Information

The following document is the write up for my software systems development coursework.   
This covers all information about the application from early designs to testing.

SSD COURSEWORK WRITE UP

Educational technology and design quiz

Joshua Carrothers

Center number: 71523

**Contents**

[**Game Background** 3](#_Toc161772527)

[**Requirements** 4](#_Toc161772528)

[**Draft Storyboards** 5](#_Toc161772529)

[**Final Storyboards** 12](#_Toc161772530)

[Login & Register template 12](#_Toc161772531)

[Login 13](#_Toc161772532)

[Register 15](#_Toc161772533)

[Main Menu 18](#_Toc161772534)

[Home 22](#_Toc161772535)

[Question Template 24](#_Toc161772536)

[Radio button 26](#_Toc161772537)

[Drag and drop 27](#_Toc161772538)

[Combo box 28](#_Toc161772539)

[Text fill 29](#_Toc161772540)

[Summary screen 30](#_Toc161772541)

[Player editor 32](#_Toc161772542)

[Player creator 34](#_Toc161772543)

[Question Editor 36](#_Toc161772544)

[Mini game – snake 38](#_Toc161772545)

[**Testing** 39](#_Toc161772546)

[Introduction 39](#_Toc161772547)

[Testing Map 39](#_Toc161772548)

[Testing table 42](#_Toc161772549)

[Failed Tests 53](#_Toc161772550)

[**Evaluation** 60](#_Toc161772551)

[User Requirements Evaluation 60](#_Toc161772552)

[System Evaluation 66](#_Toc161772553)

[Strengths 66](#_Toc161772554)

[Weaknesses 68](#_Toc161772555)

[Further improvements and future developments 69](#_Toc161772556)

[Evaluation of peer feedback 70](#_Toc161772557)

[Self-Evaluation 72](#_Toc161772558)

[Testing evaluation 74](#_Toc161772559)

