2** | Displaying the amount of questions in the deck and the deck name. |  |
|  |  |  |
| **3** | Revealing the answer to the question. |  |
|  |  | Updated: |
| **4** | Selecting Incorrect Answer |  |
|  |  |  |
| **5** | Counting the amount of questions answered incorrectly. |  |
|  |  |  |
| **6** | Selecting Correct Answer |  |
|  |  |  |
| **7** | Counting the amount of questions answered correctly. |  |
|  |  |  |
| **8** | Selecting Skip Question. |  |
|  |  |  |
| **9** | Not allowing the user to skip a question if they have revealed the answer. |  |
|  |  |  |
| **10** | Displaying a red card. |  |
|  |  |  |
| **11** | Displaying a default card. |  |
|  |  |  |
| **12** | Displaying a green card. |  |
|  |  |  |
| **13** | Displaying a blue card. |  |
|  |  |  |
| **14** | Displaying a yellow card. |  |
|  |  |  |
| **15** | Using the quit button to return to the Main Menu. |  |
|  |  |  |
| **16** | Choosing to quit using the Quit button on the Main Menu. |  |
|  |  |  |
| **17** | Choosing not to quit after selecting the Quit button. |  |
|  |  |  |

During my testing, I found that the questions and answers would not display properly as the list box does not automatically separate the text onto a new line if it is too long. Because I don’t have enough time to write the code to split the string and display it on separate lines, I decided to enable the horizontal scroll bar. This means that the user can still see the whole question and answer. Although it is less convenient for the user, it is easier to implement this late in the project. I had not come across this problem previously because the deck I had been using to test my program during implementation had not had questions which were long enough for this to be a problem.

## Timed Quiz Window

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Displaying a question when the quiz is begun. |  |
|  |  |  |
| **2** | Displaying the amount of questions in the deck. |  |
| **2** |  |  |
| **3** | Revealing the answer to the question. |  |
|  |  |  |
| **4** | Selecting Incorrect Answer |  |
| **4** |  |  |
| **5** | Counting the amount of questions answered incorrectly. |  |
|  |  |  |
| **6** | Selecting Correct Answer |  |
|  |  |  |
| **7** | Counting the amount of questions answered correctly. |  |
|  |  |  |
| **8** | Selecting Skip Question. |  |
|  |  |  |
| **9** | Not allowing the user to skip a question if they have revealed the answer. |  |
|  |  |  |
| **10** | Displaying a red card. |  |
|  |  |  |
| **11** | Displaying a default card. |  |
|  |  |  |
| **12** | Displaying a green card. |  |
|  |  |  |
| **13** | Displaying a blue card. |  |
|  |  |  |
| **14** | Displaying a yellow card. |  |
|  |  |  |
| **15** | Will end revision after the selected time has ended. |  |
|  |  |  |
| **16** | Using the quit button to return to the Main Menu. |  |
|  |  |  |
| **17** | Choosing to quit using the Quit button on the Main Menu. |  |
|  |  |  |
| **18** | Choosing not to quit after selecting the Quit button. |  |
| **18** |  |  |

## Completed Quiz Window

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1** | Can display correct amount of right answers. |  |
|  |  |  |
| **2** | Confirming that the user has achieved the high score if there are already high scores for this deck. |  |
|  |  |  |
| **3** | Writing the new high score to an external file if there are no other high scores. |  |
|  |  |  |
| **4** | Confirming that the user has achieved the high score if there are already high scores for this deck. |  |
|  |  |  |
| **5** | Sorting the player's score with the other high scores. |  |
|  |  |  |
| **6** | Sorting the player's score with the other high scores. |  |
|  |  |  |
| **7** | Sorting the player's score with the other high scores. |  |
|  |  |  |
| **8** | Sorting the player's score with the other high scores. |  |
|  |  |  |
| **9** | Writing the newly sorted high scores to the external file. |  |
|  |  |  |
| **10** | Can display to the user that they have achieved a high score. |  |
| **10** |  |  |
| **11** | Can recognise if the user has not achieved a high score. |  |
|  |  |  |
| **12** | Can display to the user that they have not achieved a high score. |  |
|  |  |  |
| **13** | Using the Finish button to return to the Main Menu. |  |
|  |  |  |
| **14** | Choosing to return to the Main Menu. |  |
|  |  |  |
| **15** | Choosing not to quit after selecting the Finish button. |  |
|  |  |  |

## Input Validation Testing

|  |  |  |
| --- | --- | --- |
| **No.** | **Screenshots of Input** | **Screenshots of Actual Results** |
| **1.1** | Normal Name Input |  |
|  |  |  |
| **1.2** | Exceptional Name Input |  |
|  |  |  |
| **1.3** | Exceptional Name Input |  |
|  |  |  |
| **1.4** | Exceptional Name Input |  |
|  |  |  |
| **2.1** | Normal Question Input |  |
|  |  |  |
| **2.2** | Extreme Question Input |  |
|  |  |  |
| **2.3** | Extreme Question Input |  |
|  |  |  |
| **2.4** | Exceptional Question Input |  |
|  |  |  |
| **2.5** | Exceptional Question Input |  |
|  |  |  |
| **3.1** | Normal Answer Input |  |
|  |  |  |
| **3.2** | Extreme Answer Input |  |
|  |  |  |
| **3.3** | Extreme Answer Input |  |
|  |  |  |
| **3.4** | Exceptional Answer Input |  |
|  |  |  |
| **3.5** | Exceptional Answer Input |  |
|  |  |  |
| **4.1** | Normal Time Input |  |
|  |  |  |
| **4.2** | Extreme Time Input |  |
|  |  |  |
| **4.3** | Extreme Time Input |  |
|  |  |  |
| **4.4** | Exceptional Time Input |  |
|  |  |  |
| **4.5** | Exceptional Time Input |  |
|  |  |  |
| **4.6** | Exceptional Time Input |  |
|  |  |  |
| **4.7** | Exceptional Time Input |  |
|  |  |  |

# End User Testing

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 10 |
| **Enjoyable** | 9 |
| **Ease of Use Compared to Physical Flash Cards** | 9 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | yes, it didn’t count the right or wrong answers when I completed the quiz |
| **Any areas that you feel could be improved and what could be improved?** | nope |
| **Any additional comments?** | nope |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 8 |
| **Intuitive Interface** | 7 |
| **Enjoyable** | 8 |
| **Ease of Use Compared to Physical Flash Cards** | 5 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | It told me that when doing a timed quiz that I took no time. However, I set the time to 2 seconds so this was an error. |
| **Any areas that you feel could be improved and what could be improved?** | I cannot think of any additions that could be made |
| **Any additional comments?** | - |

During these tests the users experience issues due to the fact that the quiz window had been disposed of, which meant that the values were not displayed of properly. To fix this issue the quiz window is now hidden until the user has returned to the main menu after the quiz is completed, when the quiz window can be disposed of without causing any issues.

