GUESS   

THAT PHRASE™

Contents

Background

I wanted to design a game that would promote inclusive play amongst all age groups and create a competitive but friendly environment so players could have fun and also make new friends. Thus, I made the decision to use preexisting emojis that are universal and don’t require translation for anyone to understand them; as well as the fact that they are used everyday by people from all over the world, including myself. The idea for the game exists on various social media platforms such as ‘Tik Tok’ and ‘Instagram,’ however it has not been made into a formal application. Currently, there is similar competition on the market with games based off popular television shows, such as, ‘Catchphrase’ and ‘Guess the word’ but there is no direct competition with the inclusion of emoji’s - giving this project a USP.

I have always enjoyed playing against myself and others in various different games, along with a desire to understand how these programs have evolved over time. I believe that one of the most interesting parts of game design is the backend code used and developed to make these as enjoyable and addictive as they are.

The first emojis were designed by Shigetaka Kurita in 1999 who worked for a Japanese cell phone company, with the literal meaning of moji being a blend of two Japanese words: picture and letter. Communication through pictures and drawings can also be dated back even further to ancient Egyptian hieroglyphs, which has allowed this ancient form of communication to be deciphered and may even be the base of modern day emojis.

It is apparent that there are a limitless number of ways to progress this project idea into a fully functioning game. The OOP principles that I will be using will most likely include multiple variations of arrays, objects and classes that will all be implemented within the windows forms app project in the Visual Studio 2022 IDE.

My target audience will primarily be children and young teens, as I believe the fun gameplay aspects of my project will be enticing to this demographic. However, it is evident that even older adults and younger children who may be unfamiliar with technology can still use and understand emojis and so the target audience can be very broad. It will also be beneficial to adults to encourage critical thinking in a form outside of the usual workplace, increasing mental health. The appeal of my project will be an illustration of its ease of use and replay ability as it is clear to me that these alongside proper error free programming are the most important factors in creating a suitable project for AS SSD.

User Requirements

The User requirements that are set by the end user of my project will deal with how a user will interact with my system and will illustrate what that user expects within my program. There are twenty user requirements that I will include that are both functional and non-functional features, these are:

Functional Requirements

UR1	The application must have a splash screen
UR2	The application must have a login and register screen
UR3	The application must allow the user to register as new user
UR4	The application must have password validation with a strength check and visual indication of strength e.g. strength bar
UR5	The application must allow the new user to choose an avatar, UserName and Password
UR6	The application must allow a user to login with a saved username and password
UR7	The application must allow an admin account to login
UR8	The application must have a main menu screen
UR9	The application must allow an admin to change account details of users
UR10	The application must allow an admin user to delete accounts
UR11	The application must have multiple levels of difficulty (Easy, Medium, Hard) for the game with appropriate information provided
UR12	The application must show if the users answer was correct or incorrect
UR13	The application must record the users score and other information in a database csv file
UR14	The application must have a Leader Board to compare users score
UR15	The application must have a variety of questions to be asked
UR16	The application must time the user until the questions are finished
UR17	The application must be optimised and run well on school grade PCs
UR18	The application must allow a user to log out

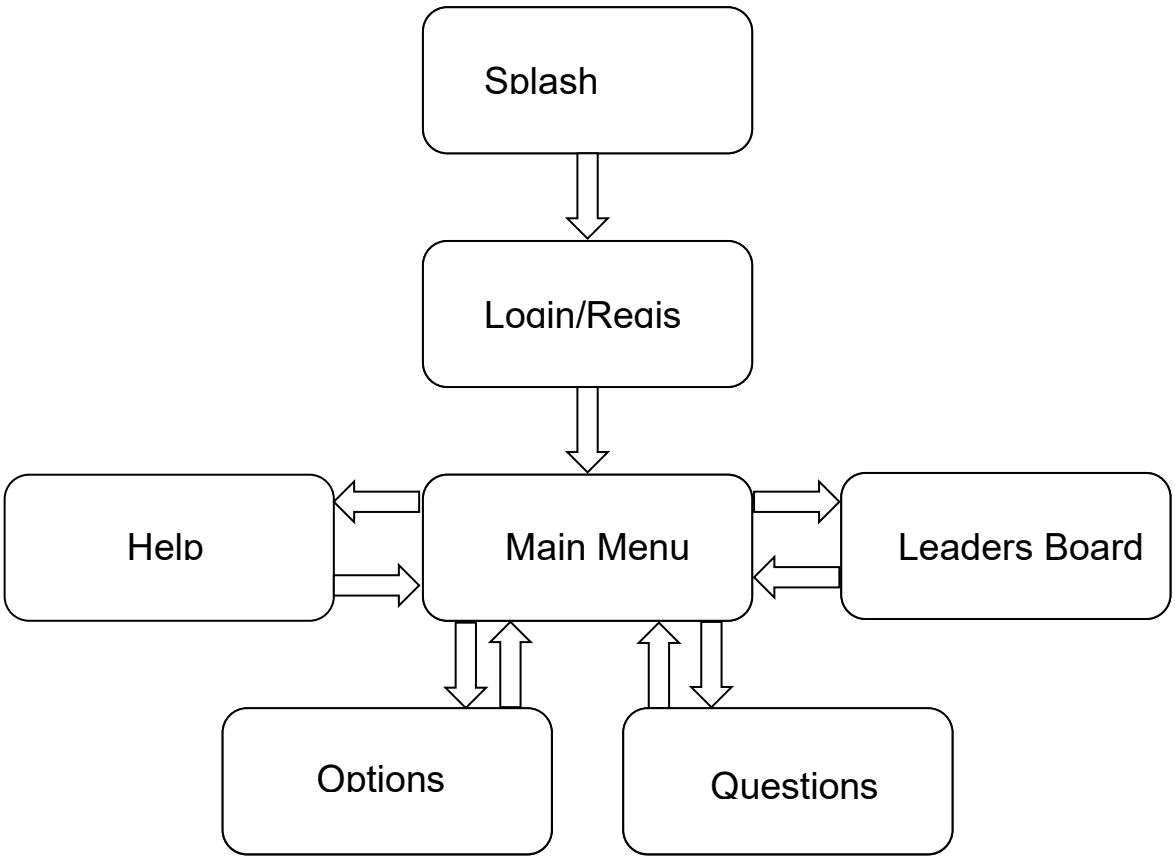
Non- Functional Requirements

UR19	The application must have appealing imagery that fits into the emoji theme throughout.
UR20	The application must have engaging headings that fit each section of the game with easily readable font.
UR21	The questions must be universal and fit the difficulty level, allowing for speed of entry to make the game addictive and encourage the user to play again.
UR22	The application must be appealing to attract more users to the game to in turn, add more users to the leader board rankings.

INITIAL DESIGNS AND PLANS

Flow Diagram

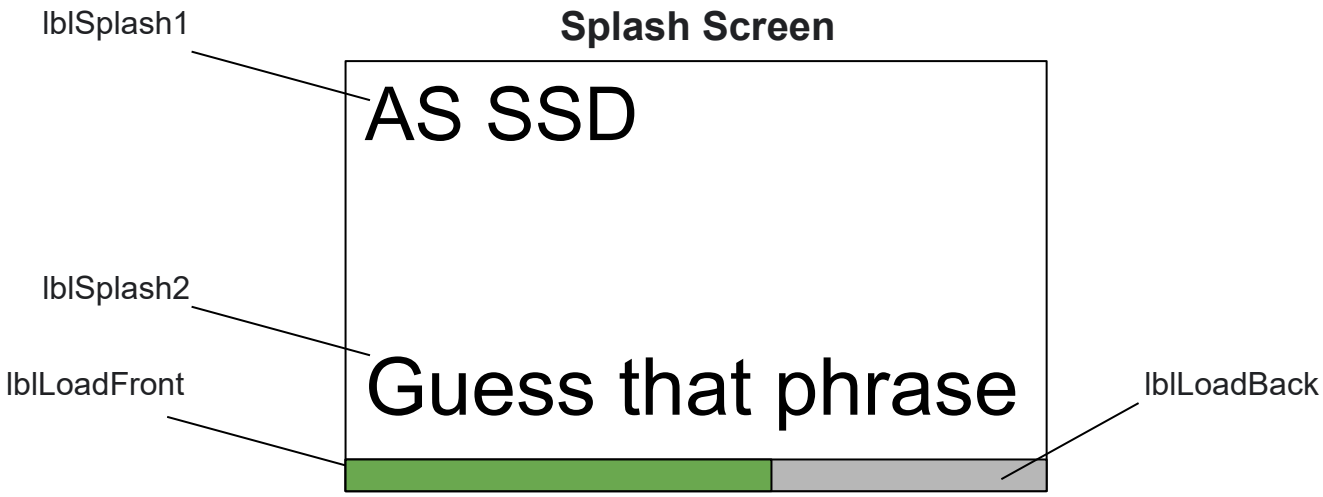
This is my application flow diagram which illustrates the navigation between forms that my application will feature, with coloured arrows representing the different pathways that the user can make use of to fully operate the program.



The application is designed to make navigation between forms as easy as possible through having the Main Menu form act as a central hub in which all other forms are reached. This helps me to meet my target audience's expectations by emphasising simplicity and keeping engagement throughout the application.

GUI Designs(Initial)

The storyboards below depict both the visual design of the application and the logical design. These are initial representations of the aesthetics of the application and will change and develop over time as the application evolves utilising peer reviews from my target audience. These designs were created using the paint technique.