# **Game Background**

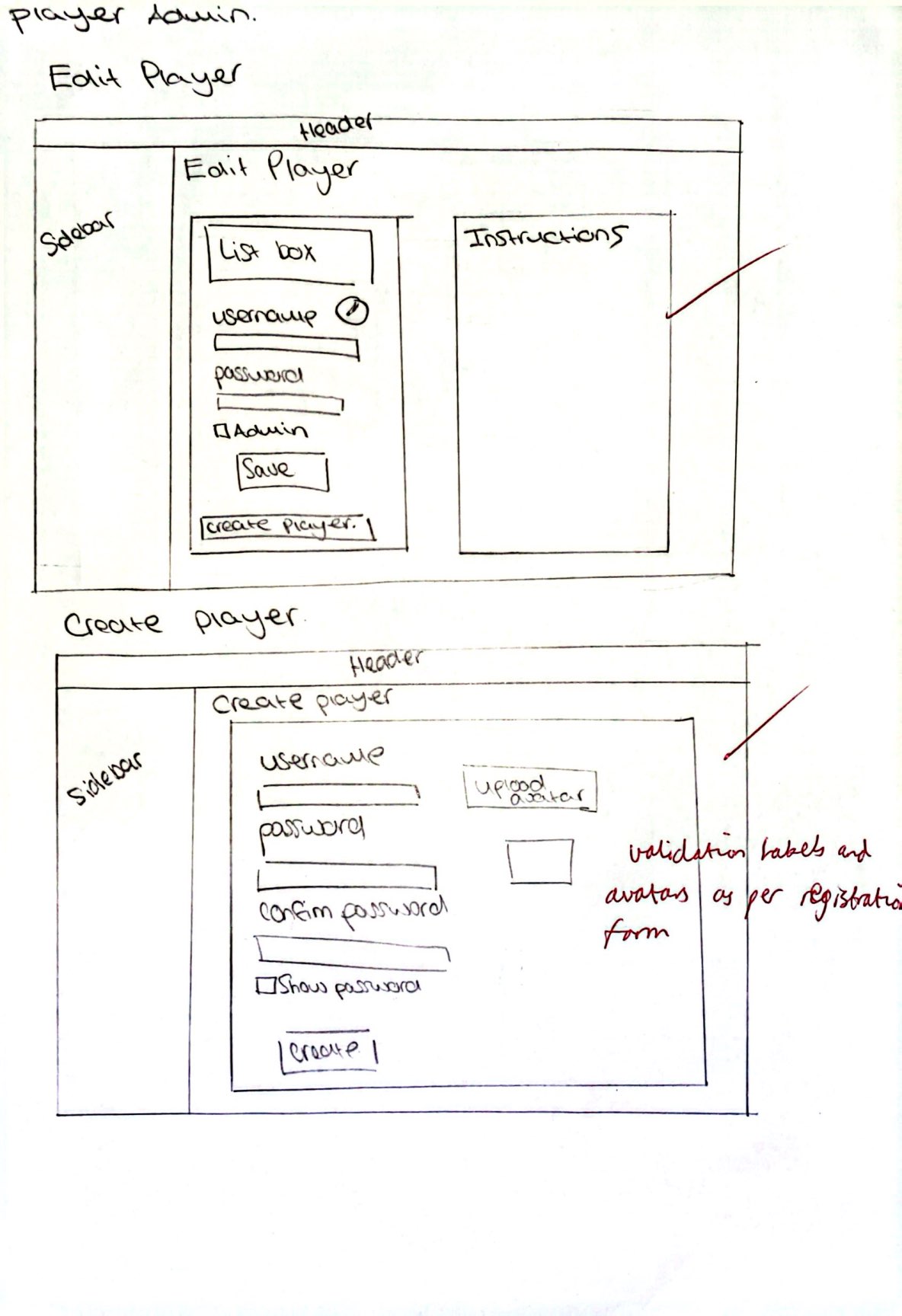
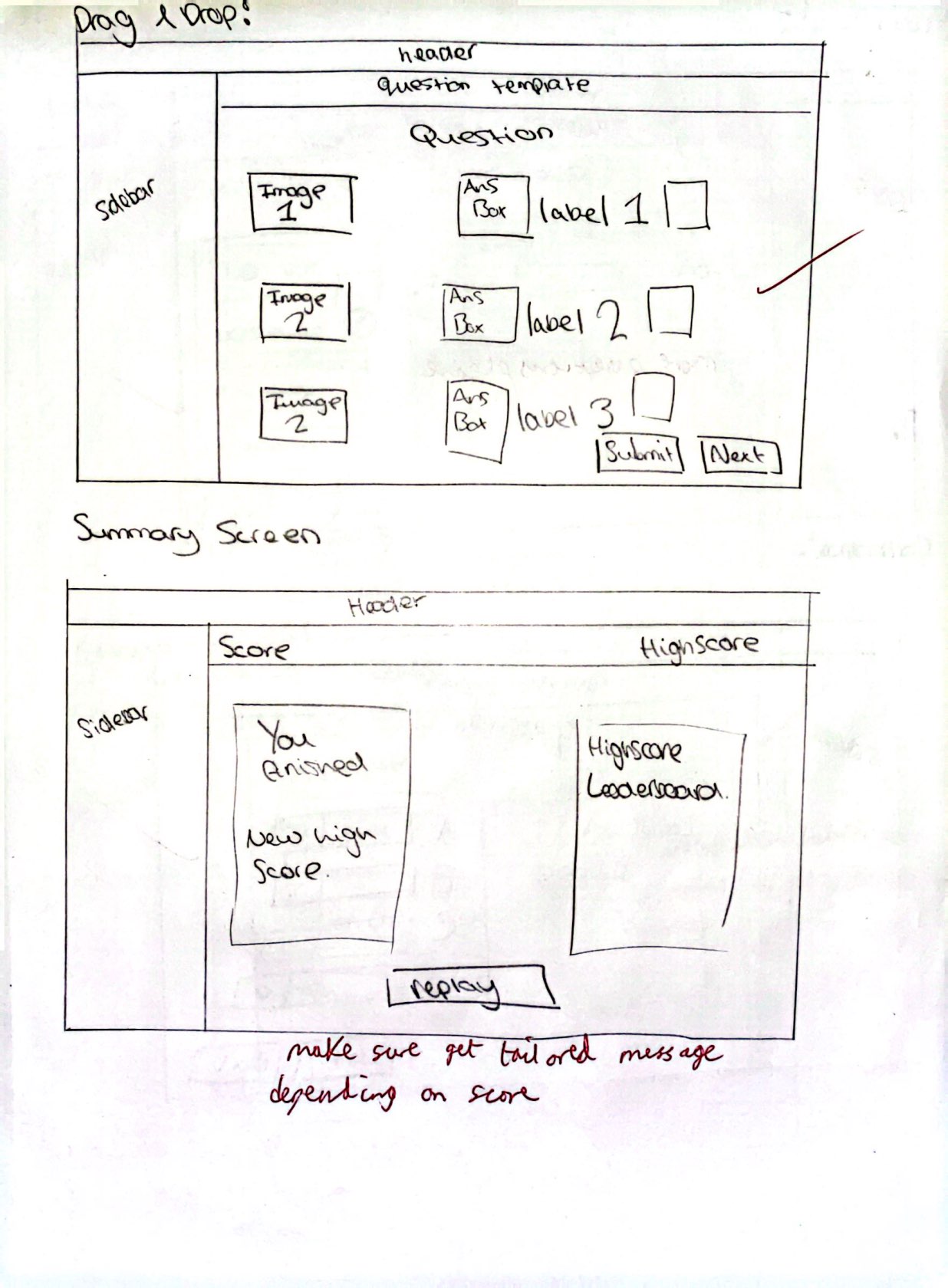
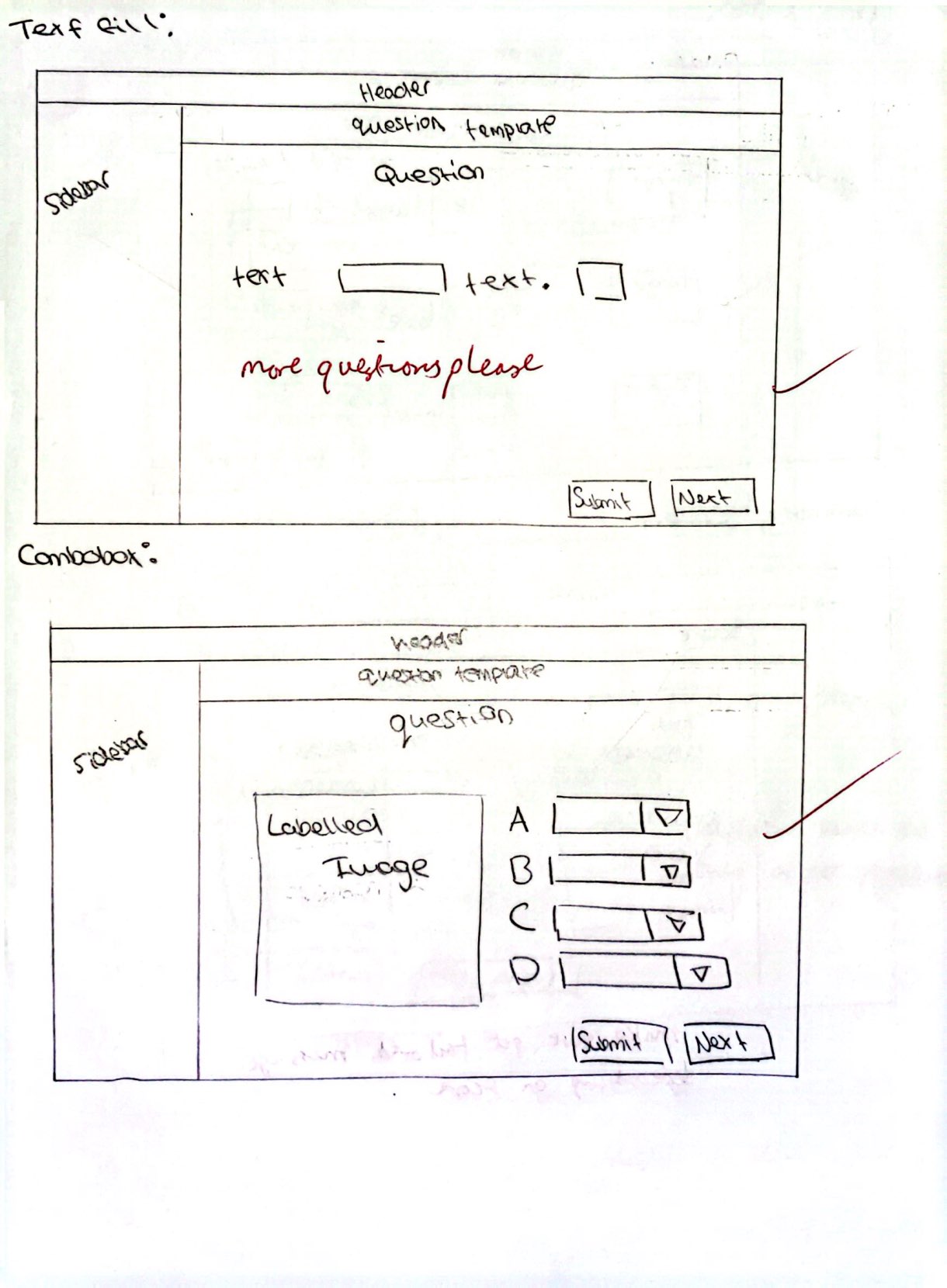
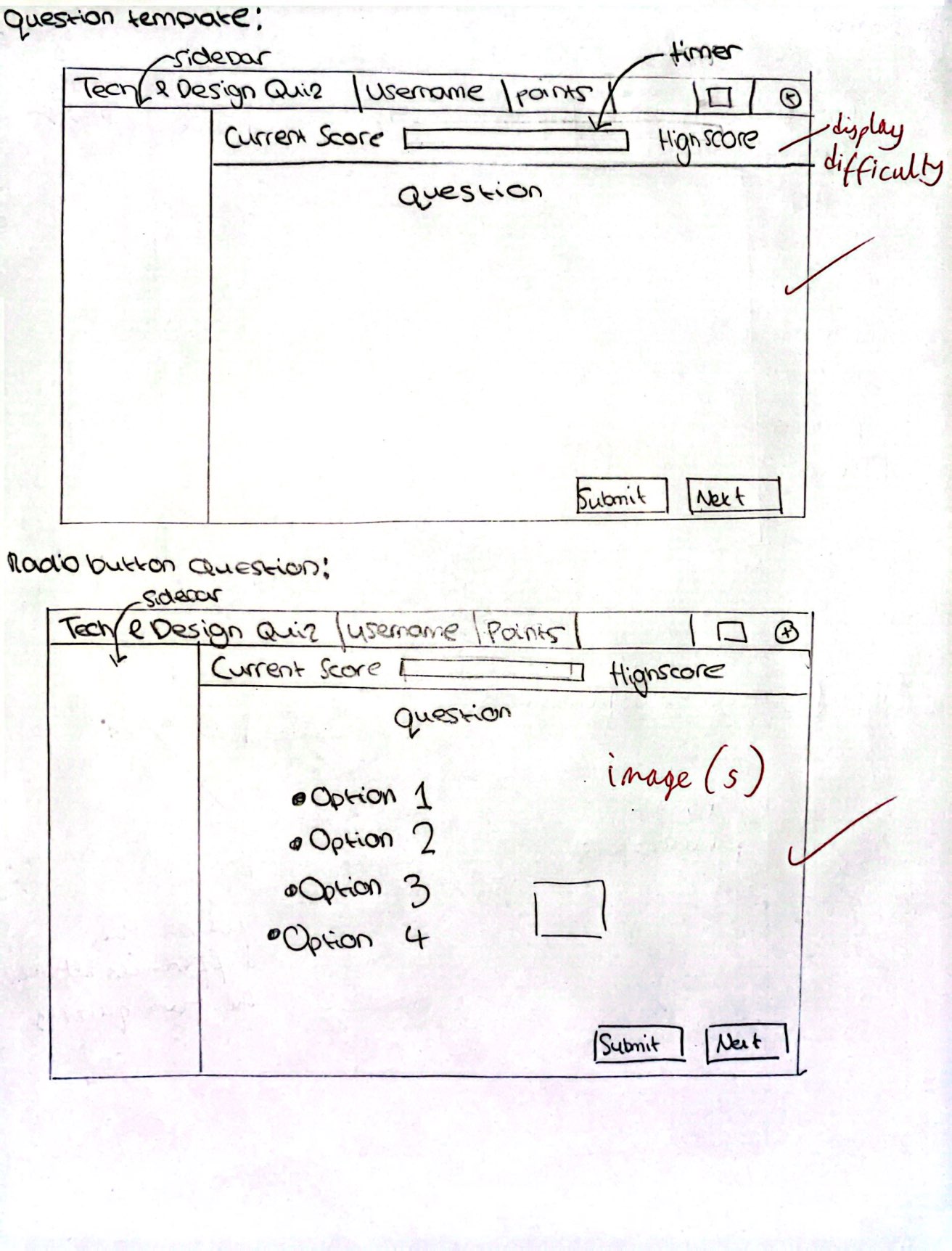
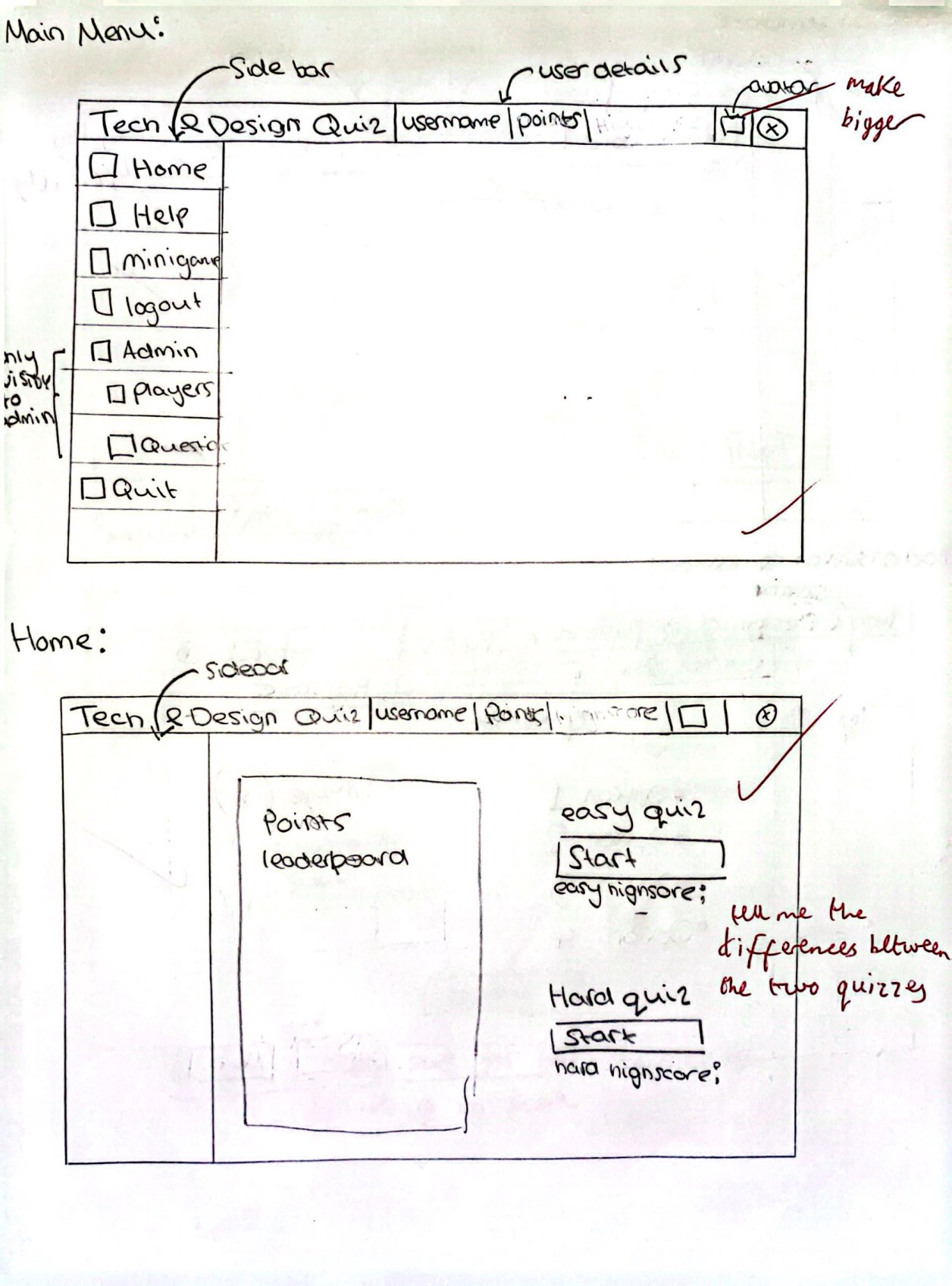
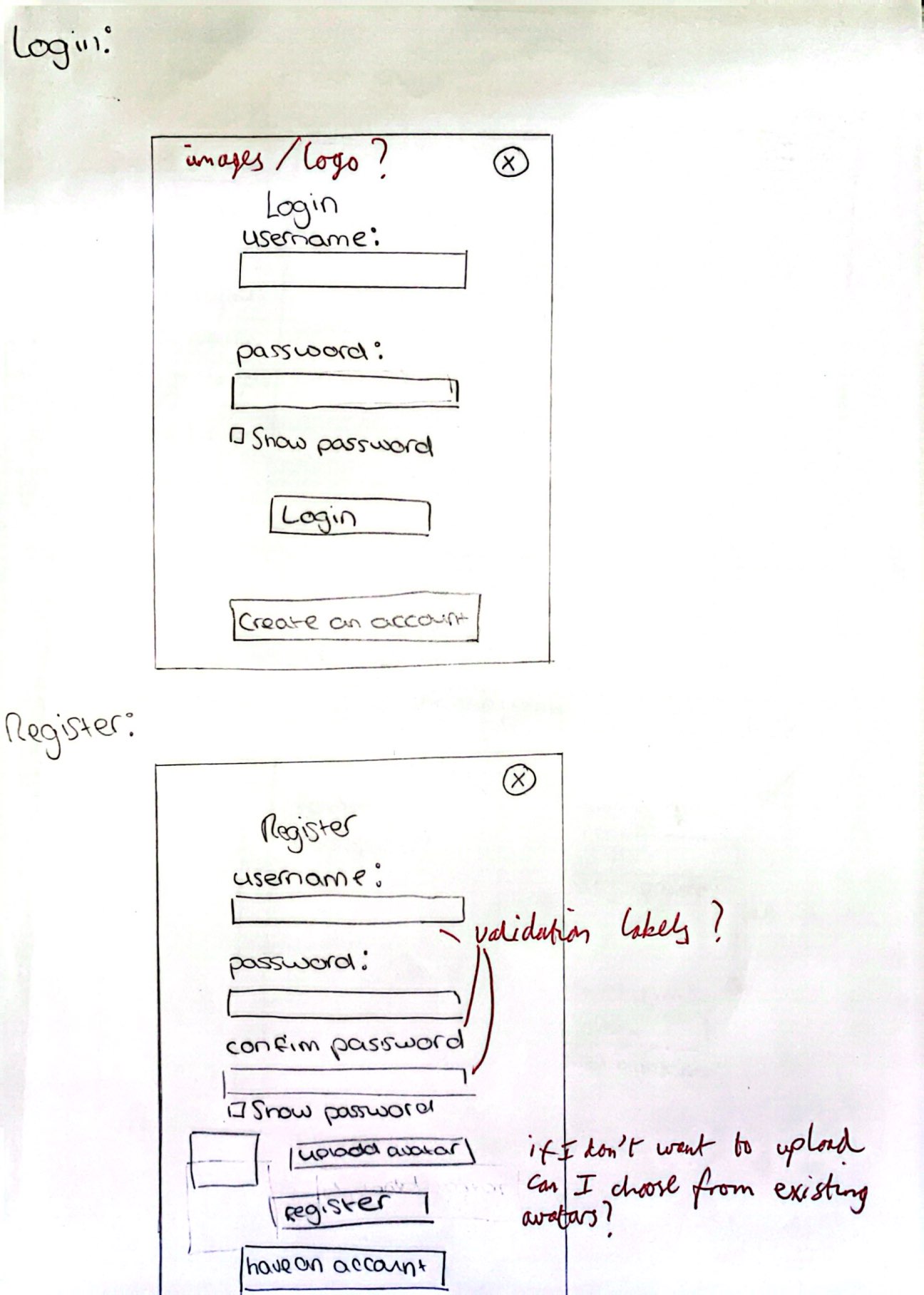
My Technology and Design teacher informed me that some students are finding certain aspects of the GCSE technology and design specification challenging. To support these students, he requested that I design and develop an educational quiz game based on the subject. The game should be enjoyable, easy to use, and engaging, with its primary purpose being an educational tool to help students challenge themselves and build knowledge on the subject. In discussions with my teacher, it was emphasized that the quiz should cater to both struggling and advanced students, potentially including a hard mode. We determined the appropriate difficulty level for the questions, ensuring they align with the current GCSE specification. To encourage continuous engagement, the game will feature competitive elements. I plan to implement a leader board system ranking the players with the top three highest points and the top five high scores, allowing players to view and compare their stats and fostering a sense of competition. This will motivate students to strive for improvement in their scores.