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 8 |
| **Enjoyable** | 10 |
| **Ease of Use Compared to Physical Flash Cards** | 8 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | no |
| **Any areas that you feel could be improved and what could be improved?** | no |
| **Any additional comments?** | no |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 7 |
| **Enjoyable** | 10 |
| **Ease of Use Compared to Physical Flash Cards** | 9 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | NO |
| **Any areas that you feel could be improved and what could be improved?** | NO |
| **Any additional comments?** | LOVED IT!!!! |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 8 |
| **Enjoyable** | 10 |
| **Ease of Use Compared to Physical Flash Cards** | 10 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | nah, it's pretty good |
| **Any areas that you feel could be improved and what could be improved?** | nah |
| **Any additional comments?** | more fun decks |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 10 |
| **Enjoyable** | 10 |
| **Ease of Use Compared to Physical Flash Cards** | 7 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | Nope |
| **Any areas that you feel could be improved and what could be improved?** | Nope |
| **Any additional comments?** | LOVED IT :D |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 8 |
| **Intuitive Interface** | 6 |
| **Enjoyable** | 9 |
| **Ease of Use Compared to Physical Flash Cards** | 6 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | nope |
| **Any areas that you feel could be improved and what could be improved?** | The instructions need to be clearer, it’s too long. If you number the instructions it’s easier to read. |
| **Any additional comments?** | Add misty |

To fix the suggestions that this user experienced, I separated the text in the instructions window so that it is easier to read. I also changed the title label so that it now updates and displays the section of instructions that the user is viewing. This will allow them to know at all times what they are looking at.

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 7 |
| **Intuitive Interface** | 8 |
| **Enjoyable** | 7 |
| **Ease of Use Compared to Physical Flash Cards** | 5 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | Don’t press tab or double click the decks, the program stops working |
| **Any areas that you feel could be improved and what could be improved?** | Fix the stuff above |
| **Any additional comments?** |  |

|  |  |
| --- | --- |
| **Category** | **Rating (1-10)** |
| **Ease of Use** | 9 |
| **Intuitive Interface** | 8 |
| **Enjoyable** | 10 |
| **Ease of Use Compared to Physical Flash Cards** | 10 |

|  |  |
| --- | --- |
| **Did you encounter any bugs or errors while using the program? If yes please specify what they were and when you encountered them.** | When a deck with return carriage was opened it crashed. |
| **Any areas that you feel could be improved and what could be improved?** | no |
| **Any additional comments?** | no |

These testers experienced issues due to the fact that tabbing or using the enter key can sometimes unintentionally input these commands into the values being saved, and the external files. This then means that the files cannot be read by the program, as they are not in a format which the program recognises. Although this is a significant issue within the program, I was not able to find a solution for this problem, however other than this issue due to the incorrectly formatted files, the program functioned completely correct during both tests.

Evaluation

# Fitness for Purpose

Overall, my project manages to meet the requirements I set at the beginning of the project. Various features have been added throughout the development, but the project still fulfils its purpose and is suitable to be used by those who I intended it to be used. The user is still able to use the program for each function which I specified at the beginning of the project, which means I have been successful in creating a solution which meets these requirements.

## Maintainability

To ensure that my solution is accessible to other programmers, or myself at a future date, I had made sure to design my program in a way which means it is easy to understand if you are not familiar with the system. This is in case any changes need to be made in the future, and the person making the changes does not have the same level of familiarity I do with the solution.

To guarantee that this is possible, I have made use of internal commentary to explain the various functions and attributes of the program to anyone who is reading the code. The function of each variable is detailed in internal commentary next to it so that anyone viewing the code can clearly see the purpose each variable has in the program. I have also aimed to name my variables something clear, which indicates what information it is storing and where it might be used. If this is not clear enough, then anyone reading the code can refer to the explanation of the variable in the internal commentary to confirm the function of the variable. This will make it easier for anyone reading the code to understand it, as they will be able to see where certain information is being stored and what sections of code it is being used in without having to work out its function themselves.

I have also used internal commentary to explain the sub routines of the program and what certain sections of code do, if I think it may be unclear. This allows the person reading the code to see what each procedure does, and if they have to edit the code, they can easily find the section which carries out the process they need to change. It will also make the code more maintainable, as the processes inside the sub routines are explained in more detail, which means that anyone who needs to change a function can easily understand which parts of each section of code carry out each function, and if there is a certain issue or area which needs to be changes, they can use the internal commentary to find the correct lines of code to apply the changes.

## Robustness

The program is able to deal able to deal with unexpected inputs, for instance if it expects an integer but a string is entered. It does this by validating the input before it continues the program, which means that it will not try to execute a function with an incorrect data type. This allows the program to be more robust, as it is not broken by the user inputting an unexpected data type, it simply allows the user to fix the incorrect input and continue using the program. However, the program is not as robust if the external files are not correctly formatted. The program will be able to continue if any of the expected external files such as high scores are not present in the correct folder, as it simply informs the user and will create a new file if necessary, but if the file is present but is not in the form expected by the program, this will cause it to stop functioning. This is because the procedures to read in the files do not contain processes if the information is not stored in the format which is expected. For instance, if the program tries to read in a deck but there is no value stored for the answer, this will cause a run-time error which will cause the program to stop. This error was experienced by some of the end user testers, who accidentally changed the layout of programs which the program could not deal with. To improve the robustness of the solution, I would need to include processes for the program to execute if it is reading a file which is not formatted as expected, which would allow it to continue to run after encountering an invalid file. This would allow users who may not be aware of this issue to use the program without accidentally selecting files which will result in the program stopping.

## Reliability

Throughout the testing, I examined and documented the outputs the program produced when given various inputs, to confirm that it would do what it is meant to do. It was able to create the expected output in most cases, and if it did not I was able to rectify the issues so that it provided the output which was expected. Because I have proven that the program is able to function as I expected to, I have confirmed that my solution is reliable. If a user operates the program as it is expected to be used the program will function as it was planned to, and will interact with the user correctly, as it is designed correctly.

## Efficiency

While sorting the high scores, instead of a bubble sort, a more efficient sort such as a quick sort would improve the efficiency of the solution. Although the sort efficiency is unlikely to make a significant difference due to the length of the array it is sorting, this would make the solution more efficient, as it would not have to pass through the entire loop each time it is making a swap, and repeating this process until the sort is completed.

The efficiency is also decreased because the program reads in the high scores file every time the quiz is completed to confirm whether they have achieved a high score. However overall this increases the efficiency as it means the program can simply compare the user’s score to the scores saved in the external file, instead of having to sort the score against these scores even if it is lower than all of them. If the program did not read in the file to check if the user had achieved a high score the alternative to this would still involve reading in the file eventually, but instead of just having to make some comparisons the program would be required to sort the user’s score to see if it was one of the high scores, which would decrease the efficiency as it would require the program to execute the sort function every time the quiz is completed. Instead of this, the program only executes the sort procedure when it is required, because it has already confirmed that the user has achieved a high score.

The efficiency of the procedure which selects a question could also be improved, as it does not have a way to guarantee that the question chosen has not already been asked, which means that it has to check each question it selects against each question which has already been asked in case it has been. This procedure is not as efficient as it could be, as it will check the selected question against the entire list of answered questions until either it reaches the end of the list or it finds the question has already been asked. This means that even if this array of answered questions is empty or only has one or two values it will still check each value in the array against the chosen question. As well as this, once all but a few questions have been asked, it is likely that the program will select many questions which have already been asked before it will find a question which has not been asked yet. This could be improved, as with the current method it requires the program to run the procedure to check the value each time, which means that if it selects many questions which have already been asked consecutively, it is executing a lot of unnecessary code.

Overall the solution is satisfactorily efficient, as it does not place any unnecessary strain on the system running it, and it is able to run without any noticeable delays, however there are some areas where the efficiency could be improved.

## Testing

Through my test plan which I had created earlier in the development process, I was able to identify potential flaws, such as the user not being able to see the question if it was too long. This was beneficial for me as I had not experienced this problem before, due to the fact that the deck I had been using to test the program before then only had short questions. I found that using the comprehensive testing method I had laid out allowed me to confirm that my solution was successful and that it had achieved all of the requirements I had specified. Because the tests were specifically chosen to ensure that the program met all of the requirements, completing these tests successfully meant that the solution had achieved all the functions it was meant to.