USER INPUT

Remove AS SSD as it takes away from the “Game like feal”

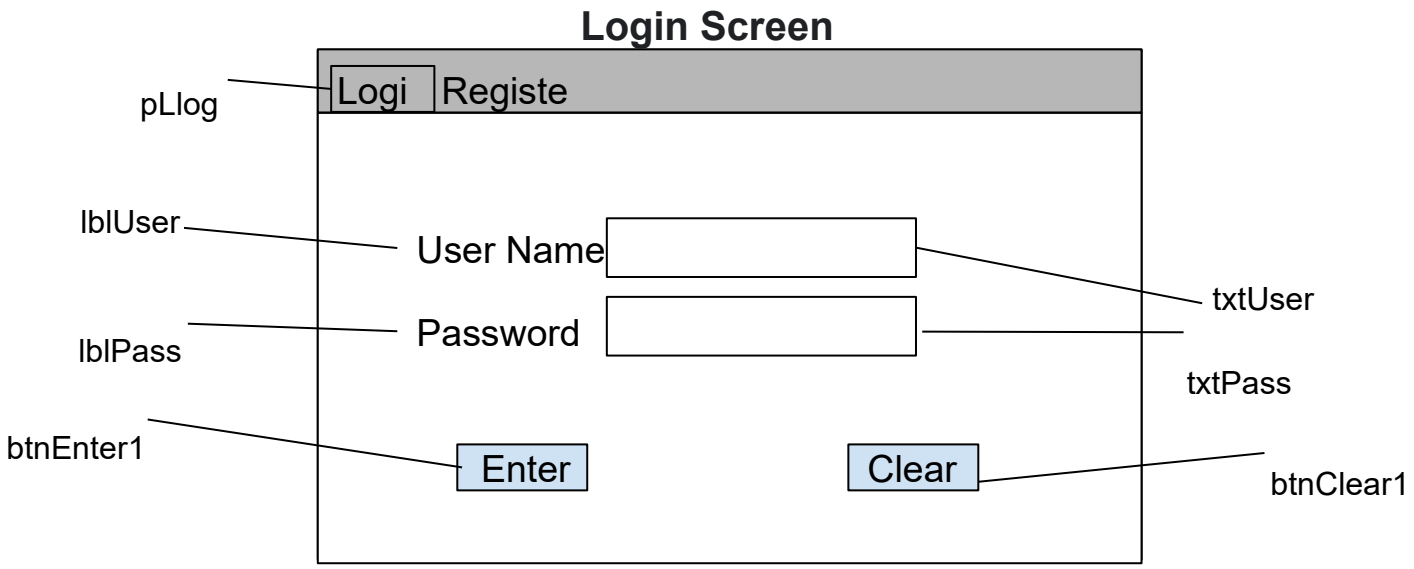
Use a more engaging font

Tie in the emoji them to this screen

Component	Function
lblSplash1	Primary title label
lblSplash2	Secondary title label
lblLoadFront	Progress bar to illustrate loading of the application
lblLoadBack	Label to contain the progress bar

. This form's function will be to display the applications initial loading sequence, It will last for 6 seconds with lblLoadFront expanding across the screen in order to add visual interest to this sequence. This screen will also be smaller than the following forms in order to add a more user friendly experience on running the application

LINK TO USER REQUIREMENT 1



USER INPUT

Change the white display to black

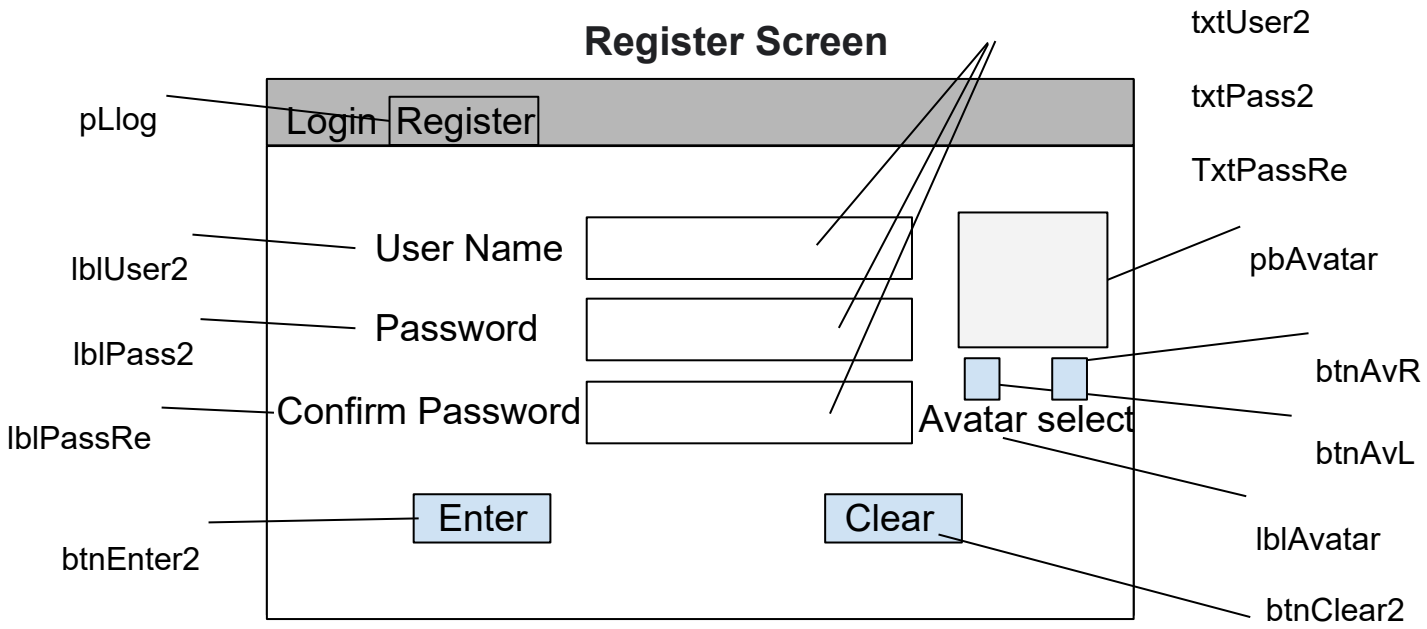
Remove panel to separate the login and register screens

Tie in the emoji theme to this screen

This form will allow the user to login to a persisting account that they have registered through the GuessThatPhrase application. The screen size for this form will be the universal size of each screen throughout the application to maintain consistency, providing a high level professionality in the whole application. This form will also have built in validation functions in order to ensure that the user has entered valid credentials to gain access to the rest of the application.

Component	Function
pLlog	This panel will be programmed to switch between the Login and Register control this will reduce applications size due to less screens
lblUser	This label contains a title for username as a prompt for the user
lblPass	This label contains a title for password as a prompt for the user
btnEnter	This label is attached to a click method that will run the login sequence
txtUser	Textbox for user to input their account information
txtPass	Textbox for user to input their account information
btnClear1	This label will be attached to a click method that will clear all fields off data for increased user experience

LINK TO USER REQUIREMENT 2



USER INPUT

Change the white display to black

Allow the user to add their email to their account

Tie in the emoji theme to this screen

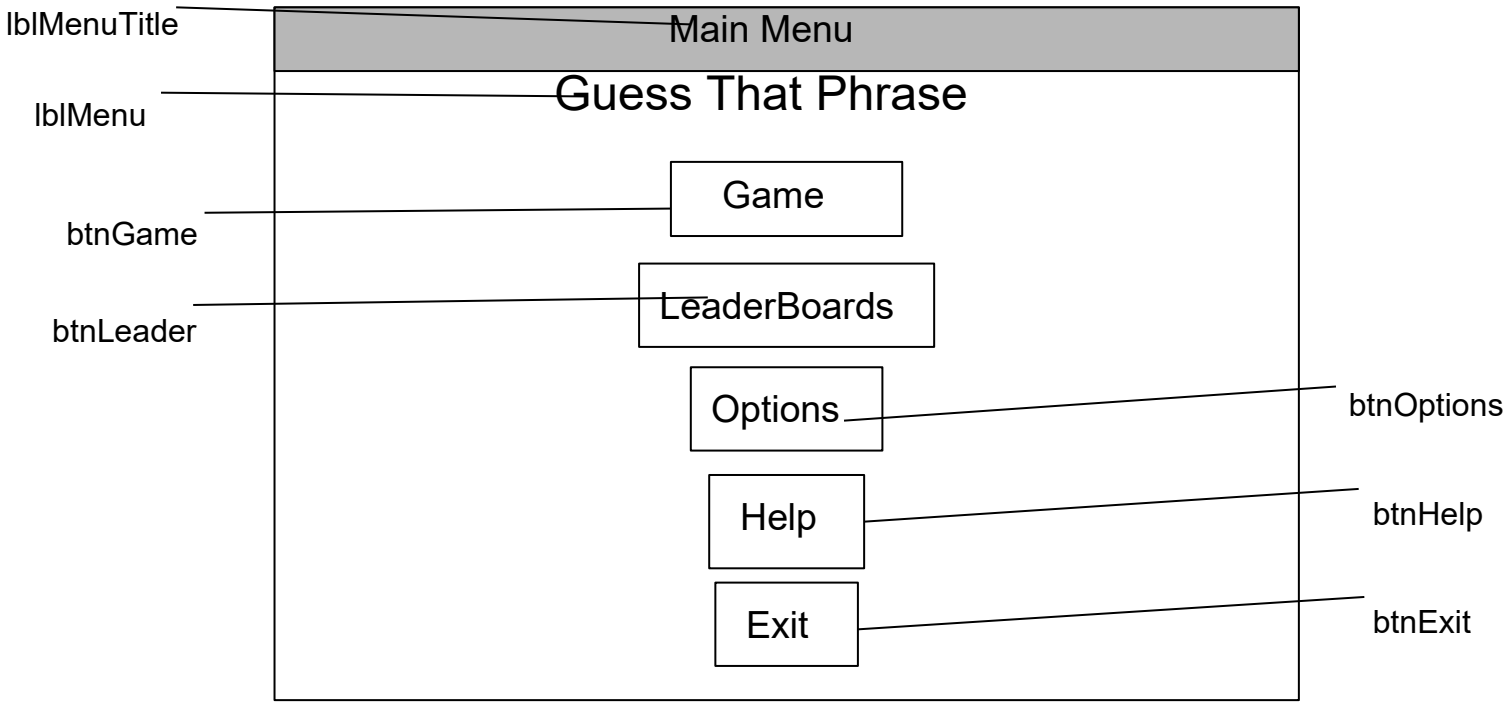
Decrease textbox and picture box size to make screen less cluttered

This screen will allow the user to create an account for the GuessThatPhrase application. It will contain a confirm password field to prevent the user from accidentally misspelling their password. It will also contain many validation checks to ensure valid credentials have been inputted such as: a high level of password strength, a username that is an appropriate character length and a username that is unique which will prevent miss identification on concurrent screens.