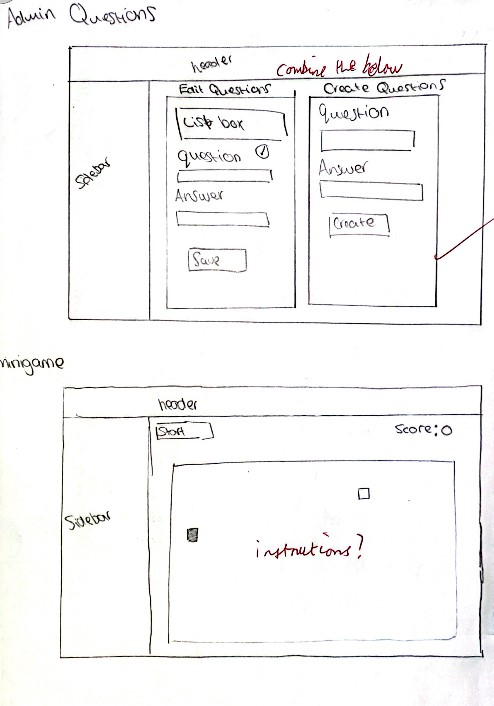
I will be using Microsoft’s Visual Studio 2019 to build the game and it will be written in C#.

# **Requirements**

1. A quiz aimed at GCSE technology and design, it should be fun and engaging.
2. The quiz should have a consistent colour scheme throughout.
3. The quiz should be intuitive and easy to navigate.
4. The quiz will feature two difficulty levels: easy and hard. The hard level will become accessible once the user achieves a certain specified number of points through playing the easy variant.
5. The quiz must have a register and login facility.
6. There should be username and password validation criteria that users must meet when registering.
7. The user should be able to upload an avatar when they are creating an account.
8. Once the user has logged in, they should always be able to log out, quit, or access a help menu.
9. When playing the quiz, some of the questions should not appear more than once.
10. There must be a variation in question types.
11. All question types should be easy to understand; however, instructions should be available to the user.
12. Each question should have a time limit.
13. When the user gives an answer, whether correct or incorrect, they should be clearly notified.
14. Once an answer is submitted, the user should not be able to submit the same answer again.
15. The user should have a score that increases if they give a correct answer, the score should also increase if they answer before the halfway mark.
16. The hard difficulty questions should give the player more points than the easy questions, and a higher score bonus for answering before the halfway mark.
17. There should be an admin facility that can create, edit, and delete users and questions. Admins should also be able to add and revoke other admins.
18. The quiz must display the player’s current score, username, avatar and high score.
19. There should be a leaderboard that displays the top three players.
20. At the end of the quiz, the user should be able to replay without going back to the home page.
21. There should be an unlockable mini game within the quiz.

# **Draft Storyboards**

****

****

# **Final Storyboards**

## Login & Register template

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblTitle | Label | Forecolour: Black | Null | Null |
| B | btnExit | Button | Null | H | 1 |
| C | pbBrand | Picture box | Null | Null | Null |
| D | pnlForm | Panel | Null | Null | Null |

Algorithms:

1.

When pbExit is clicked

If user confirms choice

Close all open forms

## Login

A screenshot of a computer screen

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblLogin | Label | Forecolour: Black | E | Null |
| B | lblUsername | Label | Forecolour: Black | Null | Null |
| C | txtUsername | Textbox | Null | E | Null |
| D | lblPassword | Label | Forecolour:  Black | Null | Null |
| E | txtPassword | Textbox | Null | E | Null |
| F | cbShowPassword | Checkbox | Null | Null | 1 |
| G | btnLogin | Button | Forecolour:  Black | E | 2 |
| H | btnGoToReg | Button | Forecolour:  White | E | 3 |

Algorithms:

1.

On cbShowPassword click

Reveal txtPassword.Text

2.

When btnLogin is clicked

If username and password match a player object from the bin file

Login as that user

Bring player to main menu form

3.

when btnGoToReg is clicked

Close the current form

Open the register form

## Register

A screenshot of a design quiz

Description automatically generated

N

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblRegister | Label | Forecolour: Black | E | Null |
| B | lblUsername | Label | Forecolour: Black | Null | Null |
| C | txtUsername | Textbox | Null | E | Null |
| D | lblPassword | Label | Forecolour:  Black | Null | Null |
| E | txtPassword | Textbox | Null | E | Null |
| F | lblConfirmPassword | Label | Forecolour:  Black | Null | Null |
| G | txtConfirmPassword | Textbox | Null | E | Null |
| H | cbShowPassword | Checkbox | Null | Null | 1 |
| I | pbUploadedAvatar | Picture box | Null | G | Null |
| J | btnUploadAvatar | Picture box | Null | G | 2 |
| K | btnRegister | Button | Null | E | 3 |
| L | btnGoToLogin | Button | Null | E | 4 |
| M | lblValidation | Label | Null | F | 5 |
| N | pbPresetAvatars | Picture box | Null | Null | 6 |

Algorithms:

1.

On cbShowPassword click

Reveal txtPassword.Text

Reveal txtConfirmPassword.Text

2:

When btnUpload is clicked

If image is valid

Let user upload jpeg/png image

Set imgAvatar to uploaded image

Set the player objects avatar to whatever was uploaded

3.

When btnSubmit is clicked

If requirements are met and username hasn’t been taken

Create new player object with the details the user has provided

Add new player object to ‘Players.bin’

Login the user

4.

when btnGoToLogin is clicked

Close the current form

Open the login form

5

If criteria met

Make label green

If not met

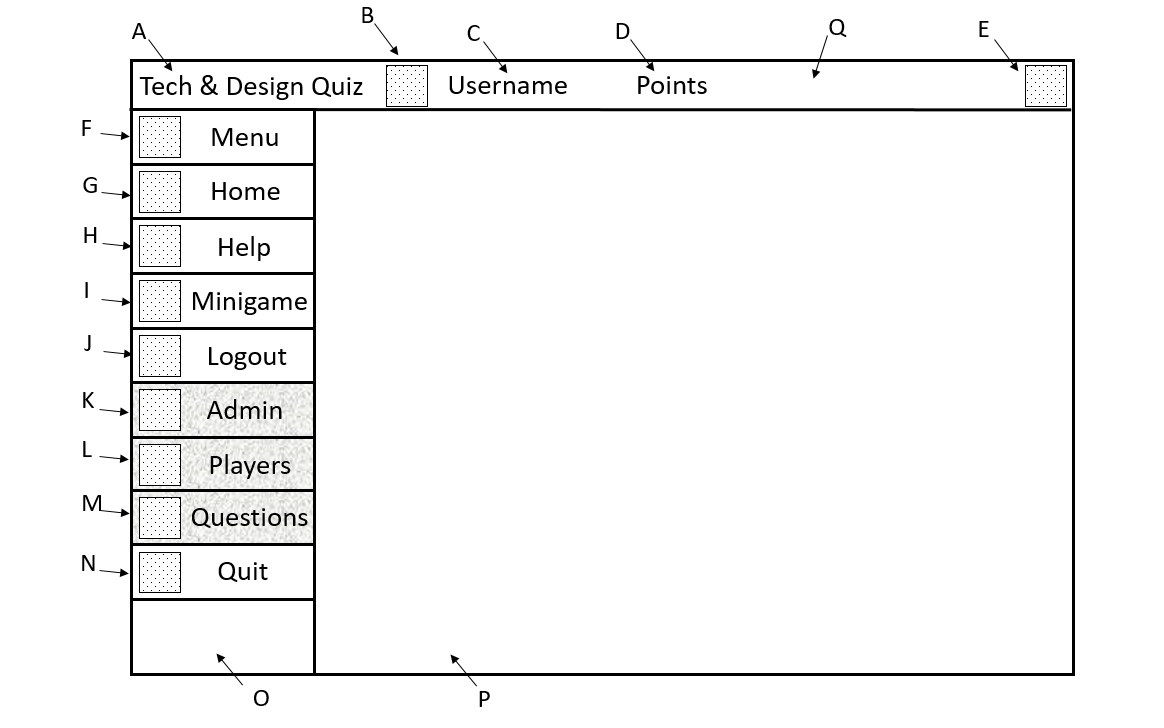
Make label red

6.

On click

Set image in picture box to the player’s avatar

## Main Menu



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblTitle | Label | Forecolour: Black | Null | Null |
| B | pbAvatar | Picturebox | Null | R | 1 |
| C | lblUsername | Label | Forecolour: Black | R | 1 |
| D | lblPoints | Label | Forecolour:  Black | R | 1 |
| E | pbExit | Picturebox | Null | H | 2 |
| F | btnMenu | Button | Forecolour:  White  Backcolour:  Black | C | 3 |
| G | btnHome | Button | Forecolour:  White  Backcolour:  Black | C | 4 |
| H | btnHelp | Button | Forecolour:  White  Backcolour:  Black | C | 5 |
| I | btnMinigame | Button | Forecolour:  White  Backcolour:  Black | C | 6 |
| J | btnLogout | Button | Forecolour:  White  Backcolour:  Black | C | 7 |
| K | btnAdminExpand | Button | Forecolour:  White  Backcolour:  Black | C | 8 |
| L | btnPlayers | Button | Forecolour:  White  Backcolour:  Black | C | 9 |
| M | btnQuestions | Button | Forecolour:  White  Backcolour:  Black | C | 10 |
| N | btnQuit | Button | Forecolour:  White  Backcolour:  Black | C,H | 11 |
| O | pnlSideBar | Panel | Dock: Left | Null | Null |
| P | pnlCurrentForm | Panel | Dock: Fill | C | Null |
| Q | pnlHeadbar | Panel | Dock: Top | C, H | Null |

Algorithms:

1.