## End User Testing

Through the end user testing I was notified of various areas of my solution which could be improved, or of issues which I had not encountered so far during my final testing or my testing while implementing the program. For example, I had not come across the issue that was caused by unexpected inputs changing the formatting of the external file to a format that is not expected by the program. This is because the users will do different things when they use the program than I will, as I know exactly how it functions, whereas they do not know how it works, and they have to work out for themselves how to use it, which may lead to problems and errors. This is evident by the fact that their input caused an issue that I had not experienced before. This is useful, as it allows me to investigate the issues that are discovered further and attempt to rectify them. Without the inclusion of end user testing, these issues would not be found during development, and would only be discovered by real users, which would be after the release of the program, making it much harder to fix.

## Potential Improvements

Although my solution is successful in meeting the requirements which were specified at the beginning of the development, there are still areas of the design which could be improved if I had more time to develop the solution. For instance, when the user skips a card, the program finds a new card to display, but because the skipped card is not marked as answered, it is possible that the program will select the same question that the user just skipped. This is not a huge issue, as the user can simply select the skip button again and the program will select a different question, which will likely be a different question, but this could be changed to improve the convenience for the user. If I marked the skipped question so it cannot be skipped either, then the program would always choose a different question.

As well as this, if I were to continue the development of my solution, I would improve the robustness of the program by adding error trapping to my code, which would allow the program to continue functioning even if it runs into an error, for instance when reading in a deck or high scores file which is an unexpected format. This would allow me to put in place processes for the program to execute in the situation where the file is incorrectly formatted, which will otherwise stop the program from functioning, as it cannot read it correctly.

I could also include a function which would separate the questions and answers if they are too long to be displayed on one line in the list box, which would allow the user to see the whole text at the same time. This would make the program more convenient for the user, but unfortunately I was not able to complete this function because I did not have enough time before the project needed to be completed.

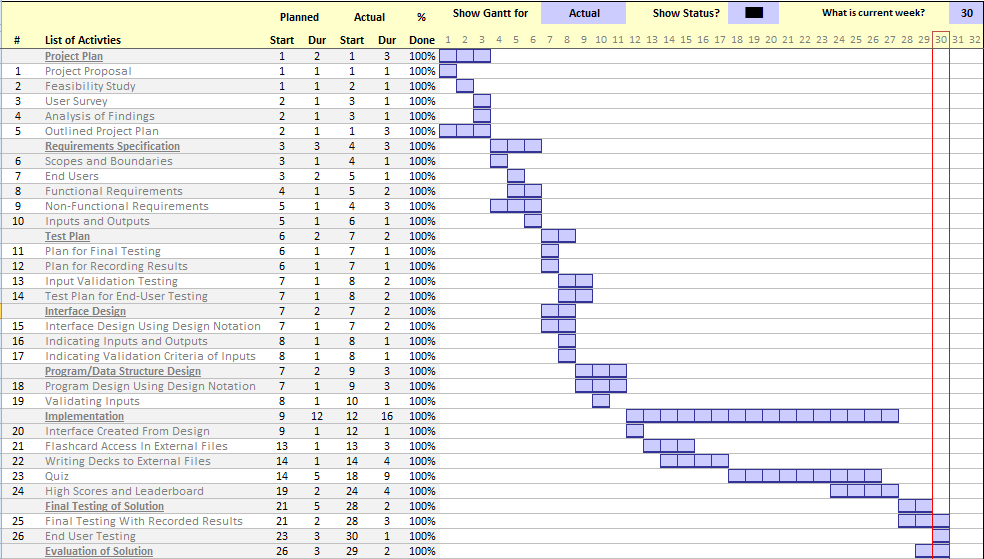
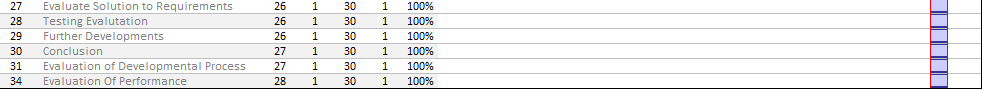
# Personal Performance

## Development Process

Although I had planned to use a traditional waterfall development process while developing my solution, during the project I found it more advantageous to use a more iterative approach. Instead of following the original plan I had made at the beginning of each stage of the project, I developed my initial ideas throughout the implementation, went back, and changed my plans for sections to improve them. I think this method was more beneficial for this project because during later stages of development and implementation, I discovered various areas which could be done differently from how I originally planned, through experience or feedback, and this allowed to me to improve my project. Although this method was not what I had planned to use at the beginning of the project I think that this was the best method for me to use in this circumstance, as it allowed me to evolve my ideas during the development of the project, and reflect changes and improvements in my earlier designs.

## Schedule

Although I used a Gantt chart to ensure that I would stay on schedule to complete the project for the deadline, certain areas like implementation took more time than I predicted, due to errors and bugs which needed fixing. This meant that although I had a clear schedule to follow, I found that I was behind schedule due to my inaccurate predictions, and it was difficult to catch up with the work I had to complete to be back on schedule. As well as this, it was necessary to return at the end of the project to update my earlier plans to ensure that they corresponded with the final solution, which also took time which I had not accounted for at the beginning of the project. Because of this, I have learnt that I need to be more generous with predictions of time to ensure that I allow myself long enough to complete the task to the best of my ability.



By comparing the actual Gantt chart to the planned Gantt chart, it is clear that I expected the project to take less time in most sections than it actually did. Although I had to devote more time to the project than I expected at the end of the project to ensure that I was able to complete it, I was still able to complete the project by the deadline, despite the scheduling issues I had experienced.

## Potential Improvements

To improve my performance in a project similar to this one, I would give myself more generous time allowances for each section of the project, as I have now learned that there will be issues which will arise which I cannot foresee, whether it is a problem in the project itself, or an outside problem which prevents me from achieving my goals on time. If I give myself more time to complete tasks in the future, it is more likely that I will be able to deal with unexpected problems without falling behind schedule, and I will be able to work at the problem at a steady pace, without needing to rush to meet a deadline, which will increase my capability to complete the project to the best of my ability. I would also include more detail when detailing what I had completed in my record of progress, as I have found it useful to be able to return to it to check what I completed a certain time, and what choices I made. It can also be useful if an error occurs as it can be used to find what changes have been made which may have cause the difficulty.