Component	Function
pLlog	This panel will be programmed to switch between the Login and Register control this will reduce applications size due to less screens
lblUser2	This label contains a title for username as prompts for the user
lblPass2	This label contains a title for password as prompts for the user
lblPassRe	This label contains a title for confirm password as a prompt for the user
btnEnter2	This label is attached to a click method that will run the login sequence
txtUser2	Textbox for user to input their account information
txtPass2	Textbox for user to input their account information
TxtPassRe	Textbox for user to input their account information
btnClear2	This label will be attached to a click method that will clear all fields off data for increased user experience
pbAvatar	This picture box will display the users avatar choices
btnAvr	This button will allow the user to cycle their avatar choices right
btnAvL	This button will allow the user to cycle their avatar choices left

LINK TO USER REQUIREMENT 2

Menu Screen



USER INPUT

Add a bonus game section

Change the white display to black

Remove grey banner

Tie in the emoji theme to this screen

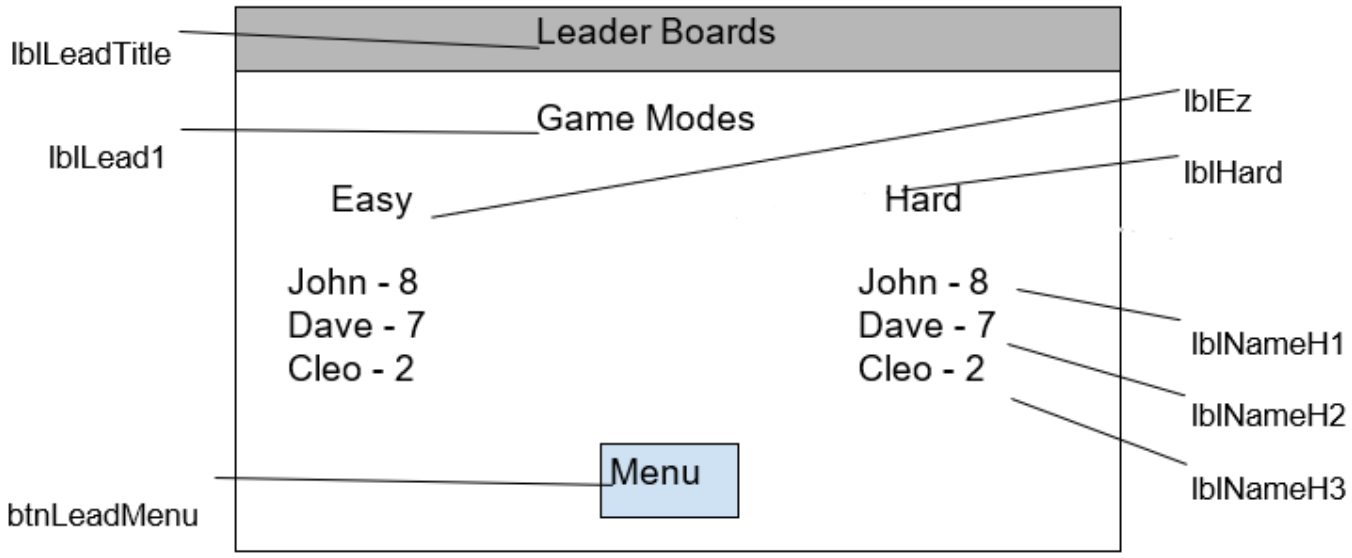
Increase main menu text size

The Main Menu is the crossroads in which all roads of the application meet. From this screen the user can access the dedicated Game forms, the dedicated Leader Board form, the Options and help labels and exit the application. Behind the scenes the current user object is being passed between forms allowing for efficient information transfer.

Component	Function
lblMenuTitle	This is the title label for the main menu from which will allow easy identification of the form that the user is currently on
lblMenu	This label contains the applications name allowing further promotion of the GuessThatPhrase brand
btnGame	This label will be tied to a click method that will set the game selection label visible which from that the user can select their desired game difficulty
btnLeader	This label will be tied to a lick event that will take the user to the Leader Board screen
btnOptions	This label is tied to click event that will take the user to the Options screen
btnHelp	This label is tied to click event that will take the user to the Help screen
btnExit	This label will be tied to a click method that will log the user out and bring them to the login screen

LINK TO USER REQUIREMENT 9

Leader Board Screen



USER INPUT

Change the white display to black

Remove grey banner

Tie in the emoji theme to this screen

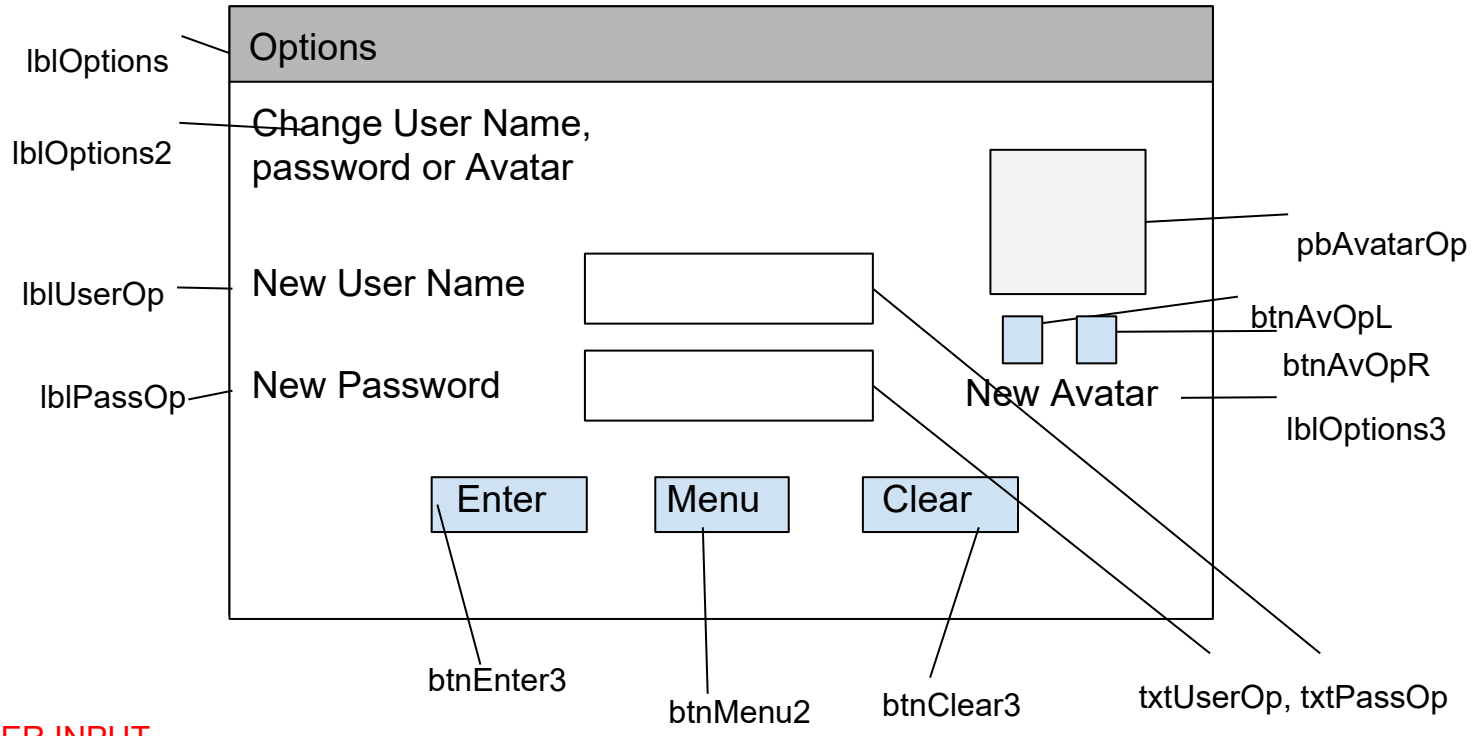
Simplify the screen by only taking scores from the hard mode for increased competitiveness

The Leader Board form will display users performances within the GuessThatPhrase application. The users will be ranked in descending order with the highest scoring users placed at the top. This will add competitive flair to the application and allow for more replay ability and increase fun as users can compete against their friends, to see who is the best at Guessing That Phrase.