On form open

Set labels & picture box to user details

2.

When pbExit is clicked

If user confirms choice

Close all open forms

3:

when btnMenu is clicked

If sidebarExtended is true

Close sidebar

Else if sidebarExpanded is false

Open sidebar

4:

when btnHome is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the home form in the “pnlCurrentForm” panel

5.

when btnHelp is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the help form in the “pnlCurrentForm” panel

6.

When btnMinigames is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the help form in the “pnlCurrentForm” panel

7.

When btnLogout is clicked

If user confirms choice

Return to login form

8.

When btnAdminExpand is clicked

If adminBarExpanded is true

Close adminBar

Else if adminBarExpanded is false

Open adminBar

9.

When btnPlayers is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the admin player form in the “pnlCurrentForm” panel

10.

When btnQuestions is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the admin question form in the “pnlCurrentForm” panel

11.

When btnQuit is clicked

If user confirms choice

Exit the application

## Home

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | tlpLeaderBoard | TableLayoutPanel | 3 rows, 3 columns | S | 1 |
| B | lblEasyHeader | Label | Fore colour: Black | Null | Null |
| C | lblEasyHighscore | Label | Fore colour: Black | R | 2 |
| D | lblEasyInfo | Label | Fore colour:  Black | Null | Null |
| E | btnStartEasy | Button | Fore colour:  Button | D | 3 |
| F | lblHardHeader | Label | Fore colour: Black | Null | Null |
| G | lblHardHighscore | Label | Fore colour:  Black | C | 4 |
| H | lblHardInfo | Label | Fore colour:  Black | Null | Null |
| I | btnStartHard | Button | Fore colour:  Button | Null | 5 |
| J | lblLeaderboardHeader | Label | Fore colour:  Black | Null | Null |
| K | pbLock | Picture box | Image of lock | D | 6 |

Algorithms:

1.

On load

Read all player data

Populate the leaderboard from top score to bottom score.

2.

On load

Populate lblEasyHighscore with the players easy high score.

3.

When btnStartEasy is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the first easy question form in the “pnlCurrentForm” panel

4.

On load

Populate lblEasyHighscore with the players easy high score

5.

When btnStartHard is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the first hard question form in the “pnlCurrentForm” panel

6.

If player score below X

Disable btnStartHard

Make pbLock visible

## Question Template

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblScore | Label | Fore colour: Black | R, O | Null |
| B | pnlHeader | Panel | Null | Null | Null |
| C | QuestionProgressBar | Progress Bar | Null | L | 1 |
| D | lblQuestion | Label | Fore colour:  Black | Null | Null |
| E | lblHighscore | Label | Text = “easy high score” or “hard high score” | R | Null |
| F | btnSubmit | Button | Null | N | 2 |
| G | btnNext | Button | Null | J | 3 |

Algorithms:

1.

On tick

Increase progress bar value

2.

On click

Check if answers are correct

Provide user feedback

Disable all question controls

3.

On click

When btnNext is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the second hard/easy question form in the “pnlCurrentForm” panel

## Radio button

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | pbImage | Picture box | Null | Null | Null |
| B | rbOne | Radio button | Null | J | Null |
| C | rbTwo | Radio button | Null | J | Null |
| D | rbThree | Radio button | Null | J | Null |
| E | rbFour | Radio button | Null | J | Null |
| F | pbFeedback | Picture box | Null | M | Null |

## Drag and drop

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | pbImages | Picture box | Null | J | Null |
| B | pbAnswerBoxes | Picture box | Null | J | Null |
| C | lblAnswerText | Label | Null | J | Null |
| D | pbUserFeedback | Picture box | Null | M | Null |

## Combo box

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | pbImage | Picture box | Null | Null | Null |
| B | lblAnswer | Label | Null | Null | Null |
| C | cbAnswer | Combo box | Null | J | Null |
| D | PbFeedback | Picturebox | Null | M | Null |

## Text fill

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblLeftText | Label | Null | J | Null |
| B | txtAnswer | Textbox | Null | J | Null |
| C | lblRightText | Label | Null | J | Null |
| D | PbFeedback | Picturebox | Null | M | Null |

## Summary screen

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblMessage | Label | Null | Null | Null |
| B | lblInformation | Label | Null | Null | 1 |
| C | lblTailored | Label | Null | Null | 1 |
| D | btnReplay | Button | Null | T | 2 |
| E | pnlLeft | Panel | Null | Null | Null |
| F | lblHeader | Label | Null | Null | Null |
| G | tbLeader | TableLayoutPanel | 5 rows, 3 columns | S | Null |

Algorithms:

1.

If user beat high score

Change lblMessage and lblTailored accordingly

If user scored the same as high score

Change lblMessage and lblTailored accordingly

If user scored less than high score

Change lblMessage and lblTailored accordingly

2.

When btnReplay is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the first easy question form in the “pnlCurrentForm” panel

3.

On load

Read all player data

Populate the leaderboard from top score to bottom score.

## Player editor

A screenshot of a quiz

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lbPlayers | List box | Null | Q | Null |
| B | btnEdit | Button | Null | Q | 1 |
| C | lblUsername | Label | Null | Null | Null |
| D | txtUsername | Textbox | Null | Q | Null |
| E | lblUsernameVal | Label | Null | F | Null |
| F | lblPassword | Label | Null | Null | Null |
| G | txtPassword | Textbox | Null | S | Null |
| H | lblPasswordVal | Label | Null | G | Null |
| I | cbShowPassword | Check box | Null | Null | 2 |
| J | cbAdmin | Check box | Null | Q | 3 |
| K | btnSave | Button | Null | Q | 4 |
| L | btnCreate | Button | Null | Q | 5 |
| M | lblDetailInstructions | Label | Null | M | Null |
| N | lblHeader | Label | Null | Null | Null |
| O | lblHeader2 | Label | Null | Null | Null |

Algorithms:

1.

On click

Enable text boxes

2.

On click

Show/Hide password

3.

On click

Make player admin/not admin

4.

Update the player data and save changes

5.

When btnCreate is clicked

Clear the “pnlCurrentForm” panel of any other forms

Open the adminCreatePlayer form in the “pnlCurrentForm” panel

## Player creator

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblHeader | Label | Null | Null | Null |
| B | lblUsername | Label | Null | Null | Null |
| C | txtUsername | Textbox | Null | Q | Null |
| D | lblUsernameVal | Label | Null | F | Null |
| E | lblPassword | Label | Null | Null | Null |
| F | txtPassword | Textbox | Null | Q | Null |
| G | lblPasswordVal | Label | Null | F | Null |
| H | lblConfirmPassword | Label | Null | Null | Null |
| I | txtConfirmPassword | Textbox | Null | Q | Null |
| J | lblConfirmVal | Label | Null | F | Null |
| K | cbShowPassword | Checkbox | Null | Null | 1 |
| L | btnCreate | Button | Null | Q | 2 |
| M | pbPresetAvatar | Picture box | Null | Q | Null |
| N | pbPresetAvatar1 | Picture box | Null | Q | Null |
| O | btnUploadAvatar | Button | Null | Q,G | 3 |
| P | pbUploadedAvatar | Picture box | Null | Null | Null |

Algorithms:

1.

On click

Show/hide password

2.

On click

Open file

Add player to file

3.

When btnUploadAvatar is clicked

If image is valid

Let user upload jpeg/png image

Set imgAvatar to uploaded image

Set the player objects avatar to whatever was uploaded

## Question Editor

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | lblHeader | Label | Null | Null | Null |
| B | lbQuestion | List box | Null | Null | Null |
| C | lblType | Label | Null | Null | Null |
| D | txtQuestion | Textbox | Null | Q | Null |
| E | txtCorrect | Textbox | Null | Q | Null |
| F | txtIncorrect1 | Textbox | Null | Q | Null |
| G | txtIncorrect2 | Textbox | Null | Q | Null |
| H | txtIncorrect3 | Textbox | Null | Q | Null |
| I | btnDeselect | Button | Null | Q | 1 |
| J | btnAdd | Button | Null | Q | 2 |
| K | btnEdit | Button | Null | Q | 3 |
| L | btnSave | Button | Null | Q | 4 |
| M | btnRemove | Button | Null | Q | 5 |
| N | lblHeader2 | Label | Null | Q | Null |
| O | lblInfo | Label | Null | Null | Null |
| P | pnlInfo | Panel | Null | Null | Null |
| Q | pnlEditor | Panel | Null | Null | Null |

Algorithms:

1.

On click

Deselect picked question from list box

Clear text boxes

2.

On click

Gather text boxes text and write to player object

3.

On click

Enable text boxes

4.

On click

Update question object

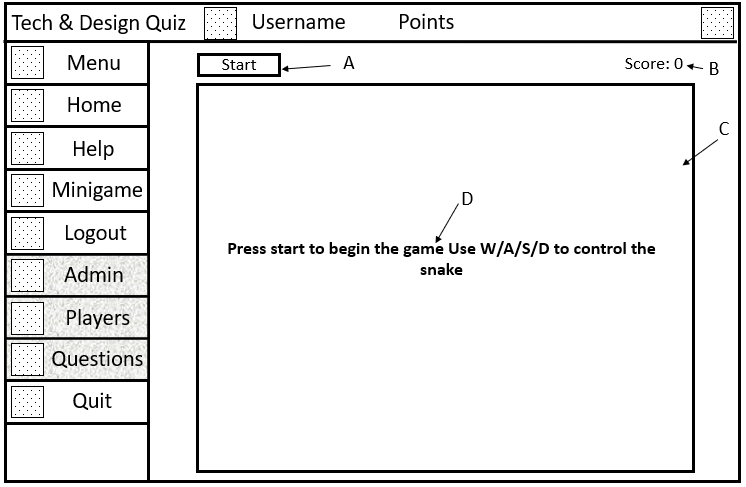
Write to bin file

5.

On click

Remove player object from bin file

## Mini game – snake



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Letter | Control Name | Control Type | Properties | User Requirement | Algorithm |
| A | btnStart | Button | Null | U | 1 |
| B | lblScore | Label | Null | U | Null |
| C | pbPlayableArea | Picturebox | Null | U | Null |
| D | lblInfo | Label | Null | U | Null |

Algorithms:

1.

On click

Generate a random piece of food

Start moving the snake

Make lblInfo invisible

# **Testing**

## Introduction

For the testing of my system, I will be ensuring that every part of it works accordingly. However, for things such as the quit, logout and help buttons that are present on all forms, I will only test them once.