# Record of Progress

|  |  |
| --- | --- |
| 16/6/17 | Brainstormed ideas and started creating a project proposal for my idea. I considered what ideas would be useful and who my ideas may be useful for. I decided to use the idea for a flash card program, as I think this would be complicated enough to be a good project, but still achievable in the time I will have. |
| 18/8/17 | Continued analysing how my project would work and what I would need to do, and added details to my project proposal, with an outline of the program's functions and design. I decided to use external files to store the decks as this would allow the user to access them anytime, and would meet the requirement for the complexity of the project set for Advanced Higher. |
| 23/8/17 | Added detail to how my project would work and what exactly the program would have to do. I also did some research into what my plan has to include and the feasibility of my idea. To do this I used resources I found on the SQA website about the project and information on other schools’ websites to find out what must be included in my plan. |
| 25/8/17 | I continued my proposal, including how I was planning to use an array to store the questions and answers, as I have experience with 2D arrays, and this would allows the questions and answers to be stored in the same array. I also began my feasibility study, looking at schedule, legal, economic and technical feasibility, and who my project is aimed at. I decided that legal feasibility would be fairly simple as long as I avoid using any pictures which are copyrighted, because other than that my program is not affected by any laws.I also wrote that economically the project was feasible as no money was involved, so there were no losses or profits to be considered. |
| 28/8/17 | I added the final details to my project proposal about how the program will handle the high scores in an external file, and started to complete my feasibility study. I added more detail to my technical feasibility, considering the fact that I do not need any more software and the fact that I am using a language I know already, which makes my project much more feasible. |
| 4/9/17 | I finished my feasibility study by completing the section on the schedule, where I included my plan to create a gantt chart to display smaller sub tasks which would make it easier for me to complete the project. I then went on to begin my gantt chart, outlining the deadlines for the different sections of my project. |
| 7/9/17 | I made a user survey for my program, which will allow to me find out from my end users what kind of functions will be useful for my program to have. I asked various questions for things I was unsure about or functions which could be carried out in different ways, so that my solution is as useful as it can be. |
| 10/9/17 | I continued to work on my gantt chart, filling in most of the smaller tasks which I would need to complete. I also began to work on my requirements specification. I began to write the basic requirements for each window. I decided that the user’s name should be entered when the program begins as this means that I do not have to take their name at a less convenient point in the program. For instance if it took their name at the start of the quiz, they would need to enter their name each time the quiz began, which would be unnecessary and inconvenient for them. I also decided to include the top 3 scores in the high scores as I think this is a good amount of high scores. It gives a good range of players without including too many scores. |
| 11/9/17 | I continued on to the requirements for the Deck Editor, which I have decided must provide the user with the ability to move through the deck using buttons, so that they can access each card in the deck easily. I have also included the requirement for the cards’ colour to be customizable, as was requested through my user survey. I have taken results from the survey and decided to follow the results given in it, as I agree in most cases. |
| 16/9/17 | I detailed the requirements for the quiz window, deciding that the display colour will change to display the selected colour of card, and that the user must be able to skip cards which will then repeat later in the quiz. I also created the requirements for the timed quiz, as this is the same as for the regular quiz except it must end when the time reaches the time limit. I outlined the scope and boundaries for my program, by including the various functions I had already thought about for my program. I also had to consider what would not be possible for my boundaries. The solution must be achievable within the time I have been given to me, which I have included. |
| 18/9/17 | I finished specifying the boundaries, including that it must be Advanced Higher level, but still possible for me to achieve within the time constraints. I also added some information on my end users. I realised that they would mainly be younger people, as it would mostly be students who would want to use flash cards. However older people may want to use it, so I will try to make the program as intuitive as possible. I also outlined the inputs and outputs of the program. |
| 25/9/17 | Specified some of the basic operational requirements that my solution will have to meet, for instance that it must use radio buttons to allow the user to select the colour of the card. I used radio buttons for this function as it reduces the risk of the user entering a colour which is not available. |
| 27/9/17 | I continued to add detail to the operational requirements, adding how I planned to complete the timed quiz using a timer. I also decided what information would need to be displayed on the quiz window, and decided that the score and time is important for the user to see how well they did. |
| 1/10/17 | Began to create a table for my test plan, for the tests I would need to do for each section of the program. I used a table with numbered tests so that it is easy for me to use the tests I have prepared and organise the screenshots which I will be the evidence I use from my tests. I also included a reason for testing so that it is clear why I have decided that each section is necessary to be tested. I am trying to include a test for each requirement, to ensure that all requirements are met by the end of the solution. |
| 2/10/17 | Add some more requirements to my requirements specification for the quiz section, and continued work on my test plan. I decided that I had not included enough tests for my high scores, as I had not actually tested the ability to display the high scores from the external file, so I have added this. |
| 4/10/17 | Added the non-functional requirements for my quiz to the test plan, as well as some of the functional requirements. I also added the plan for the end user testing for when I have to get other people to try my program. I made sure to check whether the program is intuitive, easy to use and more efficient than physical flash cards, as this is the target for the solution. I also added space for them to report bugs that they find, so that I can fix them, and a space to report any adjustments that need to be made to make the program easier to use. |
| 6/10/17 | I completed my test plan for the final window, which means that the test plan is now finished. I began to design the interface for my solution, creating the Main Menu window, and the Deck Editor window. I did this by creating basic sketches on paper, so I can get an idea of what it will look like. I chose to display the deck which is currently open on the Main Menu to make the program easier to use for the user. I also decided to display the card number that is currently being viewed on the Deck Editor for the same reason. |
| 8/10/17 | I continued to design the interface for my program, by designing the window for high scores, and the window that is displayed when the quiz is completed. On the High Scores page I changed my mind about what I am going to display on this page. I realised that I didn't need to display both the points and the questions answered correctly, so I only included the points in my design. |
| 9/10/17 | I finished the basic design of my solution by designing the Quiz window of my program. While designing this window I made decisions about how to make the questions and answers clear, so I made these list boxes the biggest part of the window, to ensure that the design is intuitive to use. The design also includes the amount of cards which are in the deck, so that the user has an idea of how long the quiz will be. |
| 10/10/17 | Added the input validation to my test plan, as well as the tables for recording the screenshots I will take when I am testing my solution. I have numbered each section so that I can find which part of the program I am testing by referring to my tables which specify what needs to be tested. |
| 17/10/17 | I began to annotate the basic designs I had made for the UI of my program. I included what each symbol signified, for instance a button or a list box, and what its function would be. This means that when I implement the design I have a clear guide and it is also obvious what each object is there for, and what the requirements for it are. I have annotated all but the Completed Quiz window. |
| 21/10/17 | I completed the annotations of the designs for my UI.  I then went on to make a more updated version of the UI, on paint, which is more clear and includes some changes where I no longer agree with the original design.I reorganized my main menu slightly to include radio buttons, to make it easier for the user. I have also changed my mind on what I wish to display on the high scores window. Because I am no longer rewarding more points for a user answering questions correctly, the points and the amount of correct answers are the same, so I have now decided to just display the amount of questions which were answered correctly. |
| 25/10/17 | Before I began the pseudocode, I created a simple diagram which displays how the program moves between the forms I have used, which makes it clearer how the user can move through the program. I made basic pseudocode for the main menu of my program. Before I began the full pseudocode for my Main Menu window, I made a basic outline of the functions of the window. After I had completed the outline, I began writing basic pseudocode for each function, which can be refined into more detailed pseudocode. I also began the basic pseudocode for the Deck Editor. |
| 1/11/17 | I finished my basic pseudocode for my Main Menu window, and began adding detail to it, so that it is more useful when I am using it to implement my program. I have completed the pseudocode for some functions of the Deck Editor. |
| 2/11/17 | I created basic pseudocode for my high scores window to use as an outline. I also realised I had some repeated code in the pseucodode for my Main Menu, so I added it to a separate procedure to make it more efficient. |
| 3/11/17 | Updated the dates on my record of progress because I realised I had entered the incorrect month for a lot of my entries. I decided to use a linked list to store the properties of the flashcards, instead of an array which I had originally planned, as using a linked list will make it much easier to implement the editing the deck. This is because it is easier to add new values on to a linked list, instead of redefining the size of the array or having to count the amount of lines in each file to find the size. If I use a linked list then it is just a case of adding a new node on to the existing list, which will be easier to make and more efficient, because I won't have to read each file twice to create an array of the correct size and read in the data. |