Component	Function
lblLeadTitle	This is the title label for the main menu from which will allow easy identification of the form that the user is currently on
lblLead1	This is a secondary title label for the Leader Board screen
btnLeadMenu	This label will be tied to a lick event that will take the user to the Main Menu screen
lblNameH1 lblNameH2 lblNameH3 lblNameE1 lblNameE2 lblNameE3	These labels will display the current top 3 scoring users and there scores in the GuessThatPhrase application. They will be ranked in descending order with the highest scores at the top. Scores will be taken form the easy and hard sections of the game.
lblEz	This is a sub heading label to display which level the scores below have been taken from
lblHard	This is a sub heading label to display which level the scores below have been taken from

LINK TO USER REQUIREMENTS 14 & 15

Options Screen



USER INPUT

Change the white display to black

Remove the grey banner

Fore security make only the admin able to change the users information(The user should be notified of any changes)

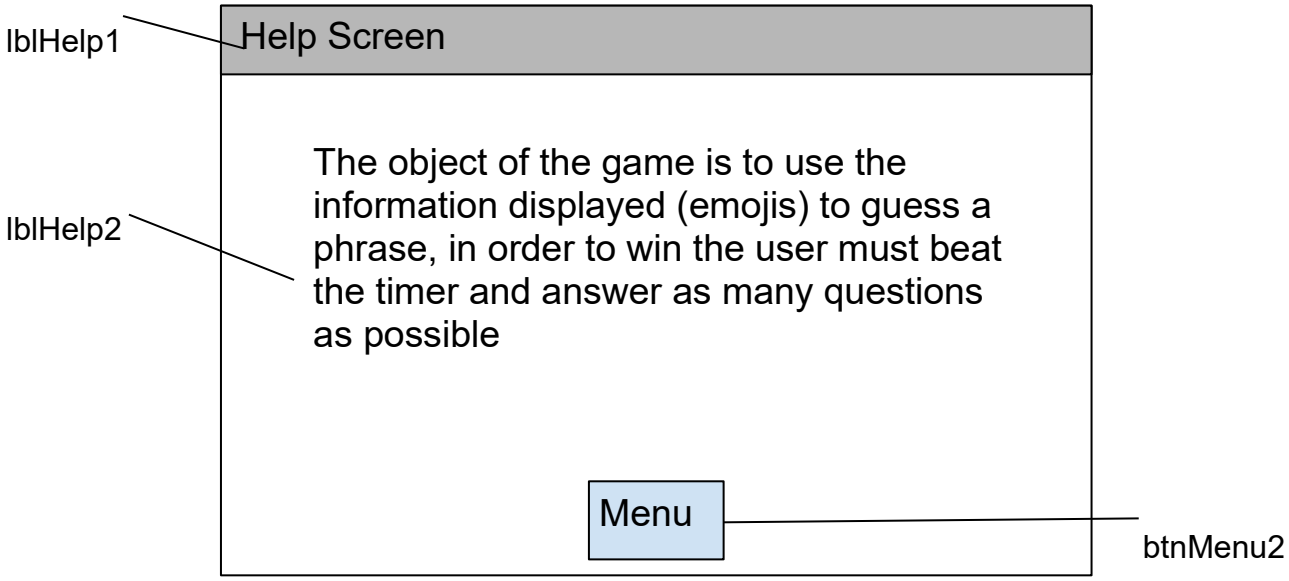
Decrease textbox and picture box size to make screen less cluttered

The options screen will allow the current user to alter their account information including their: Username, Password and avatar. This will allow the user to resecure their account in the case of a data leak. It will also add to the user experience as it will give them more freedom to customise their own account. Behind the scenes there will be validation to ensure that the user has entered correct information.

Component	Function
lblOptions	Primary title label
lblOptions2	Instructions for the user to edit their information
lblUserOP	Prompt for the user to enter their new username
lblPassOP	Prompt for the user to enter their new password
pbAvatarOP	Avatar picture box that will contain the same images on the Register screen to ensure continuity
btnAvOpL	Button to cycle avatar picture box left
btnAvOpR	Button to cycle avatar picture box right
lblOptions3	Prompt for the user to select a new avatar
txtUserOp	Textbox for user to input their new account information
txtPassOP	Textbox for user to input their new account information
btnEnter3	Button for the user to submit the changes to their account
btnMenu2	Button to take user to the Main Menu
btnClear3	Button to clear all fields within this screen

LINK TO USER REQUIREMENTS 5 & 10

Help Screen



USER INPUT

Change the white display to black

Remove the grey banner

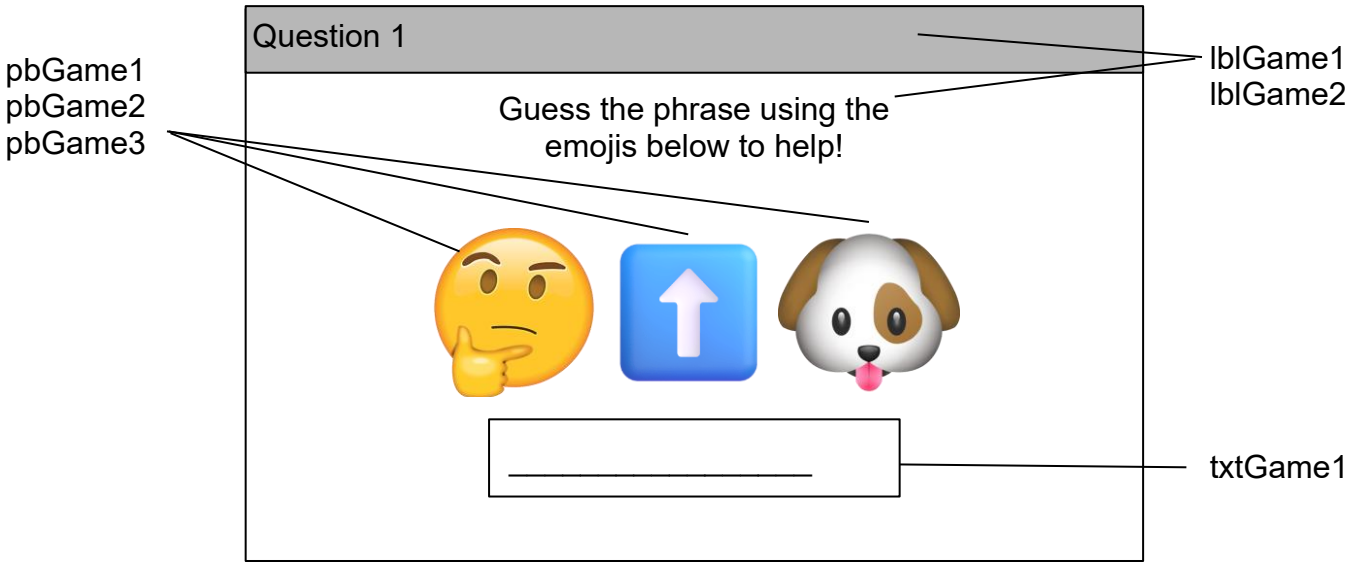
Put this in the Main Menu screen

Component	Function
lblHelp1	Title label
lblHelp2	Information label about how to play and enjoy the application
btnMenu2	Button to return the user to the main menu screen

The help form is needed if the user does not understand how to play the game. It opens from the main menu and uses a label to display the information required to play and enjoy the game.. A separate help form has been chosen to allow the user to spend as much time reading about how to play the game as they need, as opposed to 'pop-up' on the game form, which also prevents the cluttering of the game form.

LINK TO USER REQUIREMENT 12

Game Easy Screen



USER INPUT

Change the white display to black

Remove the grey banner

Add a visual element to the timer function

Ensure variation between questions

Put the question phrase in a single label

Display the users score on this screen

Component	Function
lblGame1	Primary title label
lblGame2	Secondary title with added information
pbGame1 pbGame2 pbGame3	These picture boxes will hold the questions for the game forma and will be updated according to the phrase asked
txtGame1	This is where the user will enter their answer

The game form Is the applications primary form from that the user will spend most of their time on. The questions are designed to be interesting and topical, they will be randomised with the possibility of repeating to build up memory and typing skill. This ensures a high level of competitiveness is kept throughout the application preventing boredom from monotonous questions. Behind the scenes a timer object will be created linked to question methods that will robustly ask and receive answers to ensure continued play.

LINK TO USER REQUIREMENT 13 & 14

GUI Designs (Final)

The storyboards below depict the final visual and logical design of the application. These designs are the resulting compilation of self and peer review and will serve as an imported image within the form control acting as a base for textboxes and labels to be built upon. The designs were designed in Pixlr.com and by using the paint method, designs in windows forms may differ slightly.