## Testing Map

|  |  |
| --- | --- |
| Test subject: | Test No |
| * Register and login |  |
| * + Ensuring both register and login facilities are present and accessible | 1 |
| * Register form |  |
| * + Validations |  |
| * + - Fill all fields. | 2 |
| * + - Username checks | 3 |
| * + - Custom avatar | 4 |
| * + - Password checks | 5-7 |
| * + Duplicate user checks | 8-10 |
| * Login form |  |
| * + Fill all fields | 11 |
| * + Correct username & password checks | 12-13 |
| * All forms |  |
| * + Ensuring a user can logout, quit or exit on all forms. | 14 |
| * + Help menu accessible | 15 |
| * All question forms |  |
| * + Ensuring the user gets score, and gets more score on a hard question. | 16 |
| * + User should get more score if they answer faster. | 17 |
| * + User feedback on correct/incorrect answers | 18 |
| * + Timer checks | 19-21 |
| * + Ensuring the questions are applicable to tech and design GCSE students. | 22 |
| * + Duplicate submission check | 23 |
| * + Progress saved check | 24 |
| * Main Menu form |  |
| * + Displaying user details check. | 25 |
| * + Sidebar functionality check. | 26-27 |
| * Question forms |  |
| * + Input box form |  |
| * + - Ensuring answers are not case sensitive | 28 |
| * + Drag and drop form |  |
| * + - Functionality checks | 29-31 |
| * + Combo box form |  |
| * + - Ensuring items are shuffled | 32 |
| * Quiz gameplay checks |  |
| * + Randomisation | 33-34 |
| * + Unlocking hard difficulty | 35 |
| * + Question type check | 36 |
| * End screen form |  |
| * + Tailored message | 37-38 |
| * + Replay check | 39 |
| * + Timer check | 40 |
| * Mini game form |  |
| * + Ensuring game functions | 41-48 |
| * Admin facility | 49 |
| * + Question editor – creating questions. | 50-53 |
| * + Player editor – editing and creating players | 54-60 |
| * Leader boards |  |
| * Ensuring correct order of leader boards | 61--63 |
| * Help forms – ensuring accessibility | 64 |
| * UI – colour scheme and navigation | 65-66 |
| * Ensuring admin regeneration and question regeneration, if bin files are deleted. | 67-68 |

## Testing table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Form | Reason for Test | Requirement | User Action/  Test Data | Expected  Outcome | Pass?  T/F | Corrective  Action |
| 1 | Register and login | Ensuring a login and register facility is present. | E | N/A | After the system has loaded the login and register should be accessible. | T | N/A |
| 2 | Register form | To check that the user can’t continue without filling all fields. | F | “administrator”, in the password textbox and leave username empty and vice versa. | The submit button will remain disabled if no username/password/ avatar is selected/entered. | T | N/A |
| 3 | Register form | To check the user can’t register if the username is not greater than 5 characters. | F | “e”, in the username textbox and “administrator” in the password & confirm password textbox on the register page. | The register button will remain disabled because the length requirement is not met. The validation label should stay red under the username text box. | T | N/A |
| 4 | Register form | To check the user can upload a custom avatar. | G | Click the upload avatar button. Select an image from your files. | File dialog should open allowing the user to select an image from their file system. This photo should now be shown in the picture box. | T | N/A |
| 5 | Register form | To check the user can’t register if the password is not greater than 7 characters. | F | Type “administrator”, in the username textbox and “e” in the password & confirm password textbox on the register page. | The register button will remain disabled because the length requirement isn’t met. The validation label under the password should stay red. The matching passwords validation label should be green. | T | N/A |
| 6 | Register form | To check the user can’t register if the password doesn’t contain a number. | F | “administrator”, in the username textbox and “administrator” in the password textbox on the register page. | The register button will remain disabled because the password doesn’t contain a digit. The validation label under the password should stay red. | T | N/A |
| 7 | Register form | The Register button must remain disabled until the both the password and confirm password match. | F | “administrator1”, in the password textbox and “administrator2” in the confirm password textbox. | The register button will remain disabled. The validation label under the confirm password textbox should stay red. | T | N/A |
| 8 | Register form | The user should be notified if a player with the same username has already been registered. | F | “admin”, in the username textbox and “testing” in the password textbox. | A message box should appear, notifying the user that an existing player with the same username already exists. | T | N/A |
| 9 | Register form | Players must be able to use register an account with a username that was previously deleted. | N/A | Go to the admin menu, delete a user, log out, register an account with the same username. | Account will be created, with the same username, but no other statistics from the older user. | T | N/A |
| 10 | Register form | Ensure a new user can be registered after the “players.bin” file is deleted. | N/A | Delete the players.bin file. | When the register form is opened if a players.bin file is not found one will automatically be created. | T | N/A |
| 11 | Login form | To check that the user can’t continue without filling all fields. | F | “administrator”, in the password textbox and leave username empty and vice versa. | The submit button will remain disabled if no username/password is entered | F | Ensure submit button is disabled until all fields have been entered |
| 12 | Login form | Ensure users will be notified if the username they entered does not exist. | N/A | “example” entered in the username and password textbox. | A message box should appear, notifying the user that this account doesn’t exist. | T | N/A |
| 13 | Login form | Trying to log in with username that does exist but wrong password. | N/A | Enter “testing” in the username textbox and “123” entered in the password textbox. | A message box should appear, notifying the user that the username or password is incorrect. | T | N/A |
| 14 | All forms apart from login, register. | Ensuring the user can logout, quit, exit at any point after logging in. | H | Enter “admin” into the username text box and “admin” password text box on the login screen. | If logout clicked open form should close, and the login page should open. If exit/quit clicked. Application should close. | T | N/A |
| 15 | All forms apart from login, register. | Ensuring the user can view the help page at any point after logging in. | K | Help button Clicked. | The help form should open. | T | N/A |
| 16 | All question forms | Ensuring the user will get score when answering a question, and will get more score when answering a hard question than an easy question. | P, O | Answer a hard question and easy question both at the same point on the timer so that the time won’t affect the score. | The user should receive double the points answering the hard question than the easy question. Answer the question correctly and the user should receive points. | T | N/A |
| 17 | All question forms | Ensuring the user will get more points answering below the half way mark on the timer(both difficulties) . | P | Submit an answer before the half way mark | User should get 2 extra points on the easy mode and 3 extra points on hard mode | T | N/A |
| 18 | All question forms | The user should be notified in a clear way if they give the correct/incorrect answer. | M | Click “Submit” button with the correct answer inputted. Then repeat but with an incorrect answer. | For correct answers a tick will appear and an X for incorrect answers. | T | N/A |
| 19 | All question forms | If the user doesn’t submit an answer and the quiz times out, they shouldn’t receive score. | N/A | Wait until timer times out, without pressing submit. | No score should be awarded to user and X’s should appear beside the questions. | T | N/A |
| 20 | All question forms | Questions should have a time limit, if time runs out they should be alerted. | L | Start the quiz and wait for the time to run out. | The user should gain 0 points for that question and they will be notified that they ran out of time via a custom message box. | T | N/A |
| 21 | All question forms | Questions should have a ticking noise during the last three seconds of the timer. | L | Start the quiz and wait until the last three seconds. | A ticking noise should play for the duration of three seconds. | T | N/A |
| 22 | All question forms | The quiz must have appropriate questions that are suitable for the target audience. | A | Login to the application and start the quiz. | There should be suitable questions for GCSE technology and design students. | T | N/A |
| 23 | All question forms | The user should not be able to submit an answer twice. | N | Answer a question and press submit. | Submit button should be disabled after the initial click. | T | N/A |
| 24 | All question forms | If the quiz is active, and the user presses a sidebar button they should be alerted that progress won’t be saved. | N/A | Start the quiz, and press a side bar button | Should be a prompt alerting the user that progress won’t be saved. | T | N/A |
| 25 | Main menu-header | When the users finish the quiz, their points on the top of the screen should update accordingly. | R | Finish the quiz with some score. | Points should be updated with the previous amount of points + the new score. | T | N/A |
| 26 | Main menu- sidebar | When the user presses a sidebar button, and a quiz isn’t active. It should navigate them to the respective form. | C | Press the sidebar button after logging in. | Screen will change form. | T | N/A |
| 27 | Main menu-sidebar | Ensuring reactivity with the sidebar. | N/A | Click the burger menu on the top left of the form. | The side bar should expand and compress dependent on its previous state. | T | N/A |
| 28 | Input box question form | Answers provided by the user should not be marked case sensitive or sensitive to spaces “ “. | N/A | Input right answer for a text box question eg;  A\_\_\_\_ is a tool   * pLaNe | The user should still be marked correct. | T | N/A |
| 29 | Drag and drop question form | If the user clicks the reset button the form should be reset allowing the user to change their answer. | N/A | Click the reset button. | Any pictures that were dragged and dropped should be reset to their initial position. | T | N/A |
| 30 | Drag and drop question form | If the user pressed the submit button, can they still reset? | N/A | Click the submit button. | Reset button should be disabled. | T | N/A |
| 31 | Drag and drop question form | If the user moves their answer from answer box to another answer box, are they still marked correct? | N/A | Select a picture and move it to a answer box, then shuffle it between the answer boxes. | Correct score and user feedback should be awarded. | T | N/A |
| 32 | Combo box question form | Are the order of the items in the combo box shuffled when quiz is replayed? | N/A | Play the quiz twice. | Combo box items should be in a different order | T | N/A |
| 33 | Quiz gameplay | Ensuring radio button questions are randomized and that the user should be notified via a message box when they run out of radio button questions. | I | Play the quiz multiple times exhausting all the radio button questions. | A message box should appear telling the user that all questions have been exhausted and they may will see repeats from this point forward. | T | N/A |
| 34 | Quiz gameplay | Questions should become available again when the user logs out or exits. | I | Play the quiz and exhaust every question bar one. | When you log back in, questions should be reset | T | N/A |
| 35 | Quiz gameplay | The hard game mode must become available once the account reaches 20 points. | D | Reach 20 points on the easy variant. | The lock on the hard button should disappear and the hard button should enable. | F | I changed the equality signs in the code. |
| 36 | Quiz gameplay | There should be 4 different question types. | J | Play the quiz on easy mode. | When playing the quiz, there should be 4 different question types, Radio button, Drag and drop, Combo box, Input box. | T | N/A |
| 37 | End screen form | To check the tailored message displayed at the end of the quiz changes relative to what the user scored against their high score. | N/A | Play the quiz initially.  Beat your high score.  Score the same as your high score.  Score less than your previous high score. | Message should change accordingly. Dependent on the scenario. | T | N/A |
| 38 | End screen form | To check the new high score / You scored the same as / You scored less. Title changes accordingly. | N/A | Play the quiz and beat your high score, score the same as your previous high score, score less than your high score | Message should change accordingly. Dependent on the scenario. | T | N/A |
| 39 | End screen form | To check that the replay button navigates the user to the correct question difficulty, and their score is refreshed. | T | Finish the quiz, press replay. | On the easy quiz, when you press replay, it should relocate you to the first easy question, on the hard quiz, it should be the first hard question. | F | The hard replay button navigated to the first easy question. To fix this I changed the form that was being opened. |
| 40 | End screen form | Ensure the ticking noise doesn’t play on the end screen, as it does inherit from the question template. | N/A | Finish the quiz and wait 25 seconds | No sound should play on the summary form. | F | I disabled the sound on the end screen. |
| 41 | Mini game - Snake | The game should only start after the start button is pressed. | N/A | Click on the “Start” button. | The snake should start to move. | T | N/A |
| 42 | Mini game - snake | Before the game starts, the player should be told the controls. | U | View the game, before pressing start. | Instructions should appear telling the player the controls. | T | N/A |
| 43 | Mini game - snake | The game should end when the snake hits the sides of the form. | U | Click on the “Start” button and direct your snake to the edge of the form. | A game over message should display and the snake should stop moving. | T | N/A |
| 44 | Mini game - snake | The game should end when the snake hits itself. | U | Click on the “Start” button and direct your snake in a circle so it touches its own tail. | A game over message should display and the snake should stop moving. | T | N/A |
| 45 | Mini game - snake | The score should increase by 1 every time the snake eats a white square. | U | Click on the “Start” button and direct the snake to the white squares. | The score at the top of the form should increase by 1 point. | T | N/A |
| 46 | Mini game - snake | When the snake eats a white square, it should disappear, and a new one should appear in a random location on the form. | U | Click on the “Start” button and direct the snake to the white squares. | When the snake eats a white square, it should be removed, and a new white square should appear elsewhere. | T | N/A |
| 47 | Mini game - snake | When the snake eats a white square, their length should increase. | U | Click on the “Start” button and direct the snake to the white squares. | When the snake eats a white square, it should be added on to the length of the snake. | T | N/A |
| 48 | Mini game - snake | The snake should not die unless it hits the edges or itself. | U | Click on the “Start” button and direct the snake around the playable area. | The snake should not die unless. It hits the edges or itself. | F | The snake died in the middle of the playable area. I made the playable area smaller, preventing unknown deaths. |
| 49 | Admin facility | Ensure the admin facility is not accessible to users without admin status. | Q | Login as a regular user, if unsure of credentials, create an account. | The admin facility should not be visible on the sidebar. | T | N/A |
| 50 | Question editor form | To check admins can view all the questions created for the quiz. | Q | N/A | All the questions should be displayed in a scrollable list box. | T | N/A |
| 51 | Question editor form | To check the admins can create new questions for the quiz. | Q | Populate all text boxes and press the add button | The question should be added to the questions.bin file. | T | N/A |
| 52 | Question editor form | To check a question can’t be created if any fields are left empty. | Q | Populate just the question text box with “Hello”. | The add button should be disabled. | T | N/A |
| 53 | Player editor form | To check users can view all the existing accounts that have been registered. | Q | N/A | All the players should be displayed in a scrollable list box. | T | N/A |
| 54 | Player editor form | To ensure players can be deleted from the admin page. | Q | Select a player from the list box and click the “Remove” button | The selected player should be removed from the players bin file as well as the admin players’ form. | T | N/A |
| 55 | Create player form ( Admin ) | To ensure new account can be successfully created. | Q | On the player editor form, press create a player, type “example67” in the username box, type “example67” In the password box, type “example67” in the confirm password box, and select an avatar. | The account should be created and should now be viewable from the admin player editor form. | T | N/A |
| 56 | Create player form ( Admin ) | To ensure the requirements needed for registering on the register form are also present when creating a player from the create player form. | Q | Type “example67” in the username text box and “example” in the password and confirm password box. | The register button should be disabled and two validation labels should be red, password must include a number & passwords don’t match. | T | N/A |
| 57 | Edit player form | To ensure accounts can be successfully edited. | Q | Select a player from the list box, press the edit button, make changes and press save. | The edited account should have the changes that you made. | T | N/A |
| 58 | Edit player form | To ensure the requirements needed for registering on the register form are also needed to edit a player from the edit player form. | Q | Type “example” in the username text box and “example” in the password and confirm password box then press the submit button. | The user should be notified that the password must contain a digit. | T | N/A |
| 59 | Edit player form | Ensure admins can grant admin permissions to other users. | Q | Select a user, press edit, check the admin checkbox, press save. | When logging into the edited account you should see the admin facilities on that account. | F | Correctly updated the player object setting the admin attribute to true. |
| 60 | Edit player form | Ensure no changes can be made to the master admin. | Q | Select the account with the username “admin” from the list box. | An alert should show saying you cannot edit this account, and all textboxes should be disabled. | T | N/A |
| 61 | Leaderboard – main menu | The Leaderboard should display the top 3 players ordered by points | S | Log into the game, take note of the leaderboard on the main menu, gain points on an account, | Watch the account that you are signed into go up on the leaderboard. | T | N/A |
| 62 | Leaderboard – easy quiz | The easy high score leaderboard should display the top 5 players ordered by easy high score. | S | Increase easy high score on an account. | Watch the accounts username go up on the leaderboard. | T | N/A |
| 63 | Leaderboard – hard quiz | The hard high score leaderboard should display the top 5 players ordered by hard high score. | S | Increase hard high score on an account. | Watch the accounts username go up on the leaderboard. | T | N/A |
| 64 | Help forms | User should be able to get information on how to answer all questions. | K | Open help menu on side bar, and press each button on the right. | Each button updates the labels and images, depending on the pressed button. Providing instructions to the user. | T | N/A |
| 65 | UI | Ensuring the quiz has a consistent color scheme throughout. | B | N/A | The quiz should maintain a similar color scheme throughout. | T | N/A |
| 66 | UI | Ensuring the quiz has an easy to navigate interface for users. | C | N/A | Buttons should have clear text and | T | N/A |
| 67 | An admin should always be present | When players.bin file is deleted, a new admin account should be generated. | N/A | Delete players.bin file, start the system, type username: admin password: admin, press submit, | Admin account should be found, as It was created on load of the login form. | T | N/A |
| 68 | When there are no questions, the user shouldn’t see a blank screen. | Ensuring the quiz is still navigable when there is no question available. | N/A | Delete questions.bin file, log in and go to the first easy question. | The form should not crash. | F | Added a method that skips the user to the next question, if there are no questions in the bin file. |