| 6/11/17 | I began to implement my design in Visual Studio, using the outline which I created and annotated earlier. I created the Main Menu, and chose to move the buttons slightly closer together than they were in my original sketches as it looked cleaner. I also finished my pseudocode for the deck editor. |
| 8/11/17 | I continued to create my user interface, by creating the High Scores window and the Deck Editor window. I decided to use list boxes for the high scores as it would make it easier to group the different categories together. I also began looking into how the use the picture boxes to make the arrows on the Deck Editor window. I then began my pseudocode for the quiz window. |
| 10/11/17 | Today I worked on finishing my pseudocode for the quiz window, and then I began getting my program to be able to take in the file name of the file selected by the user. To do this I had to separate the file name from the file path and the extension. To do this, I had to find the file name from the file dialog window, which meant I had to find out how to do this is Visual Basic, and to do this I used this page: <https://stackoverflow.com/questions/30308485/get-full-path-using-openfiledialog>. I also needed to make sure the file was the correct type of file, i.e. a .csv file, so the program needed to check the extension and make sure that it is .csv. To do this I used the split function once I had separated the file name from the path, and separated the file name from the extension at the full stop, which gave me just the name of the file, so I could display the name of the open file, and save the file successfully. |
| 13/11/17 | I continued making my UI, making the window for the quiz and the window which is displayed once the quiz is completed. I then programmed the buttons which take the user from window to window, so the program can now move between each window in the way it would when the user is operating the program. After this I made the basic pseudocode for the completed quiz window. |
| 14/11/17 | I finished the basic pseudocode for the completed quiz window, and added more detail so that it will be easier for me to use when I am implementing my program. |
| 15/11/17 | I started making the code to read in the file which had been selected by the user, now that it could successfully get the name of the file. To do this I made the class I am going to use to store the values of each card in the deck. I created a class with attributes for the question, answer and colour of each card, as well as an attribute which serves as the pointer of the linked list which will be created to store each card in the deck. I then made 2 instances of the class, one called start which is the head of the linked list, and one called current which enables the program to move throughout the list. |
| 17/11/17 | To read in each line of the document and separate the different sections to save them in the different attributes, I used the split function to split the line which had been read in where the comma was, as the file is a csv file. I then saved these split values in a temporary variable, which is an array of strings, so I could then separate them into the attributes they are meant to saved under; questions, answers and colours. |
| 20/11/17 | Once I had the code to separate the files from last time I started to make the loop to read in all the lines of the file. I used a do while loop which would loop while there was still a value to read, by checking if there was a value in the next node of the linked list before it looped. I would then separate the values on each line and continue to do this until there were no more lines to read in. |
| 22/11/17 | I found that the code to take in the information wasn't working, because the attributes in the linked list weren't set to an instance of the object. This was because I hadn't made a constructor method to use when I created the 2 instances of the class which stores the values of the cards. I made a constructor method which set the question, answer and colour to a blank string so that their values could be set later, either when a file is read in or if the deck is created in the program. |
| 23/11/17 | The loop I had used which was meant to continue until the whole file was read didn't do this, so I tried to figure out why. Instead of looping for the whole file, when I went through the process of the program slowly I found that the program wasn't executing the loop at all. I thought this might be because when it performs the check at the start of the loop to see if there is anything in the next node of the linked list, this node is automatically equal to nothing because nothing had been assigned to it yet, so it is just skipping over the list. I experimented with changing the constructor method so that the node does not get created with no values, but this code created errors because it wasn't correct, so I changed the method back to how it was before and looked for another solution. |
| 24/11/17 | I discovered that the reason the code for the loop didn't work was because of the reason above, however the method I was using isn't the way to do what I wanted it to do, which was why it didn't work. Instead, I used the peek function at the start of the loop, which checks to see if there is something to read on the next line before it reads it, and if there isn't anything it returns –1. This means I can use a do while loop to continue until the peek function returns a –1, which means it has reached the end of the file. The program could then split the values using the code I wrote before, however I ran into another issue while trying this, because the code I had only moved the linked list along, and I had forgotten to create another node to move along to. Instead before moving the list along at the end of the loop, I called another instance of the class in the linked list using the constructor method, and then the current node would be changed to the node which had just been made, meaning that when the loop continued it had a new card to store the values in. |
| 27/11/17 | Now that the program could read a file in, I decided to work on allowing it to create a new deck of cards. Because the editor is in a separate window from the main menu, this means I have to use variables from different forms, as the linked list of cards is originally created in the form for the main menu. I had to research the way to do this, using this website: <http://www.dreamincode.net/forums/topic/57847-forms-accessing-controls/> When the create card button is clicked, the program now sets the values of the current question and answer to the values inside the respective text boxes. I also added a statement which deck if there is a deck open when the High Scores button is pressed, and if there isn’t the window will not open. |
| 29/11/17 | After checking that the program was taking the values from the text box correctly, I made it so that the text boxes are automatically cleared, so that they are ready for the entry for the next card. I then used the same code which I used to make a new node and move to this node which I used in the code to read in the deck from a file on the main menu. |
| 1/12/17 | Today I decided that having both a save button and a quit button is redundant for the program, as they both essentially do the same thing. The save button writes the deck to an external file, but the quit button also confirms if the user wants to save their deck in case they haven't saved it already, which means that it will do the same thing as the save button. To make the design simpler, I decided to just get rid of the save button altogether, which means the user can just save the deck when they exit the Deck Editor. |
| 4/12/17 | Made the forward and back buttons in the Deck Editor functional, so that the user can move back and forth through the program. To do this I used the position that the card is in to loop through the linked list until it reaches either the next card or the card before. |
| 6/12/17 | I found that the program ran into an error when trying to move to the first card from the second, because there was no instance for it to move to. To fix this, I created a loop which executes separate code depending on whether the program is currently displaying the second card or another card. If it is currently on the second card, it will now make the current card the first card in the deck, instead of trying to loop through it like it does for the other cards, as this will not take it to the correct card. It then displays the card and resets the position, like the other section of the loop. |
| 8/12/17 | When the user opens a deck to edit, it should open the deck at the end, so that it is ready to create new cards. To do this, I used a Boolean variable in the Main Menu, which determines whether the user is making a new deck or not when the Deck Editor is opened. This means that it can operate differently depending on whether the user has made a new deck, in which case it needs to display a blank first card, or if they have opened a pre-existing deck, in which case it needs to display a blank card which is after the rest of the deck, and they must be able to use the back button to move to the rest of the deck. This means when the Deck Editor is opened I can use an if statement to determine which of these needs to be done. If the deck is being edited, it now displays a blank card, but the position in the deck is 1 greater than the length of the deck originally. The Deck Editor will only save this extra card if there is text added to it, as it does not save any blank cards. When the user uses the back button, it will cycle through the deck and display the last card, leaving them free to edit the cards which already exist as well. |
| 16/12/17 | Currently when the program displays the position of the Deck Editor, it is 1 less than it should be, so the first card is displayed as 0. I tried to find the cause of it, but I was unable to. On the next day I will investigate whether the file is being read in correctly. |
| 18/12/17 | I found that the problem was being caused by the way the position is calculated when a card is created. I changed the count and tested it, and the program can now display the correct position in the deck. |
| 20/12/17 | To avoid the program running into an error when opening the high scores window, I added another check, to make sure there is a scores file for the deck. The program already checks if there is a deck open, which means that the High Scores window cannot open if there is no deck open, but it also needs to confirm that there are actually any high scores for it to read in. It makes more sense to display this error to the user before the High Scores window opens, so that it is not unnecessarily loading this window to tell the user that there are no high scores. To do this I had to find if there was a predefined function which checks if a file exists, and found this page: <https://www.techonthenet.com/excel/formulas/dir.php> . This allowed me to write a conditional statement which will only open the High Scores window if the function does not return a blank value. If it returns a blank value, this means that the high scores file does not exist, and the window should not be opened. |