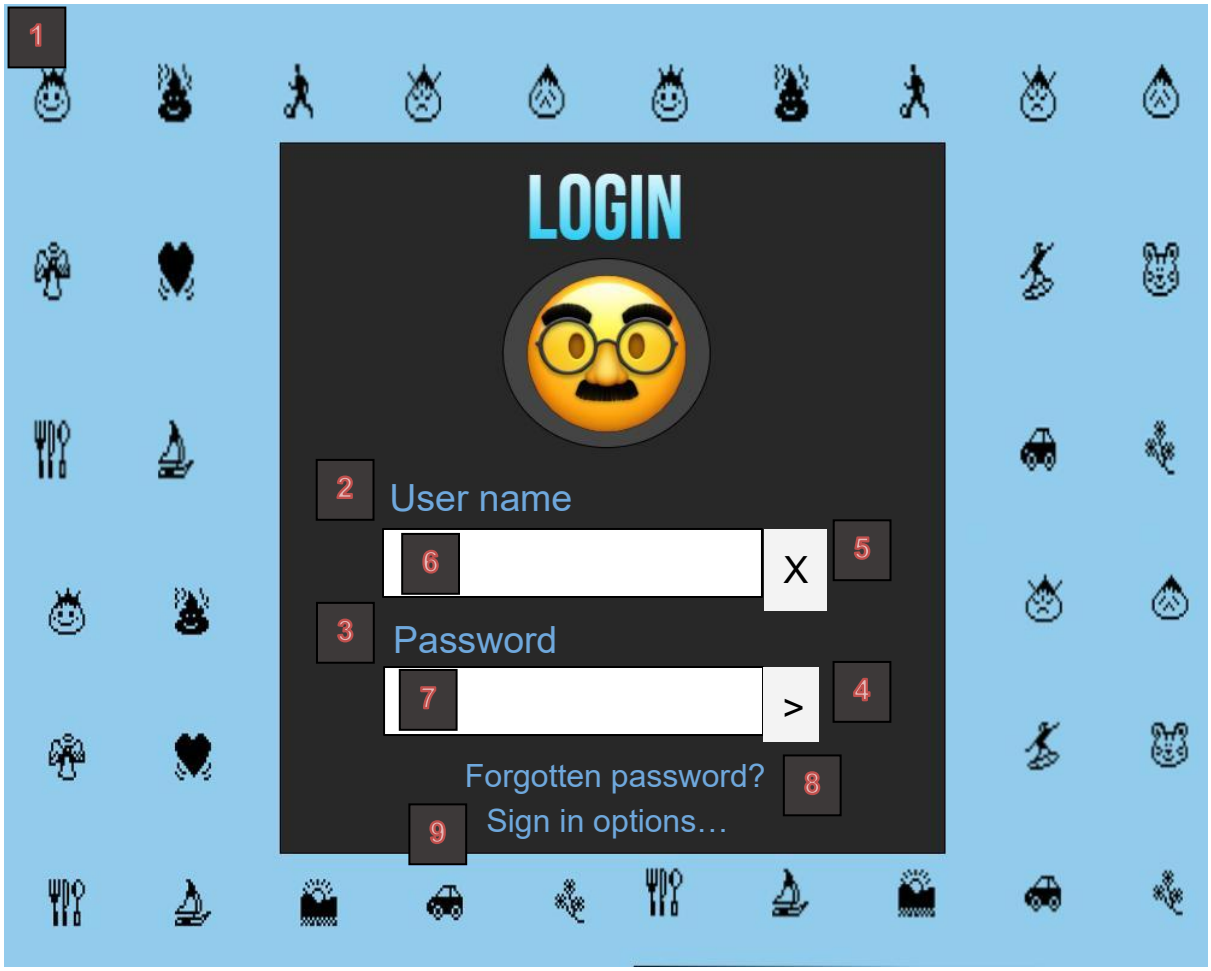
Splash Screen

This is the first form that is displayed to the user when the application is run. It will be visible for 5 seconds and then close after a timer function reaches its predetermined value. It has a loading bar for visual representation of the program functioning which is represented by a gif imported into a Label(White box).

Component	Properties
1.Form1-SplashScreen	Size: 700, 600 FormBorderStyle: None Back Colour:Null Text: Null Image:CustomPaintSS.jpeg

2.lblLoading	Size: 250, 90 Image:CustomGifss.gif

Login Form

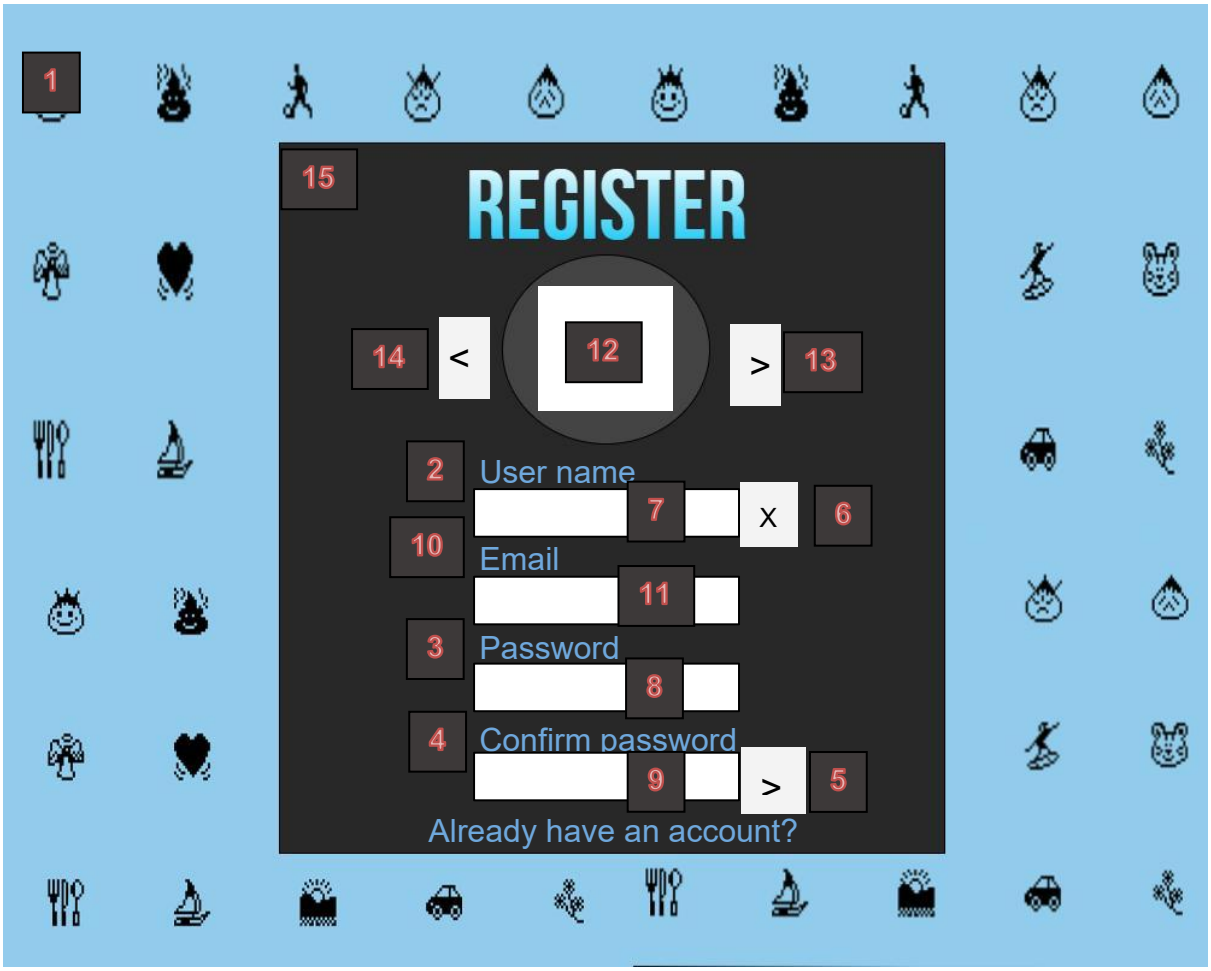


This is the second form that the program contains and is the pathway to all other forms within the application. The btnEnter button will trigger a search CSV file method that will ensure the users username and password match, if it returns true it will take you to the game, logged in to the account with the username in the txtUser text box, as long as the password in the txtPass text box matches that held by the account.

Component	Properties
1.Form2-Welcome	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Text: Null Image:CustomW.jpeg
2.uNameLoginlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “User Name”

3.pWordLoginlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “Password”
4.submitLoginlbl	Size: 100, 30 Font Size: 14pt Font: Microsoft Sans Serif Back Colour(RGB): 0, 0, 250 ForeColour(RGB): 2, 48, 32 Text: “Enter” BorderStyle Fixed3D
5.CearLoginlbl	Size: 100, 30 Font Size: 14pt Font: Microsoft Sans Serif Back Colour(RGB): 0, 0, 250 ForeColour(RGB): 2, 48, 32 Text: “Clear” BorderStyle Fixed3D
6.uNameLoginTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): Control
7.pWordLoginTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): Control
8.loginForgotPasslbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent ForeColour(RGB): 2, 48, 32 Text: “Forgot password?”
9.lgoinOptionslbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent ForeColour(RGB): 2, 48, 32 Text: “Sign in options…”