## Failed Tests

**Test 11:** To check that the user can’t continue without filling all fields, on the login form.

There was no previous logic to handle this.

I added logic to the text changed event on both txtUsername and txtPassword, ensuring both fields have text in them before they are enabled.

A screenshot of a computer

Description automatically generated

**Test 35:** The hard game mode must become available once the account reaches 20 points.

The equality signs were opposite to what they were meant to be.

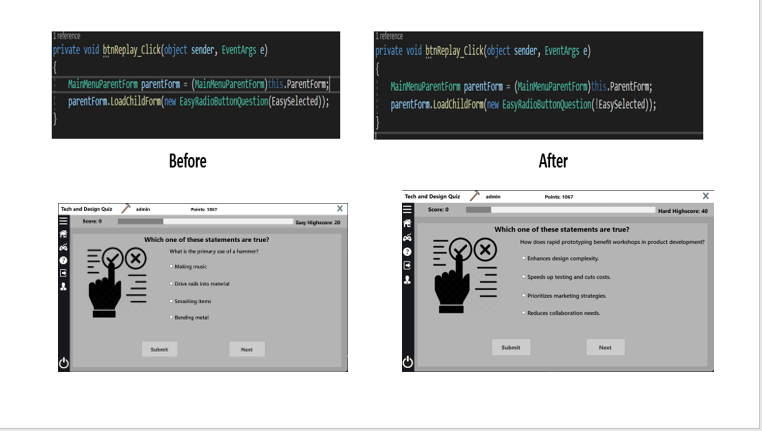
I switched the equality signs. A screenshot of a computer

Description automatically generated

**Test 39:** To check that the replay button navigates the user to the correct question difficulty, and their score is refreshed.

The Boolean value was hard coded, this made it so that when the replay button was pressed, it always sent the user to the easy variant.

Correctly changed the Boolean value that is passed through to the first form when the replay button is pressed, due to the fact that both of my difficulties are using the same form I just needed to ensure the same Boolean value was passed was correct.



**Test 40:** Ensure the ticking noise doesn’t play on the end screen, as it does inherit from the question template.

I didn’t have it coded so that It didn’t play sounds when it didn’t need to.

A screenshot of a computer program

Description automatically generatedI added a method to stop the timer, which stops the noise from playing.

**Test 48:** The snake should not die unless it hits the edges or itself.

The code was not filling the entire picture box as playable area.

I made the picture box (playable area) smaller.

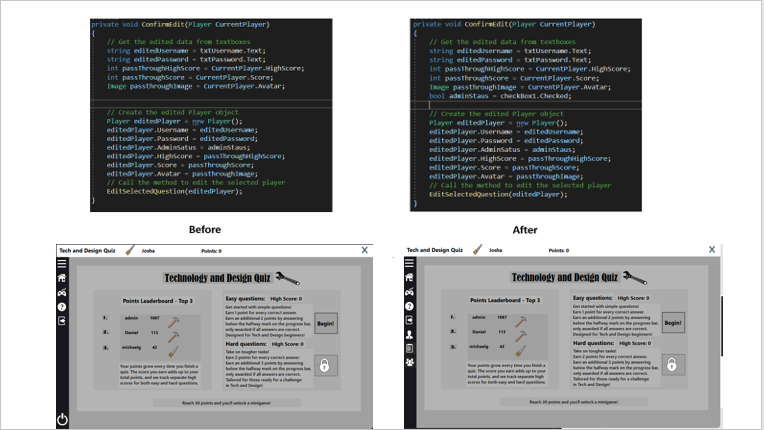
A screenshot of a computer

Description automatically generated

**Test 59:** Ensure admins can grant admin permissions to other users.

I didn’t have the code to gather the status of the check box.

I ensured that the check boxes value was used to determine if the user was an admin or not, allowing changes to be made to their admin status.



**Test 68:** To check the form functions when questions.bin file is deleted.

There was no previous logic to handle this.

I added logic to handle this scenario.

A screenshot of a computer

Description automatically generated

# **Evaluation**

## User Requirements Evaluation

1. A quiz aimed at GCSE technology and design, it should be fun and engaging. – Fulfilled.

The quiz has been developed with a focus on GCSE technology and design, incorporating a variety of topics such as manufacturing, naming tools, labelling tools/equipment. To ensure it remains enjoyable and engaging for the user, features such as unlockable mini games where added. By integrating diverse content and interactive elements, the quiz effectively captures the interest of students, making the learning process both fun and educational. The quiz was noted as “fun and engaging” via peer feedback; this further supports the fact that this requirement was fulfilled.

**See test: 22.**

1. The quiz should have a consistent colour scheme throughout. – Fulfilled

A consistent colour scheme has been applied across all elements of the quiz, the quiz follows a grey/white/light-brown colour scheme enhancing the visual appeal and user experience. This design choice ensures a professional and friendly look that is pleasing to the eye, aiding in user concentration and engagement. Such a design decision is crucial for provoking word-of-mouth; an aesthetically pleasing app is more likely to be talked about and recommended.

**See test: 65.**

1. The quiz should be intuitive and easy to navigate. – Fulfilled

The quiz features a meticulously crafted user interface, ensuring that each button is clearly labeled, accompanied by an intuitive icon or both. Strategic placement of controls aligns with user expectations, enhancing the overall navigational experience. The choice of Segoe UI for the text font prioritizes readability, further facilitating ease of navigation. An intuitive UI is crucial, as it streamlines the user's interaction with the game, making it more enjoyable and less daunting. This user-friendly design is key to preventing frustration and ensuring that players are encouraged to continue engaging with the game, rather than abandoning it due to complexity or poor design.

**See tests: 26, 66.**

1. The quiz will feature two difficulty levels: easy and hard. The hard level will become accessible once the user achieves a certain specified number of points through playing the easy variant. – Fulfilled

The implementation of two difficulty levels, easy and hard, enriches the quiz experience. Users are challenged according to their skill level, with the hard level unlocking after achieving a specified number of points, providing a motivating progression system. This progression system keeps users engaged, as they have a clear goal to work towards, which can increase the time spent interacting with the quiz. The transition from easy to hard questions encourages users to learn and retain information, effectively turning the quiz into an educational tool.

**See test: 35.**

1. The quiz must have a register and login facility. – Fulfilled.

This has been achieved by implementing a registration and login system, enabling users to create accounts, and login to them. This feature not only personalizes the experience but also allows for tracking progress and scores, adding a layer of engagement to the quiz.