| 26/12/17 | Today I worked on the function which chooses a random question to display to the user during the quiz. To do this it chooses a random number which corresponds to the question’s position in an array. It then has to make sure that this question has not already been asked, so I created an array which stores all the questions which have been asked already. The computer can then select a random number between 1 and the length of the deck, so that it will include all of the questions. It can then send this to the procedure which displays the question. I used a Loop Until loop so that the program will only return the selected question if it has not already been asked, and it can confirm that this question is valid first. |
| 28/12/17 | So that the program could use the question it had selected, I wrote a procedure which displays the card to the user. To do this, it receives a valid question from the function which I wrote before, and it then cycles through the linked list until it reaches the selected question. It can then change the form colour to the selected colour, and display the question and answer to the user in the list boxes. Later on I will remove the answer from the list box so that it is not immediately visible to the user, but for debugging purposes it is easier if I can check that the correct answer is being displayed for the question and colour. |
| 29/12/17 | Now that the program can select a question and display it to the user, I wrote the code for the buttons which allow the user to answer the question. When they answer the question either correct or incorrect, the count must be increased so that this value can be displayed to the user at the end of the quiz. |
| 2/1/18 | Today I removed the code which displayed the answer at the same time as the question, and instead added this code to the reveal button, so that the answer is only revealed when this button is pressed. As well as this, I also added code to the skip button which checks whether the user has revealed the question before it allows them to skip the question, so that the user cannot skip the question if they have revealed the answer, as this would be unfair and not testing them properly. The skip button now checks a Boolean value which is set to true when the answer is revealed. If it is false the question can be skipped. |
| 6/1/18 | Today I fixed a problem I had been experiencing with the section of code which selects a question to display to the user. When the program runs it will display some questions, but when I move on from a question it will sometimes not display any more questions, but it also does not return an error, which means that it has not actually crashed, but it is not functioning correctly. I decided this was probably caused by the function which selects a question not running correctly, perhaps if it was not completing due to an issue and was continuing to run, leaving the program waiting for its output. As I had thought, this was the issue. I have used a Boolean value to confirm that the question selected isn’t invalid, but I had not included any code which resets this Boolean at the start of the function, meaning that when the program is trying to select the second question it thinks that every question it selects is invalid, due to the fact that this Boolean is still set to true. If the program has not selected any invalid questions it will continue to function, which was why it was able to display a different amount of questions each time, but if it chose 1 invalid question it would then run into this error. To fix this I included a line of code which set the Boolean back to false at the beginning of each iteration of the function. |
| 10/1/18 | While testing the quiz, I found that although the quiz could now correctly choose a question from the list and display both the question and the corresponding answer, the quiz was finishing without displaying all the questions in the deck. Every time the quiz was started it would end 1 question before it was meant to. I used a breakpoint so that I could run through the section of the code which ran through the quiz in detail and found that the variable which stored the length of the deck, which is found when the deck is read in on the Main Menu form, stored a number which was 1 lower than it should have been. To find out why the number was being counted incorrectly, I checked the piece of code which counted the length of the deck when it is read in, in the Main Menu form. At first, I thought that the code was correct, as for each time it read in a question and answer it added 1 to the length, but I realised that the problem was caused by the section of the loop which read in the final card in the deck. The loop continues as long as there is another line after the current one to be read in, which prevents the program trying to read the file after there are no more lines to be read. After it reads in each line, it makes a new card in the linked list to save the next card which is read in. So that the loop would not make a new card after it had read in the final card, I included an if statement which would check that there was another card to be read in before it creates a new node in the linked list to store the card. This part of the loop was what was causing the issue in counting the length of the deck, as I had included the line of code which added one to the length in the section of code which made the new card, which would only be executed if there was another card to be read after the current one. This meant while the program was reading the final card, it would check to see if there was another card to be read in, find that there was not, which meant that this if statement would be ignored, meaning that no new card was created, which was what I wanted, but also that the length would not be updated to include the final card. This issue was fixed by simply moving this line of code into the main section of the loop which assigns the questions and answers to variables, which meant that it would be completed for every card, including the final card. I went back and stepped through this section of code again and found that it now counted the length correctly. |
| 11/1/18 | While trying to make a new deck in the Deck Editor and use it in the quiz, the program ran into an error, because when it tried to read in the file for the deck I had just made, it found that there were no answers or colours stored for each question, and only the questions had been written to the file. I checked the code which wrote the card to the file when there wasn't already a file for this deck and found that I had mistyped the line of code, which led to the error I had found. I had not included the & sign between each answer, colour, and the commas in between them, which are needed to separate them in the CSV file. Because there were no & signs included the program was only writing the question for each card to the file, and not the commas or answers and colours. I also found that the reason I had not encountered this issue before was because before now I had not been making new decks in the program, I had only been editing decks I had made manually out with the program and editing them to confirm that the Deck Editor was working. I have different code to write the deck to the file depending on whether there is already a file for the deck, which meant that when I was editing a deck which already exists, it would not use the line of code which was not working, so I had not encountered this problem before. To fix it, I just copied the code which I already had for the decks which exist already to the other section, as I knew that this code already worked. I made a simple new deck to confirm that it was now working. When I checked the file it had written the deck to, I found that the answers and colours had been written there successfully, and there were commas separating them. |
| 13/1/18 | Now that the player could complete the quiz and the program would ask each question in the deck once, and repeat the questions which had been skipped, I chose to work on displaying the user’s scores, using the variable from the quiz which counted this. While I was doing this, I decided to just display the amount of correct answers as correct answers instead of points, which was what was in my original design. This is because the design of the user interface was more consistent if I used "Correct Answers" and "Incorrect Answers", instead of "Points" and "Incorrect Answers", which does not make as much sense. I have already decided in the earlier stages of my design of the program not to include points which are awarded for answering answers correctly, and my reasoning for this decision is detailed in my analysis of my user survey, as a result of the feedback I received. Because of this, I decided it made more sense to simply get rid of points entirely in my program, and just use the amount of correct answers to decide how well the user scored, as the points would equal the same value anyway. This made the design of the UI and the program overall simpler and clearer to the user. If I used points as opposed to just referring to the number as the amount of correct answers, it is more likely to confuse the user, as there is no other use for the points in the program after I decided to remove them from the quiz itself.  Because I have not yet begun the part of the program which writes or reads the scores which are stored in an external file, right now the window can only display the amount of answers the player got correct, incorrect, and the time they took to answer all the questions. To display this information, I used labels like I specified in my original design sketches, as they are easily changed and do not allow the user to edit them. Once the code has been completed to write the scores to the external file, and the user's score can be checked against these, I will add code to change the label which will display a message if the user achieves a high score. |