Register Form



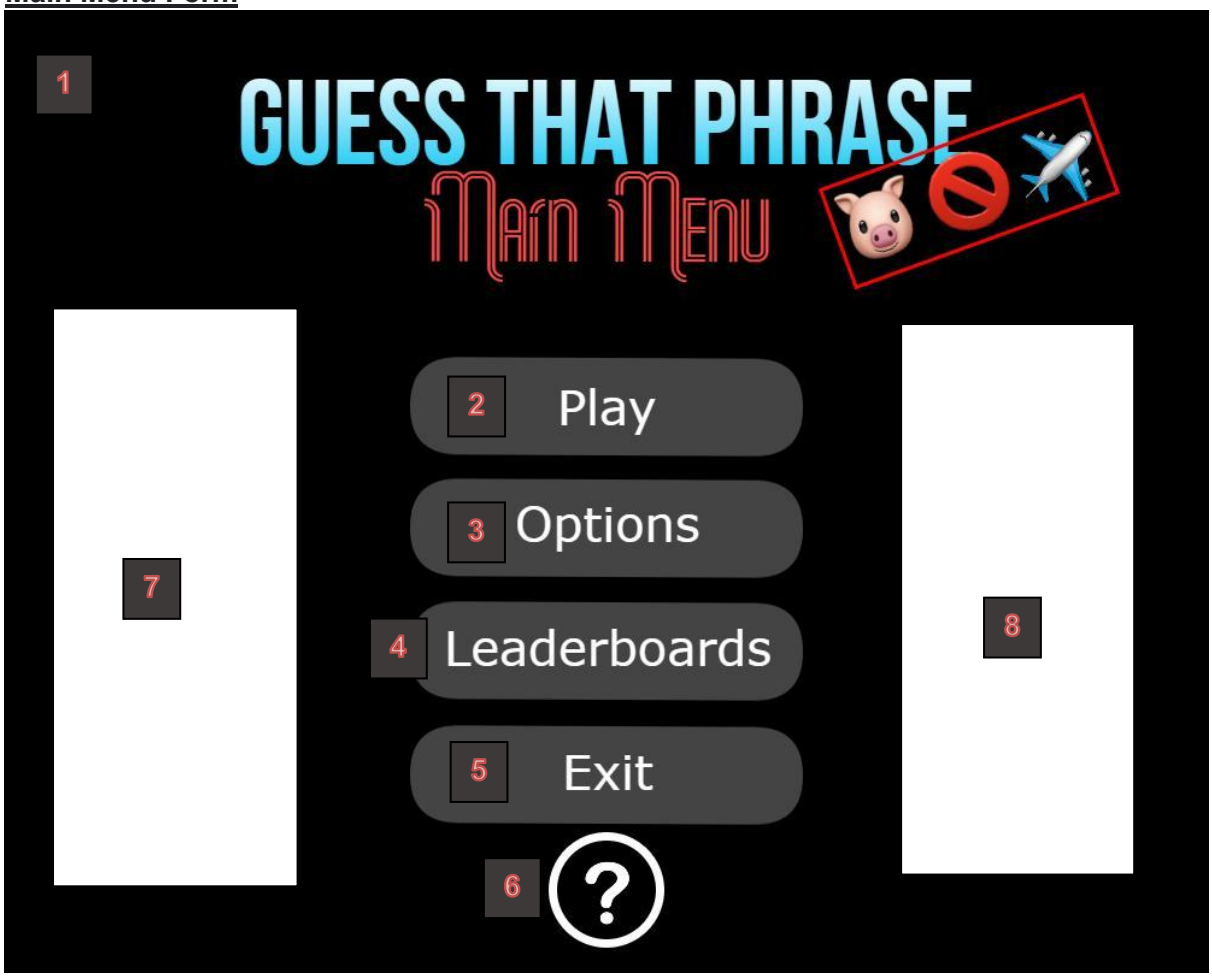
This is the third form that the program contains and is the pathway to the Login form once the user has set their user details. The user's name, email and password must comply with a set of parameters such as a minimum username length that includes multiple numeric characters and an email that contains an @ symbol.The confirm password catches typos by prompting users to type their password twice.

Component	Properties
1.Form3-Register	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Text: Null
2.uNameRegisterlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “User Name”

3.pWordRegisterlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “Password”
4.pWordRegisterConfirmlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “Confirm Password”
5.submitRegisterlbl	Size: 30, 30 Font Size: 14pt Font: Microsoft Sans Serif Back Colour(RGB): 0, 0, 250 ForeColour(RGB): 2, 48, 32 Text: “>” BorderStyle Fixed3D
6.clearRegisterlbl	Size: 30, 30 Font Size: 14pt Font: Microsoft Sans Serif Back Colour(RGB): 0, 0, 250 ForeColour(RGB): 2, 48, 32 Text: “X” BorderStyle Fixed3D
7.uNameRegisterTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
8.pWordRegisterTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
9.pWordRegisterConfirmTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
10.emailRegisterlbl	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif

	Back Colour(RGB): 255, 255, 255 ForeColour(RGB): 2, 48, 32 Text: “Email”
11.emailRegisterTB	Size: 150, 30 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
12.avatarReglbl1 avatarReglbl2 avatarReglbl3	Size: 100, 100 Back Colour(RGB): Null Images: CustomAvMale.jpeg, CustomAvFemale.jpeg, CustomAVOther.jpeg
13.LBLScrollRight	Size: 50, 60 Back Colour(RGB): 0, 0, 250 Text: “>”
14.LBLScrollLeft	Size: 50, 60 Back Colour(RGB): 0, 0, 250 Text: “<”
15.QuestionMarkRegisterlbl	Size: 60, 60 Image: CustomQMark.jpeg Back Colour(RGB): Null
registerHelpScreenlbl Appears after control 15 Click()	Size: 270, 350 Image: CustomHelp.jpeg Text:Resides in image Back Colour(RGB): Null

Main Menu Form



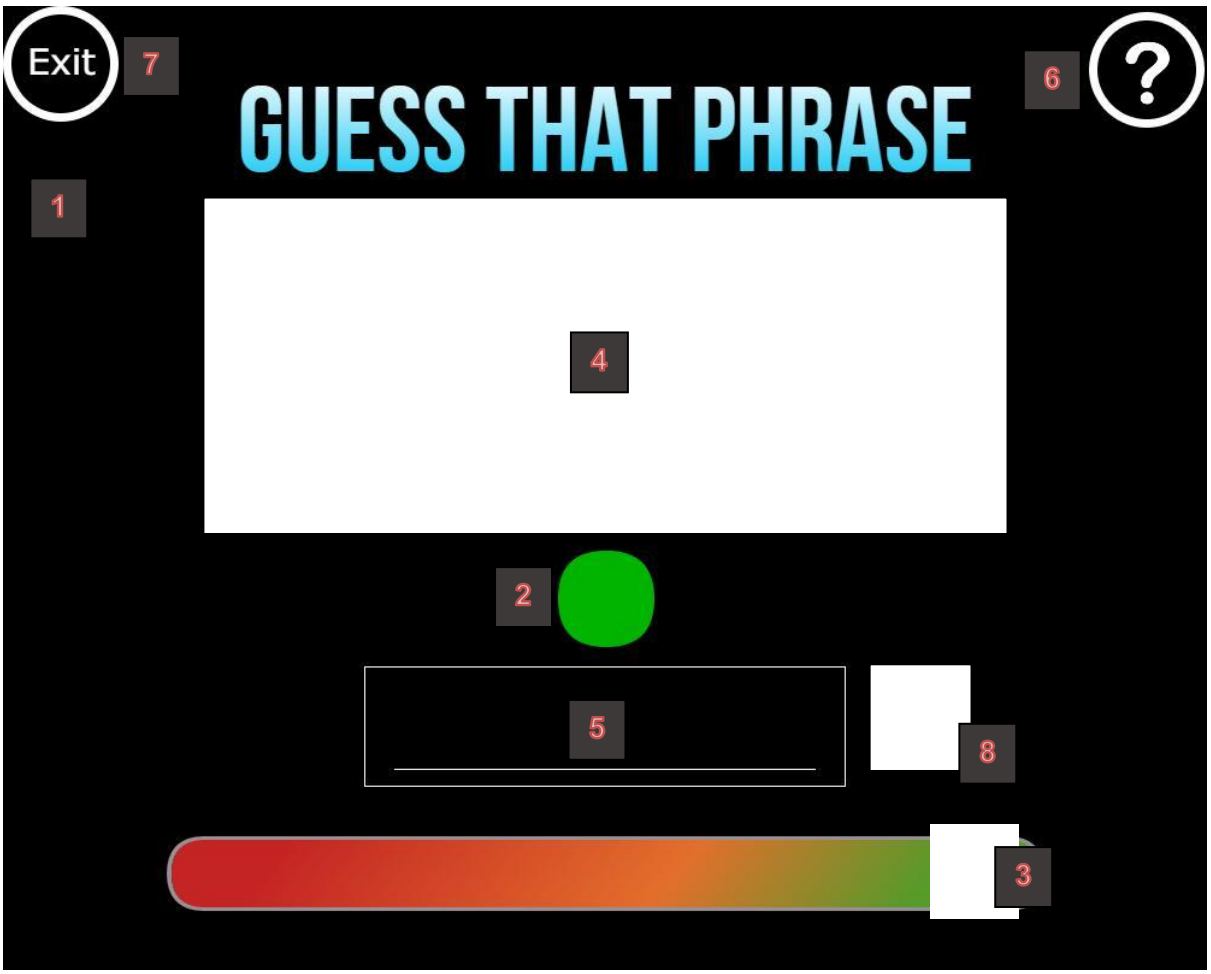
This is the Main Menu form; it holds all the buttons that take the user to the respective forms; this is handled by hiding the main menu form and showing whatever form the user selects to open. Each form in the application apart from the login and registration screen take the user back to the Menu as it serves as the hub for the application. The exit button exists as a way to stop the application from running without having to use the IDE’s built in stop program function. Text and images of main menus button exist within the image with transparent labels over them to act as buttons.