**See test: 1.**

1. There should be username and password validation criteria that users must meet when registering. – Fulfilled

Validation criteria for usernames and passwords has been implemented, this ensures security and uniqueness of user accounts; the user cannot register an account of a duplicate username, and the user cannot register until their password is 7+ characters, and their password includes a number. These measures prevent unauthorized access and enhance the overall security of the user's information. These measures prevent unauthorized access and enhance the overall security of the user's information.

**See tests: 2-8, 11.**

1. The user should be able to upload an avatar when they are creating an account. – Fulfilled

When the admin or user wants to make an account, they can press a button called “Upload avatar” this button navigates the user to their files and allows them to upload an image to be used as their avatar, granted it’s the correct file type. This feature allows the user’s account to feel more personal, enhancing sense of ownership.

**See test: 4.**

1. Once the user has logged in, they should always be able to log out, quit, or access a help menu. – Fulfilled

Implementing an always-accessible sidebar that can expand and collapse provides users with continuous access to essential functions like logout, quit, and help options, significantly enhancing the user experience. This feature ensures that users feel in complete control of their quiz journey, with the ability to easily navigate through the platform. The expandable sidebar, which reveals labels next to icons upon expansion, offers clear guidance on what each button does, eliminating any guesswork and making the interface intuitive and user-friendly. Such thoughtful design promotes a seamless interaction with the quiz, allowing users to focus on the content without being hindered by navigational challenges. This level of accessibility and ease of use not only increases user satisfaction but also encourages longer and more engaging sessions within the quiz platform.

**See test: 14.**

1. When playing the quiz, some of the questions should not appear more than once. – Fulfilled

An admin can create and edit radio button questions, this creates a diverse array of questions, the created questions are displayed randomly, and are not shown again until all questions have been exhausted. By enabling admins to create and edit a diverse pool of questions that are displayed randomly, the quiz maintains its freshness and appeal across multiple plays. This approach significantly enhances the quiz's replay value and educational impact, as users are exposed to a wide variety of questions, encouraging comprehensive learning and retention without the drawback of repetition.

**See tests: 33, 34.**

1. There must be a variation in question types. – Fulfilled

Incorporating four distinct question types—radio button, combo box, drag and drop, and text box—each displayed on its unique form, diversifies the quiz experience. This variety prevents the interface from becoming monotonous and keeps the engagement levels high, as users encounter different formats and challenges. Moreover, options like radio buttons offer users a chance to answer correctly through educated guesses, enhancing both the learning experience and the quiz's appeal by balancing challenge with accessibility.

**See test: 36.**

1. All question types should be easy to understand; however, instructions should be available to the user. – Fulfilled

While maintaining simplicity in question design, comprehensive instructions are available in the help menu- accessible via the side bar. The help menu is not accessible during the quiz as I believe this would be an opportunity to cheat, users could read the question, and open the help menu, which would pause the quiz, and therefore give the users an opportunity to think of an answer. Therefore, voiding the timer.

**See tests: 15, 64.**

1. Each question should have a time limit. – Fulfilled

Setting a time limit—15 seconds for radio button, drag and drop, and combo box questions, and 25 seconds for text entry questions—introduces a dynamic pace to the quiz. Displayed through a progress bar, the time constraint encourages users to focus and respond swiftly, enhancing engagement and cognitive processing speed. The immediacy of having to answer before time expires ensures that users remain attentive and actively participate, making the quiz not only a test of knowledge but also of quick thinking and time management.

**See tests: 20, 21.**

1. When the user gives an answer, whether correct or incorrect, they should be clearly notified. – Fulfilled

The users are presented with immediate feedback when the submit button is clicked, a tick or x icon is placed beside their answer, and the users score will increase if they get an answer right. This consistent and clear feedback mechanism ensures users are always informed of their progress and performance, significantly enhancing the learning experience by reinforcing correct answers and identifying areas for improvement.

**See test: 18.**

1. Once an answer is submitted, you should not be able to submit the same answer again. – Fulfilled

When the submit button is pressed, the users answer is marked and the submit button is disabled. This ensures each question attempt is meaningful and encourages thoughtful participation, enhancing the educational value of the quiz. Crucially, this mechanism prevents users from attempting to brute-force their way to the correct answer through repeated submissions, preserving the quiz's integrity and educational purpose by fostering a genuine learning experience.

**See tests: 23.**

1. The user should have a score that increases if they give a correct answer, the score should also increase if they answer before the halfway mark. – Fulfilled

The scoring system is thoughtfully designed to reward users for correct answers and efficiency. On easy difficulty, users earn 1 point for each correct answer. An additional incentive is provided for users who answer all questions on a single form correctly and submit their answers before the halfway mark on the timer, rewarding them with 2 bonus points. This structure encourages precision and speed, enhancing the quiz's educational and competitive aspects.

**See tests: 16.**

1. The user should have a score that increases if they give a correct answer, the score should also increase if they answer before the halfway mark. – Fulfilled

The quiz differentiates between easy and hard difficulty levels by adjusting the scoring to reflect the increased challenge. For hard difficulty, the reward is 2 points per correct answer. Moreover, if a user answers all questions correctly on a single form and submits these before the halfway point on the timer, they receive 3 bonus points. The tiered scoring system rewards both accuracy and speed, especially in the quiz's harder sections, encouraging quicker responses and deeper engagement. This strategy boosts competitive spirit and cognitive skills, increasing the quiz's replay value as users aim to surpass their previous scores by improving both precision and quickness.

**See test: 16, 17.**

1. There should be an admin facility that can create, edit, and delete users and questions. Admins should also be able to add and revoke other admins. – Fulfilled

Any user who has admin privileges can create new questions and delete and edit users and questions. These features can be accessed via the main menu when the admin user logs in. By enabling an admin to edit and add questions, they can keep the quiz interesting by ensuring that players will be asked different questions, further enhancing their experience. As well as this, there are other benefits to an admin facility. For example, giving other players admin abilities. Allowing admins to create other admins helps to grow the game, as they can create new questions and this could also act as a reward for players, as it gives them more control over the game.

**See tests: 49-60.**

1. The quiz must display the player’s current score, username, avatar and high score. – Fulfilled

There is a parent from present on every screen, the parent form contains a sidebar and a header containing the player’s details, these details are displayed throughout the entire quiz. The display is updated when the player’s details are updated. This constant presence of personal data not only fosters a stronger connection between the player and the game but also serves as a continuous motivator. It pushes players to surpass their high scores, thereby deepening their engagement with the quiz. Furthermore, the dynamic update feature, which reflects changes in real-time, ensures that players are always aware of their achievements and standing. This aspect of the design not only enhances the user experience by making it more interactive and responsive but also creates a competitive atmosphere. Players are encouraged to improve and compete with their personal bests. Overall, this choice significantly contributes to a satisfying and motivating quiz journey, encouraging players to remain engaged and strive for excellence.

**See test: 25.**

1. There should be a leader board that displays the top three players. – Fulfilled

The leader board, by spotlighting only the top three players based on their total points, fosters a competitive milieu that propels users to refine their scores for the prestige of being recognized. This strategic limitation to the elite trio sharpens the competitive edge, making the quest for the top spots more intense and focused. Differentiating leader boards according to quiz difficulty further aligns with this ethos, nudging players towards perfecting their prowess at each level before progressing, thus deepening their engagement and success across the board. This layered approach not only fuels a relentless pursuit of excellence among users but also lauds superior performance, ensuring a stimulating and enriching quiz journey for participants of varying skill sets.

**See tests: 61-63.**

1. At the end of the quiz, the user should be able to replay without going back to the home page. – Fulfilled

The quiz design allows users to initiate a new quiz session immediately after completing one, without the need to navigate back to the home page. This streamlined process enhances user experience by reducing downtime between quiz attempts, encouraging continued engagement and practice. Such an approach caters to users' desire for immediate feedback and continuous learning opportunities, keeping the momentum of the quiz experience positive and engaging.

**See test: 39.**

1. There should be an unlockable mini game within the quiz. – Fulfilled

Within the quiz, an innovative feature has been introduced in the form of an unlockable mini game, designed to enrich the user experience. This mini game becomes available as users achieve 30 points. This provides a reward, for doing better. The inclusion of this feature adds an extra layer of engagement, offering a refreshing break from the standard quiz format and incentivizing users to achieve higher points for access to unique content.

**See tests: 42-48.**

## System Evaluation

### Strengths

Randomization

The first feature I’m proud of is the randomization of questions. The random number generator chooses a number starting from 1 to the number of questions in the list. It will then select a number and then load the corresponding question from the binary file. The question will be marked as used until all questions are used or if the player logs out and logs back in again.

This feature was introduced to enhance the quiz's educational value by preventing the ease of question memorization that comes with non-randomized formats. It aims to simulate a test-like environment, thus accelerating the learning process and reducing the chances of user disengagement. The inclusion of randomized questions ensures a varied and fresh experience with each playthrough, significantly reducing repetition and boredom. Furthermore, this approach extends the game's lifecycle by providing a more effective study tool, as the continual freshness of each session keeps the content relevant and engaging over time.

Admin facilities

I’m particularly proud of this key feature, the administration facilities. An administrator account will always be accessible on the quiz even if the players.bin file is deleted the administrator account will be regenerated as soon as the players bin is recreated. Logging in with the administrator account will grant access to the admin facility. The admin facility will appear as two buttons on the sidebar, a “players” button, and a “questions” button. From the player’s section of the admin facility you can view, edit, delete, create players. The questions section will allow you to view, edit, delete, or create questions that will be displayed on the quiz.

This feature serves as a significant asset by allowing clients to inject fresh questions into the quiz, thereby maintaining its freshness and allure. It fosters deeper user engagement through a stream of regularly updated challenges and shifting educational goals. Beyond keeping the quiz relevant and appealing over time, this feature simplifies the question-creation process. It assures that administering the game doesn't require coding expertise since questions are integrated without the need for hard coding. This makes the quiz more enduring and accessible, fortifying its position as a dynamic and inclusive educational tool.

Timer

The third feature I am going to mention is the timer. When the timer runs out the question is automatically marked wrong. The progression bar at the top of the screen indicates the time used creating the feel of an intense countdown.

Timed responses alongside randomized questions create exam type conditions and so aid faster learning and interest. This feature not only simulates exam-like conditions, fostering quicker learning and sustained interest, but also incentivizes swift responses by offering more points for faster answers. The combination of these elements significantly boosts user engagement, encouraging a more dynamic and interactive quiz experience that rewards knowledge and quick thinking.

Smart picture boxes

The fourth feature I am going to mention is the smart picture boxes. To ensure a consistent and visually appealing layout, I have implemented a smart positioning system for the Picture Boxes on the randomized question. The system calculates the necessary positional adjustment based on the Radio Button's text length. By measuring the text length and considering the font size, the system computes an offset value. This offset determines how far to shift the Picture Box either to the left or right. The result is that the feedback icon (tick or cross) in the Picture Box is always appropriately placed to the right of the Radio Button's text, maintaining a balanced and uncluttered interface regardless of the text length.

This ensures a flexible and responsive UI design, enhancing user experience. This well-designed, uncluttered interface minimizes cognitive load, allowing users to focus more effectively on the questions and answers. Moreover, by ensuring that feedback icons are consistently placed, the quiz reinforces user understanding and interaction with the quiz, promoting a more intuitive and frictionless engagement.