| 23/1/18 | To write the scores to an external file, the program first needs to sort the high scores and names so that when they are read into the program they are already in order. I have decided to sort the scores when the high scores are written to an external file, which only occurs if the player has achieved a high score. This is more efficient, as it won't rewrite the high scores after each game if there are no changes to be made to them, and it also means that once the scores have been sorted correctly once it does not need to be done again until a change needs to be made, as it will be stored in the correct order. If the scores were written to an external file without being sorted, they would need to be sorted every time they were read in to be displayed to the user or used by the program, which would be less efficient and can be avoided.  As well as this, I have decided to read in the high scores for the deck which is currently open after the player completes a game, so that the program can then check whether the amount of correct answers the current player has just achieved is higher than any of the high scores, which would mean that their score would need to be added to the list of high scores instead of one of the current scores. This means that the scores need to be read into the program from the external file after every game is completed, so that the program can confirm whether the player has achieved a high score. However, I have decided that this method is preferable to the alternative if it did not check whether they have achieved a high score before starting the process to write the high scores to the file. Using this method, the program just needs to read the file in, but if it did not do this, it would be reading the file in after each game anyway so that it could sort the high scores to write them to the file. In addition to reading the file in regardless, in the situation where the player has not achieved a high score, it would mean that the program would be completing a redundant sort on the scores, as they would already be in order, since the player has not got a score which needs to be added to it. If the program reads in the file to confirm if the user has a high score, this means it can then only proceed with the process to sort and write the scores to the file if they have a high score, and if they do not it does not have to do this process at all, which is more efficient. The program also needs to check if the user has achieved a high score anyway so that it can display whether they have or not on the Completed Quiz window after the quiz is finished, so once it has been checked this can be used to display the correct information.  So that the program can check if the user has a high score after they have completed the quiz, I decided to use a function which returns a Boolean value, which will be true if they have a score which is higher than one of the scores which is currently saved for the deck, and false if they do not. I decided to call the function which checks for a high score in the procedure which starts when the Completed Quiz window is displayed, so that this information can then be used in the procedure which displays the information to the user. In this function I created an if statement, so that the program would do different things depending on if the deck already has a file containing high scores. I used the Dir function with the path for the file, which is the Documents folder, along with the name of the deck and "score" which will be what the file will be named when it has been written with the high scores. I used this function to determine whether the file exists already, as it returns a numerical value which describes the file. I can use this to find whether the file exists by checking if it returns a null value, which means that the file does not exist. If it does not return a blank value, this means the file exists and can be read in to find the high scores. |
| 29/1/18 | Because the information the program is reading includes a string, which is the name, and integers, which are the amount of answers which are correct, incorrect, and the time taken, I decided to use a string to read in the values, and then once the information has been split using the Split function use a function to convert the numerical values into integers before they are stored in the variables. To do this I needed to find the function which is used in Visual Basic to change a string to an integer, which required research as I did not know what the function was. I found it on this page: <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/functions/type-conversion-functions>. With this function I could then convert the number values to integers and store them in the variables so that they could be compared with the player's score to determine if they have reached a score which is higher than the current scores.  After the values had been converted I could then use a fixed loop to compare each value which had been read in from the file to the amount of correct answers which the user just got in the quiz. If their score is higher than any of the values on the high score list, then the Boolean which the function returns is set to true, and otherwise it remains false, which means that the user did not beat any of the scores on the high score table, and therefore they do not need to be changed. |
| 1/2/18 | Because the Completed Quiz window needed to read in the high scores, I already had the code to read in the high scores, which I can use again to read in the high scores in the High Scores window when it is opened. I can use the code I used to confirm if there is a high scores deck on the High Scores window now to check if there are already high scores for this deck. After this I just needed to add a fixed loop which adds each set of high scores to the correct list boxes to be displayed to the user. |
| 2/2/18 | Now that the decks, quiz and high scores window were functioning, I decided to make the timed section of my quiz. I decided to leave this part until this point as I wanted the rest of the program to be operational before I began making the timed section, as it included using a timer, which is a function of visual basic I haven't used before. Because the regular quiz format works, this makes it easier to work with the timer, as I know that the rest of it should function correctly.  To find out how to use a timer correctly, I looked at both a page on using timers: <https://msdn.microsoft.com/en-us/library/aa231154(v=vs.60).aspx> and a YouTube video explaining how to program a timer to count once per second: <https://www.youtube.com/watch?v=7ZrLjJm_QB8&t=214s>. After watching these I felt more confident in attempting to make my own timer. I now understood that the timer needed to be added to the form and the tick length was changed manually, and that the timer must be told to add time to the variable storing the time when it ticks, as it does not do this automatically. I used a message box which displayed the time every time the timer ticked to check that my set up did work, and when the quiz started I found that it did tick once every second and add this to the time variable. |
| 3/2/18 | For the timed quiz, the program must check whether the timer has reached the limit that the user selects when they begin the quiz, so that the quiz can end when the time limit has been reached. The program now checks that the limit that was entered at the beginning of the quiz is not the same as the limit each time the user answers a question, and if it is the quiz will instantly end and it will move onto the Completed Quiz window, displaying the amount of answers they got correct and incorrect like it does if they completed the whole quiz. The Completed Quiz also checks if the quiz was timed or not, and if it was, it does not run the function to check and write high scores, as the user cannot achieve a high score with a timed quiz. |
| 4/2/18 | On the Completed Quiz window, I changed the display slightly for the label which tells the user whether they achieved a high score. Instead of only displaying a message if they did achieve a high score, and being blank if they didn't, I decided that it should display a message telling them that they didn't. I decided this it is clearer to the user and leaving the window blank makes the window look empty.  I also had to change the code which checks if the timed quiz has been completed, as when I was testing it I found that because it checked after the user answered a question, if they stayed on one question for a while without answering it the timer could count on past the limit, and because it only checked the time against the limit when the answered the question, it would not end the quiz. This meant that the user could use this to spend longer than they were meant to on one question without letting the quiz end when they ran out of time. I realised the sensible place to put the code was when the timer itself ticked, so that each time it added time to the timer it would then check the time against the limit and ensure that it had not yet reached the limit. This meant that even if the user was not moving beyond a question, if they are still thinking about it and haven’t yet answered it, it will still end the quiz when the timer reaches the limit, as it is meant to. I tested this, entering a short time of 5 seconds, and did not move beyond the first question to ensure that it would still work without any inputs made on the quiz window. As I had hoped, the quiz ended after 5 seconds, and did not require me to enter any answers, which means it is successfully checking the time against the limit when it ticks instead. |
| 5/2/18 | Today I added the input box which asks how long the quiz should be, using the radio buttons, so that if the timed quiz radio button is selected it will ask using an input box. I also added validation to this input, as only a number should be entered. To do this I researched what the best method is to find if an entire string is only consisting of numbers, to see if there was an easier way to do it than looping through the whole length of the string confirming that each part was a number. On this page: <https://stackoverflow.com/questions/6138896/how-to-check-if-a-string-contains-only-numbers> I found that there was a function already in Visual Basic which would check this for me, which would save me having to program this myself. I then had to find out how to use this function, and to do this I used this page: <https://msdn.microsoft.com/en-us/library/microsoft.visualbasic.information.isnumeric(v=vs.110).aspx>, and found that it just returns a Boolean value, which meant that I just had to check that this value was true, and if it was then the user had only entered numbers, and no letters. To use this in my input validation, the program first takes in the input as a string, in case the user has entered characters which are not numbers. This means the program won’t run into an error if this is the case. The program can then check that it only contains numbers, and if it does, the string can then be saved in the variable which stores the time limit. This avoids the program running into an error by trying to store any letters the user may have entered as an integer. |
| 6/2/18 | Earlier in the project I received some feedback that it may be useful to have some instructions explaining the program’s functions, as at a glance it is somewhat unclear. Now that the program was functional, I decided to add a button which took the user to some instructions explaining each function of the program. Because there are various functions of the program, I decided the simplest way to display the different instructions would be to use multiple buttons on a separate form for the instructions. This way, I can have separate instructions for the separate different areas of the program, such as the deck editor and the high scores. If they were all contained in one group of instructions, it would be too confusing to organise, or for the user to read. By arranging it like this the user can easily know which section they are looking at, at can also find more information on a specific area of the program easily. In these instructions I explained the basic functions of the program, as well as explaining the way the high scores are chosen and organised in the high scores section of the instructions, as this may not be clear to a new user. |