Component	Properties
1.Form3-Main Menu	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Image: CustomMainMenu.jpeg
2.MainMenuPlaylbl	Size: 330, 80 Font Size: 20pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent Text: “”
3.MainMenuOptionslbl	Size: 330, 80

	Font Size: 20pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent Text: “”
4.MainMenuLeaderBoardlbl	Size: 330, 80 Font Size: 20pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent Text: “”
5.MainMenuExitlbl	Size: 330, 80 Font Size: 20pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent Text: “”
6.MainMenuHelplbl	Size: 100, 100 Font Size: 20pt Font: Microsoft Sans Serif Back Colour(RGB): Transparent Text: “”
7.gif1	Picture Box Size:500, 470 approx. Image: CustomGif1
8.gif2	Picture Box Size:300, 200 approx. Image: CustomGif2
Optionslbl Appears after control 3 Click()	Size:400, 350 Back Colour :Dim Gray Text : ‘OPTIONS Please toggle if you would like the advanced design option for a more enjoyable user experience. Leaving this untoggled will show basic design options throughout the game (higher performance).’
optionsCheckBox Appears after control 3 Click()	Size:110, 20 Back Colour :Dim Gray Text : ‘’Advanced design’’
optionsCloselbl Appears after control 3 Click()	Size:20, 20 Back Colour :Silver
MainMenuGameInfoCloseLBL Appears after control 2 Click()	Size:20, 20 Back Colour :Dim Gray

MainMenuGameInfoLBL Appears after control 2 Click() or Appears after control 6 Click()	Size:300, 200 Back colour : Null Image : CustomGameinfo.jpeg
BonusGameslbl Appears after control 2 Click()	Size:30, 30 Text: “Bonus” Image: CustomBonus.jpeg
MainMenuGameInfoEasyLBL Appears after control 2 Click()	Size: 90, 30 Text: “Easy Mode” Colour: Olive green
MainMenuGameInfoHardLBL Appears after control 2 Click()	Size: 90, 30 Text: “Hard Mode” Back colour: Coral

Game Form



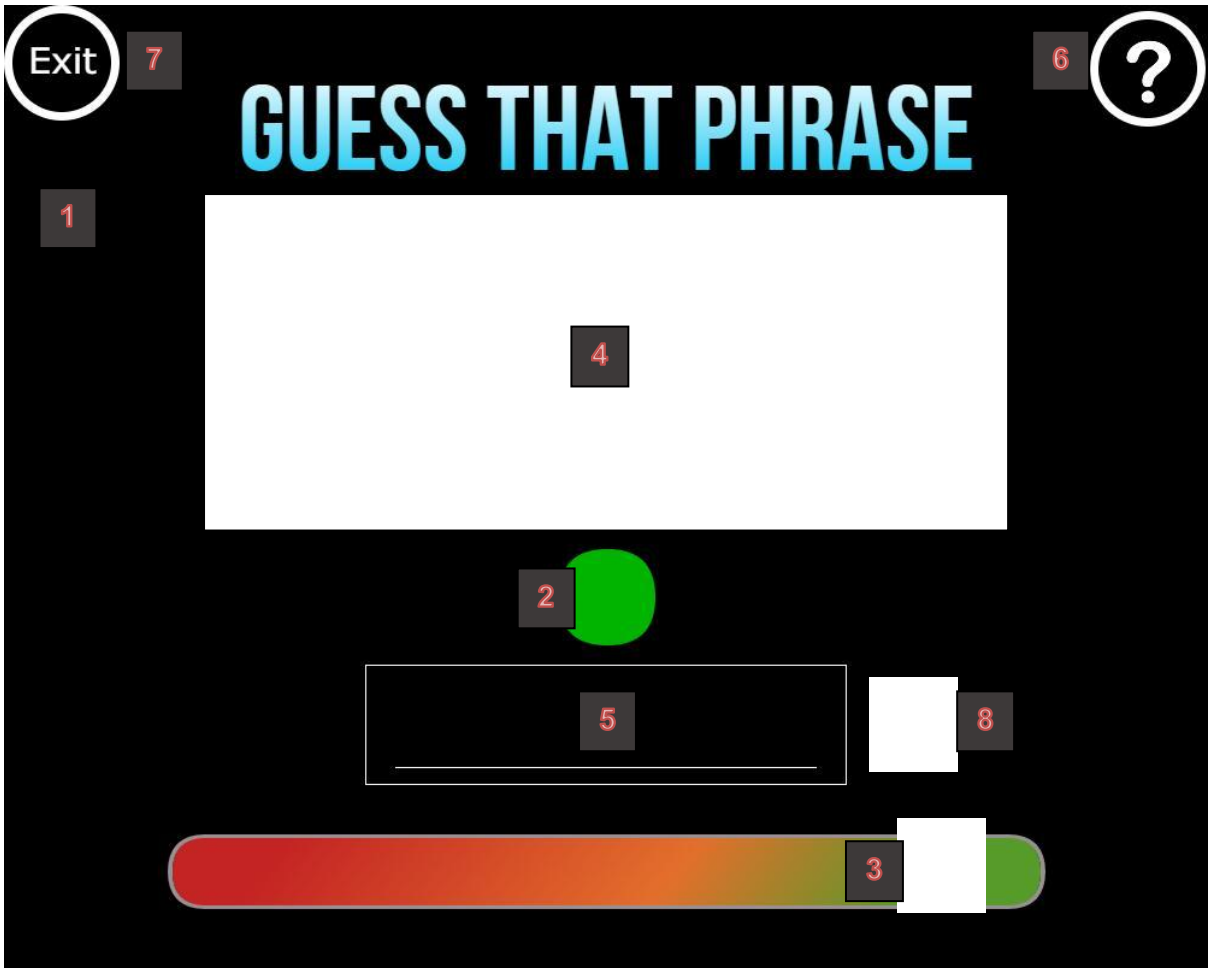
The game form is the primary form in the application; it will consist of multiple labels placed over each other that will hide and show to reflect the question being asked. The questions will be chosen at random by using a dictionary at the point of a random int. This is an example of what a question could be,with this question relating to the answer “What's up dog?”. The user will enter the answer into the textbox, however there will be code implemented to allow for the user to enter different variations of the “What's up dog phrase” such as capitalization on a certain letter or adding commas or apostrophes such as “What’s” ensure the application is user friendly.

Component	Properties
1.Form4 – Game	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Image: CustomGame.jpeg Text: Null
GameEasyGameOverlbl Appears on end sequence	Size: 880, 500 Colour: Gray Text: “Final Score: Game Over”

GameEasyReadylbl Appears on start sequence	Size: 880, 500 Colour: Gray Text: ‘’ Get Ready To Play!’’
GameEasyGameOverMainMenulbl Appears on end sequence	Size: 230, 80 Text: “Main Menu”
GameEasyGameOverPlayAgainlbl Appears on end sequence	Size: 230, 80 Text: “Play Again”
GameEasyGameOverScorelbl Appears on end sequence	Size: 100, 100 Colour: Yellow Green
2.GameEasyScoreDisplayTB	Size: 150, 60 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 Fore Colour(RGB): 2, 48, 32 Text: “The object of the game is to use the information displayed (emojis) to guess a phrase, in order to win the user must beat the timer and answer as many questions as possible”
GameInfolbl Appears on control 6 Click()	Size: 300, 260 Back Colour(RGB): Control Image: CustomInfo.jpeg
3.GameEasyClocklbl	Size: 80, 80 Back Colour(RGB): Null Image: CustomClock.jpeg
4.GameEasyQlbl1-50	Size: 850, 275 Back Colour(RGB): Control Image: CustomGameEasyPhrase1-50.jpeg
5.EasyPhraseAnsTB	Size: 300, 60 Font Size: 32pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
6.GameHelplbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
7.GameExitlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
8.SubmitGameEasylbl	Size: 100, 100 Back Colour(RGB): Transparent

	Text: “” Image: CustomGameEasySubmit.jpeg
SubmitRightGameEasylbl Appears on control 8 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image: CustomGameEasySubmitRight.jpeg
SubmitWrongGameEasylbl Appears on control 8 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image:CustomGameEasySubmitWrong.jpeg
GameEasyTrafficLightlbl1-4 Appears on start sequence	Size: 255, 300 Back Colour(RGB): Transparent Text: “” Image: CustomGameEasyTLight1-4.jpeg