Bonus Game

The final strength I am going to mention is the mini game, the user can unlock the snake mini game once they reach 30 points. The addition of mini games is good as it gives the players a reason to replay the quiz and strive for more points and unlock games.

Incorporating a mini-game like "Snake" as a reward for reaching a certain score, such as 30 points, significantly elevates user engagement and learning through gamification, encouraging users to replay the quiz for higher scores and a chance to unlock entertaining diversions. This feature not only enhances the quiz's replay value by setting tangible goals and diversifying content but also fosters a positive feedback loop of achievement and reward, boosting user satisfaction and loyalty. Additionally, it provides a much-needed break, offering stress relief and a refreshing change of pace, making the quiz platform a more comprehensive educational and entertainment tool that appeals to a wider audience.

### Weaknesses

Randomization

One weakness is that not all question forms are randomized, and only the easy difficulty is randomized. This means that the application becomes less effective as students will eventually remember previous questions. If I had more time to work on my system, I would apply randomization to all my question forms. I would also subsequently update my admin facilities to facilitate these changes, allowing admins to edit every question type.

Easy and hard categorization

Another weakness was categorizing questions into easy and hard difficulty levels. Sorting questions into two specific levels limits my quiz and adds to its inflexibility. Instead, I should have implanted a system in which questions would be sorted by “Topics”. This would have improved the user’s learning experience as the teacher could have created specific topics for their students to play instead of all questions being shown at the same time.

Help forms.

The third weakness is my help forms. I believe I did not spend enough time on them, the instructions are not thorough enough and the images I provided were not clear enough. Having a better help form would have improved the experience of beginner users who may be confused or overwhelmed.

Brute force attacks.

A fourth weakness I identified is the susceptibility of player accounts to brute force attacks. This concern arises from the lack of measures to prevent or detect multiple incorrect password attempts. Malicious users might exploit this oversight by repeatedly guessing passwords for accounts listed on the leaderboard, posing a serious security risk due to the absence of a mechanism to thwart such unauthorized access attempts.

Mini games.

The final weakness I identified is that the application features only a single mini game, which users can unlock by achieving 30 points. This approach, while initially motivating, quickly loses its appeal as users have little reason to continue accumulating points beyond this milestone.

### Further improvements and future developments

Although trying my best to use the time provided to me to add everything, there are several features which I would have added/improved on if I was given more time.

Firstly, I would have added was a levelling/XP system that would allow users to gain XP points and level up the more they played the quiz. The user would be rewarded with a badge or trophy for their account that other users could see when they viewed the profile. This would have served to encourage users to continue replaying the quiz to level up.

Secondly, as mentioned above in my systems weaknesses I would have made the entire quiz randomized. With these changes I would subsequently have to update the administration facilities. Alongside the changes to randomization, I would have added the ability for an admin to edit the timers for each question form to either make it more challenging or easier to play through. I would have put some restrictions on this such as preventing an admin from setting the timer below 15 seconds which would make the quiz impossible to complete.

Thirdly as mentioned in my systems weaknesses I would enhance the mini game section as this section is crucial for sustaining user interest and engagement over time. By introducing a diverse array of additional mini games, each with its own unique unlocking criteria, the quiz can provide users with ongoing goals and incentives. This variety not only reinvigorates user interest but also enriches the overall experience, offering new challenges and rewards that cater to a wide range of preferences and encouraging prolonged interaction with the application.

A final development I would have added if I had more time would be social features which would allow users to add, communicate, challenge their friends. These developments would further foster a sense of community and friendly competition. This social layer could significantly increase user engagement, as players would be motivated not only by the educational content but also by the opportunity to interact and compete with peers.

## Evaluation of peer feedback

Peer feedback from a Technology and Design GCSE class, my software class peers and my software system development teacher, has provided valuable insights into the strengths and areas for improvement in my project.

My classmates and peers from the year below emphasized the importance of clarity and intuitiveness in the quiz. They appreciated the quiz's intuitiveness, indicating that they could easily navigate through it and understand its purpose. Furthermore, they expressed satisfaction with the quiz layout, finding it straightforward and user-friendly.

Feedback from the year below also praised the quiz's content, noting that it included appropriate questions aligned with the specified requirements. They found the quiz engaging and enjoyable, indicating that it effectively captured their interest and maintained their attention.

My teacher's feedback echoed the sentiments of my peers, commending the professional appearance and intuitive design of the quiz. This validation from an authoritative figure in the field further underscores the success of the project in meeting its objectives.

However, generalized feedback from all sources highlighted some areas for improvement, it became clear that the current point system was causing confusion, prompting a preference for a more straightforward scoring system. To address this, I would plan to overhaul the coding related to points, transitioning the criteria for mini-game unlocks from points-based to high-score achievements. Additionally, the feedback highlighted confusion due to the existence of separate high scores for easy and hard levels. To simplify, I would consolidate these into a single high score applicable across both difficulty settings, thereby streamlining the user experience and resolving the identified issues.

Additional feedback highlighted concerns regarding password security and the potential for unauthorized account access. To address these worries and enhance account safety, I would implement two-factor authentication (2FA) through email verification. This added security measure would provide an additional layer of protection for user accounts, significantly reducing the risk of unauthorized access and bolstering user confidence in the safety of their personal information.

Upon reflecting on the feedback received from my peers and teacher regarding my quiz application, I find it both affirming and enlightening. The recognition of the quiz's clarity and intuitiveness by my classmates resonates with me as it validates the effort, I put into meeting requirement C for an intuitive and easy-to-navigate design. Their positive comments regarding the layout and user-friendliness also reassure me that I successfully fulfilled this criterion. Furthermore, their appreciation for the engaging content aligns perfectly with my intention to meet requirement A for a fun and engaging quiz. It's gratifying to know that my project effectively captured their interest and attention. Moreover, the validation from my teacher regarding the professional appearance and intuitive design further reinforces the success of my project in meeting its objectives, particularly requirement B for a consistent colour scheme. This feedback not only highlights the project's strengths but also provides valuable insights for potential improvements, ultimately contributing to its overall success in meeting the specified requirements and objectives of the assignment.

## Self-Evaluation

Throughout my application's development, I navigated numerous challenges, fostering significant growth in several areas. My journey involved mastering object-oriented programming as I transitioned from Python to C#, improving file manipulation techniques for data storage, and honing problem-solving skills through continuous debugging. Additionally, I developed effective time management strategies to meet project deadlines. Listed below are a few of the core skills that I developed explained in more detail.

I developed my understanding of core concepts related to object-oriented programming, which was essential for the development of the program. As my knowledge and skills in this area improved, l was able to apply these concepts more effectively, enhancing the efficiency, readability, and reusability of my code. Despite this progress, I acknowledge that there were aspects of my initial coding efforts that could have been better executed, particularly in the realm of naming conventions. For instance, standardizing the naming of forms with a prefix like 'frm' could have improved code clarity. My early challenges in fully comprehending object-oriented programming impacted the quality of my initial code: However, as I continued to learn and apply these principles, t was able to overcome many of these issues, underscoring the value of ongoing learning and the application of best practices in software development.

Throughout the development of my project, good time management was key, especially because I had to balance it with other school subjects. To handle this, I set personal deadlines and gave myself enough time for each task. This approach really helped me do my best on the project given the time I had. During this time, I also faced many challenges that taught me a lot, particularly about managing my time better. I realized how important it was to organize my schedule, especially since I wanted to add more features to my program but was limited by time. By creating my own deadlines for each part of my work, including other subjects, I learned to use my time more efficiently. This helped me focus on what was most important for my program and get the most out of the time I had.

Throughout the project, I naturally enhanced my communication skills in meetings with my client, who was also my software systems development teacher. We focused on clear and concise discussions about the application's requirements, ensuring I understood and implemented what was needed. These interactions were crucial for defining user requirements and promptly resolving any issues. This process helped me build upon my existing communication skills, which were essential for the project's success and effectively managing client feedback and expectations.

Practicing good file organization and making regular backups were key strategies I adopted during the project. Saving my work on various platforms, including USBs, personal computers, OneDrive, and school computers, helped safeguard against data loss. This approach was particularly beneficial when my USB was damaged, and I potentially lost my most recent program. However, because I had backed up my code, I was able to continue programming without losing significant progress, despite occasional code corruption or forgetting to save changes. This experience underscored the importance of maintaining multiple backups to mitigate the impact of unexpected setbacks.

While developing these skills, I encountered significant challenges, particularly with the slow performance of Azure (Programming environment) and issues with OneDrive. Azure's slow operation became a noticeable bottleneck, especially as my program grew more complex. The delays in loading files sometimes extended to 3- 5 minutes for my program to run in Visual Studio, turning the testing phase into a tedious process. These performance issues forced me to complete most of the development on my personal computer, as Azure's slow speeds were nearly prohibitive for efficient development work. Furthermore, OneDrive complications made images stored for the project inaccessible, requiring manual intervention to unblock them, which added to the development hurdles.

Despite these obstacles, primarily due to Azure's performance and OneDrive's file access issues, the process of creating this application significantly boosted my understanding of object-oriented programming and honed essential skills vital for any proficient software developer.

## Testing evaluation

Throughout the testing of my educational quiz application, I adopted a manual testing methodology, selecting a wide range of test cases to scrutinize both the functionality and usability of the program. Each test was carefully documented in a testing information table, a practice I found to be quite effective for keeping track of my observations and outcomes. Despite recognizing its utility, the process was notably time-consuming and, at times, cumbersome. This experience led me to reflect on the potential benefits of integrating automated testing methods, such as unit tests, which could have streamlined some aspects of the testing process.

To organize my approach and enhance efficiency, I categorized the tests into sections based on the program’s forms and functionality, creating a testing map. This step was crucial in simplifying the search for tests related to specific functionalities. Following the completion of all planned tests, I numbered them, ensuring that every possible scenario I could think of was accounted for.

One of the primary strengths of this manual testing approach was its thoroughness in covering a broad spectrum of user interactions with the application. It was particularly adept at identifying issues related to navigation, user inputs such as empty fields or exceeding character limits, and in ensuring that error messages were both meaningful and accurately displayed. This exhaustive testing confirmed that the application behaved as expected across a wide range of scenarios, demonstrating the effectiveness of the methodology in catching potential flaws.

However, the testing process was not without its weaknesses. A notable limitation was the insufficient attention that I gave to the app's randomization features. Instead of conducting individual tests for each aspect of randomization, I opted to group them under a few broad tests. This decision, in hindsight, might have compromised the depth of testing in this area, potentially overlooking specific issues related to how questions or options are randomized within the quiz.

Despite the oversight in thoroughly testing the randomization functionality, the manual testing strategy proved successful in uncovering various bugs and ensuring the application's overall reliability. It highlighted the importance of a detailed and organized approach to testing, but also the value of considering more efficient methods, such as automated testing, for certain repetitive or straightforward tasks. The experience underscored the balance needed between comprehensive manual testing and the potential time-saving benefits of automation in future projects.

The culmination of this rigorous testing process unequivocally demonstrated the effectiveness and reliability of my educational quiz application. Through the extensive manual testing approach, encompassing a wide array of scenarios and meticulously organized into a coherent testing strategy, I was able to identify and rectify a range of issues, thereby ensuring the application performed as intended. The process not only validated the system's functionality and usability but also highlighted its resilience to various input and user interaction scenarios. This comprehensive testing regimen, despite its focus on manual techniques and the subsequent identification of areas for improvement, fundamentally proved that the system works well. It has instilled confidence in the application's ability to serve its educational purpose effectively, providing a solid foundation for future enhancements and the potential integration of automated testing methods to further bolster its reliability and user experience.