| 8/2/18 | Although it is not in my original design plan for the program, I decided to add the amount of incorrect answers the user got to the display on the High Scores window, as otherwise it is not as clear how well each user did in each deck. Although it does already have the amount of correct answers, without the incorrect answers there is no way of knowing how many cards there are overall in a deck. This makes it easier to compare the high scores and clearly see how well each person did. To do this I simply added a new list box and label for the incorrect answers and added a line of code which adds each incorrect answer for the high scores to the list box when it needs to be displayed, the same as I have done for the other list boxes. As well as this, I added code in the procedure which writes the high scores to the external file, so that the file now contains the correct answers, incorrect answers, time and the name of the player. |
| 12/2/18 | So far, the files I have been using for the decks and high scores have just been saved in my Documents folder, which has made it inconvenient to find the correct file, and also made it confusing for other people who are less familiar with the program and aren't as clear on how to use it. To improve this, today I made separate folders to store the decks and scores. I also changed the file paths in the program, so that the decks and high scores are automatically written to the decks and scores folders respectively. I also decided to find out if there is a way to open the file window on a different directory than the Documents folder which it automatically opens on, and found this page: <https://msdn.microsoft.com/en-us/library/system.windows.forms.filedialog.initialdirectory(v=vs.110).aspx>, and I found that I just had to change the initial directory value of the file window to the directory path I wanted it to open on. In the Main Menu, when the user selects the deck from the files, I changed the path to automatically open in the deck folder, which avoids the confusion which is caused by either having the decks in the Documents or making the user find the correct decks folder themselves. This is much easier for the user as they are already looking at the files which they are meant to choose from. |
| 16/2/18 | To make the High Scores window clearer for the user, I worked on making any selected values in one list box match with the selected values in each other list box, so that it was easier to check a person's score against their name. I had to find the value which is used to make the value which is selected equal to the next value, so I found this page: <https://msdn.microsoft.com/en-us/library/system.windows.forms.listbox.selectedindexchanged(v=vs.110).aspx>  And I could then use the event on the selected index being changed for each list box to change the other ones as well, by making each list box's selected index equal to the selected index of the list box which had just been changed. This then meant that whenever a value was selected in any list box it would change every other list box's selected value to be equal to the first box's selected value, which gave me the result I wanted. |
| 21/2/18 | Today I realised that if the user changed any values of cards in the deck editor for cards which already existed, it would not update the values in the linked list with the values which had been changed, for instance changing the colour. To do this, I decided to make a new procedure which ran whenever the arrows were used to move from one card to another, which would update the values stored in the linked list to whatever is currently included in the deck editor. The procedure was mainly the same as the procedure to create a new card, except that it didn’t need to make a new card, it just updated a card which already existed. When I tested the procedure, I found that the cards updated correctly, except for the fact that the first card’s question and answer were set to blank values. I was unable to find the cause of this. |
| 24/2/18 | While trying to find the cause of the problem which was making the first card blank, I commented out the line which updated the cards to confirm that it was this procedure which was causing the problem. Because the procedure is called during the process for the forward and back buttons, it is called at two separate times for each button, because the buttons have different processes, one for most of the cards and one for the first or last card respectively. Although it is the first card that is being changed, I found that commenting out the update procedure for the section of the back button which moves to the first card specifically did not solve the problem, which means that the first card is being changed before this point. I confirmed this by commenting out the other time the update function is called in the back button procedure, which stopped the first card from being changed. I still could not find the cause of the problem, but I had managed to identify the point at which the issue was occurring. |
| 1/3/18 | Today I managed to fix the issue which was updating the first card incorrectly. Before I had only been stepping through the program using a breakpoint when the program displayed the first card, because I thought this was when the change was being made. Because I knew now that the card was being updated before the program displayed the first card, I stepped through the program when the back button is first pressed. I found that the card is being updated the first time the back button is pressed. I found that this was because the current card is automatically set to the first card when the deck editor is opened, and because when a deck is being edited it displays a blank card, this meant that the update procedure updated the current card, which was the first card, with the values in the text boxes, which were the blank values for the user to create a new card. To rectify the issue, I added a condition in the if statement so that it would only update the card if both text boxes contain text. This will mean that it will no longer accidentally update the first card, and it will also mean the user cannot input blank questions and answers, which is also beneficial.  I also updated my structure diagram to include the instructions. |
| 5/3/18 | I updated the test plan I had created to reflect some changes I had made throughout the design process, for instance that I had changed the file paths where the decks and scores were stored, so that they were now in separate folders. I also had now included an instructions window, which meant I had to test the functionality of this as well, so I added a test plan for this window, to ensure that it can display all of the instructions correctly. |
| 7/3/18 | I started testing my program using the test plan I had created. I completed the testing for the main menu and the high scores window, and began the testing for the deck editor. |
| 8/3/18 | I continued testing my program, finishing the testing for the deck editor window, and beginning the testing for the quiz window. |
| 9/3/18 | Due to the fact that I wanted the user to be able to re-do the quiz without having to close the program, I changed the code which closed the window to close and dispose the window, so that it would reload the window when it was next opened, and therefore all the values and objects would be reset. This would allow the user to redo the quiz easily. I also continued the testing for the quiz window. |
| 10/3/18 | I began the user testing while finishing my own testing, and discovered a problem with the program, which is included in the end user testing feedback. The completed quiz window was no longer displaying the values for the correct or incorrect answers, or the time taken. This was confusing, as this function had been working. I checked the values during the quiz by using the watch function in Visual Studio, and found that the program was still counting the values correctly, it was just not able to pass them to the completed quiz window at the end of the quiz, and when I tried to view these values it could not find them as there was not an instance of the object I was referencing. Although this was a problem while I continued to try and find the problem I received feedback on other aspects of the program from the end users. |
| 11/3/18 | I discovered that the cause of the problem I had experienced was because I had decided to close and dispose of the quiz window when it moved to the completed quiz window. Although this meant that the user could redo the quiz, it also meant that the values were no longer stored in the quiz window as it had been disposed of. Instead of disposing it once the quiz was completed, I changed this to hide the window while the values are being displayed so that the values are still accessible. The quiz window can then be closed and disposed of when the user returns to the main menu, as the values are no longer needed. I made these changes, and found that this did indeed fix the problem. Now that this problem had been fixed I could continue with the testing for the quiz and the completed quiz window. While testing the quiz, I discovered that because I was using a list box, which does not automatically wrap the text, if the question or answer was too long it would not display properly to the user. I considered writing a function which would separate the string and display it in separate lines, but I decided that I did not have enough time to complete this if I wanted to finish the project for the deadline, so I simply activated the horizontal scroll bar on both list boxes. This is not as convenient for the user, so preferably I would include the function to split the text, but time constraints have prevented me from doing this. |
| 12/3/18 | After completing the testing for each quiz, I was able to continue the user testing, as the program was now functional again. I was now able to begin my evaluation of the project. I had already outlined the areas which I needed to cover during my evaluation, and I began to add evaluation of whether the program met the requirements, as well as the development process I had used. I also added some corrective maintenance from the feedback I have received from the end user testing. One tester had asked that the instructions screen be made more clear, by separating the paragraphs of instructions, so I spread out the instructions. I also added the label at the top of the instructions window so that when the user opens some instructions the title displays what section they are reading. This will make it more clear for the user as they will be able to remember what they are reading about. |
| 13/3/18 | I created an updated data flow diagram, which included the instructions and other processes which have been added since the initial plan stage. I also added more detail to many areas of my pseudocode, as many parts of it are too vague and unclear as to how the solution should be implemented. To make it more detailed, I have also included a class diagram for my card class, which more clearly displays how it has been implemented. |
| 14/3/18 | I reorganised my test results so that it was clear what test had been carried out, and you no longer have to return to the test plan to check which test you are looking at. I also finished my evaluation of the program, including the different aspects of the program like maintainability, efficiency and robustness. This allowed me to provide a detailed evaluation, considering many different aspects of the solution. I also completed my evaluation of my personal performance, including the evaluation of the schedule I had used throughout the project, and that I had leaned to be more generous when allocating time for a large-scale project like this. |