Game Hard Form



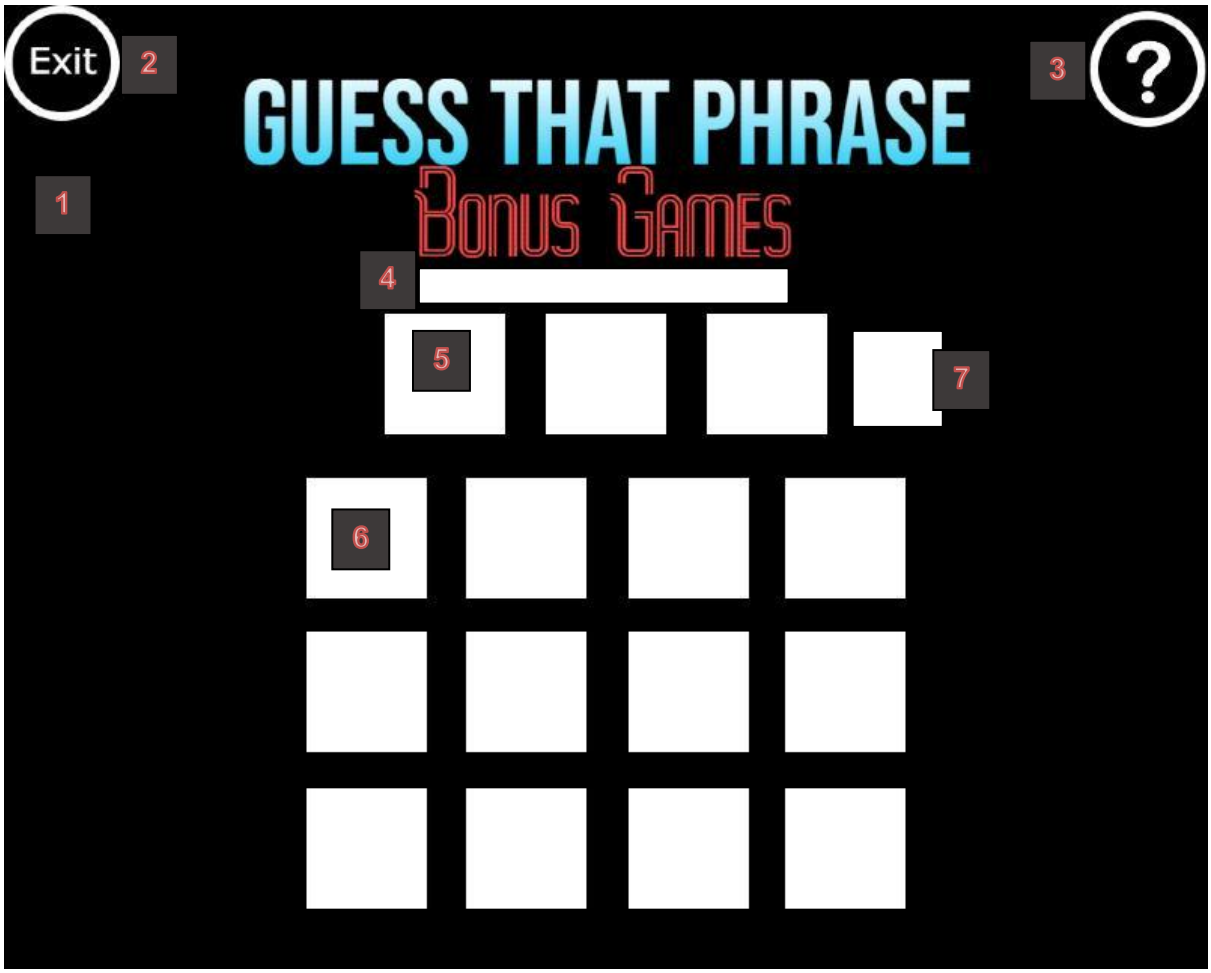
This form is similar to that of the Game Easy Form however the question set is more challenging and the timer increases by a 5 second interval every time a user answers a question correctly.

Component	Properties
1.Form5 – GameHard	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Image: CustomGameHard.jpeg Text: Null
GameHardGameOverlbl Appears on end sequence	Size: 880, 500 Colour: Gray Text: “Final Score: Game Over”
GameHardReadylbl Appears on start sequence	Size: 880, 500 Colour: Gray Text: ‘’ Get Ready To Play!’’

GameHardGameOverMainMenulbl Appears on end sequence	Size: 230, 80 Text: “Main Menu”
GameHardGameOverPlayAgainlbl Appears on end sequence	Size: 230, 80 Text: “Play Again”
GameHardGameOverScorelbl Appears on end sequence	Size: 100, 100 Colour: Yellow Green
2.GameHardScoreDisplayTB	Size: 150, 60 Font Size: 16pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255 Fore Colour(RGB): 2, 48, 32 Text: “The object of the game is to use the information displayed (emojis) to guess a phrase, in order to win the user must beat the timer and answer as many questions as possible”
GameHardinfolbl Appears on control 6 Click()	Size: 300, 260 Back Colour(RGB): Control Image: CustomInfo.jpeg
3.GameHardClocklbl	Size: 80, 80 Back Colour(RGB): Null Image: CustomClockGH.jpeg
4.GameHardQlbl1-50	Size: 850, 275 Back Colour(RGB): Control Image: CustomGameHardPhrase1-50.jpeg
5.HardPhraseAnsTB	Size: 300, 60 Font Size: 32pt Font: Microsoft Sans Serif Back Colour(RGB): 255, 255, 255
6.GameHardHelplbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
7.GameHardExitlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
8.SubmitGameHardlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image: CustomGameHardSubmit.jpeg
SubmitRightGameHardlbl	Size: 100, 100

Appears on control 8 Click()	Back Colour(RGB): Transparent Text: “” Image: CustomGameHardSubmitRight.jpeg
SubmitWrongGameHardlbl Appears on control 8 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image: CustomGameHardSubmitWrong.jpeg
GameHardTrafficLightlbl1-4 Appears on start sequence	Size: 255, 300 Back Colour(RGB): Transparent Text: “” Image: CustomGameHardTLight1-4.jpeg

Bonus Game

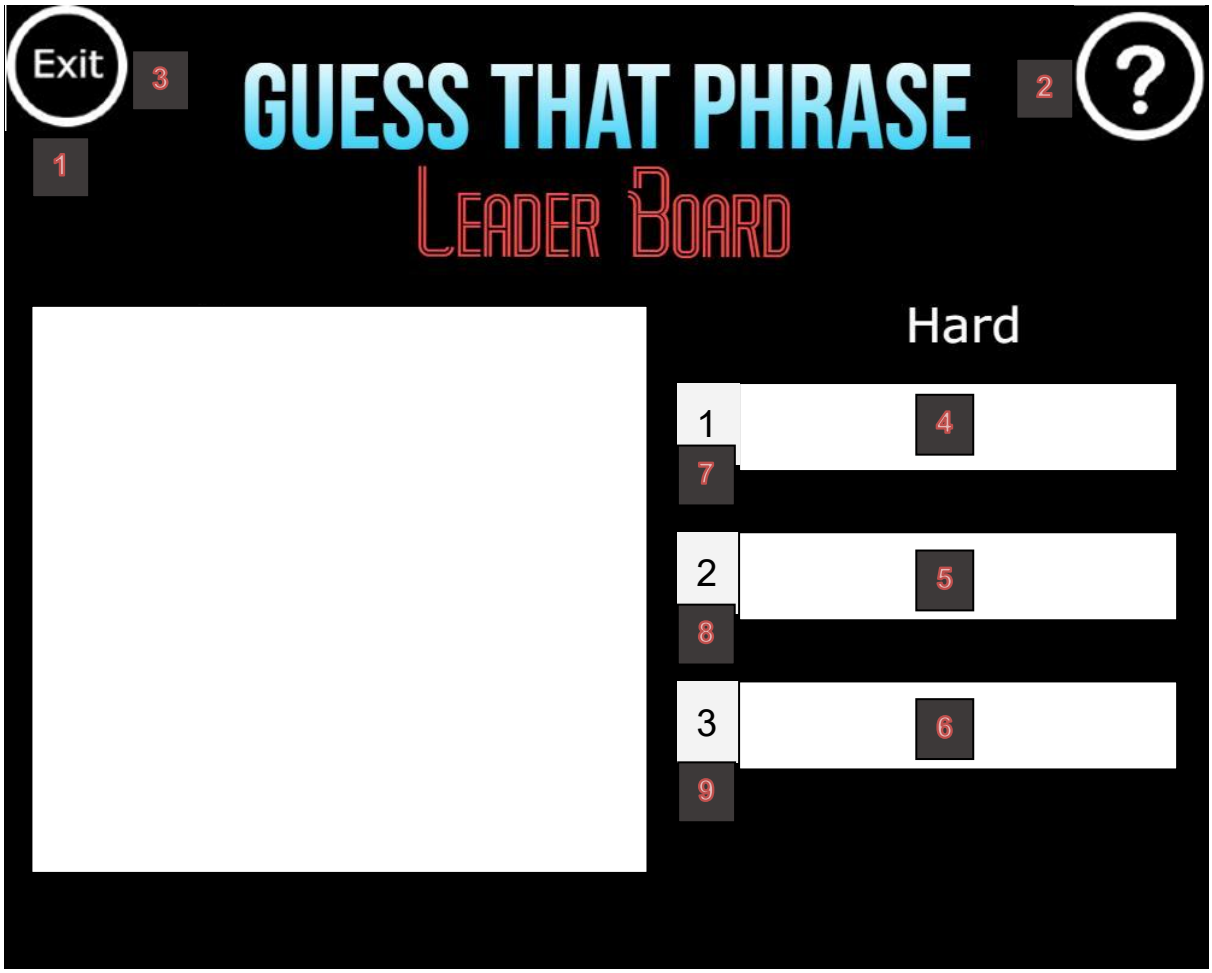


The Bonus Game form is that it will consist of 12 labels that the user can Drag and Drop, as well as 3 picture boxes that the labels image can be dragged into. The questions will be chosen at random by using a dictionary at the point of a random int, with the Spanish phrase and the English translation. The user will then Drag their answer into the perspective picture boxes in the correct order, to achieve a correct answer on submission.

Component	Properties
1.Form6 – Bonus Games	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Image: CustomBonusGames.jpeg Text: Null
2.BonusGamesExitlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
3.BonusGamesHelplbl	Size: 100, 100 Back Colour(RGB): Transparent

	Text: “”
4.QuestionLBL	Text; ”Spanish Phrase (English Translation)” Colour: Active Caption Size: 290, 60
5.pbAnswer1-3	Style: Fixed3D Size: 80, 80 Image: Null Text: Null
6.DragLbl1-12	Size: 80, 80 Back colour: White Size; 80, 80 Image: CustomBonusGamesANS1-12.jpeg
7.BonusGamesSubmitlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image: CustomBonusGamesSubmit.jpeg
BonusGamesRightlbl Appears on control 7 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image: CustomBonusGamesSubmitRight.jpeg
BonusGamesWronglbl Appears on control 7 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “” Image:CustomBonusGamesSubmitWrong.jpeg
BonusGamesInfolbl Appears on control 3 Click()	Size: 300, 260 Back Colour(RGB): Control Image: CustomInfo.jpeg

Leader Board Form

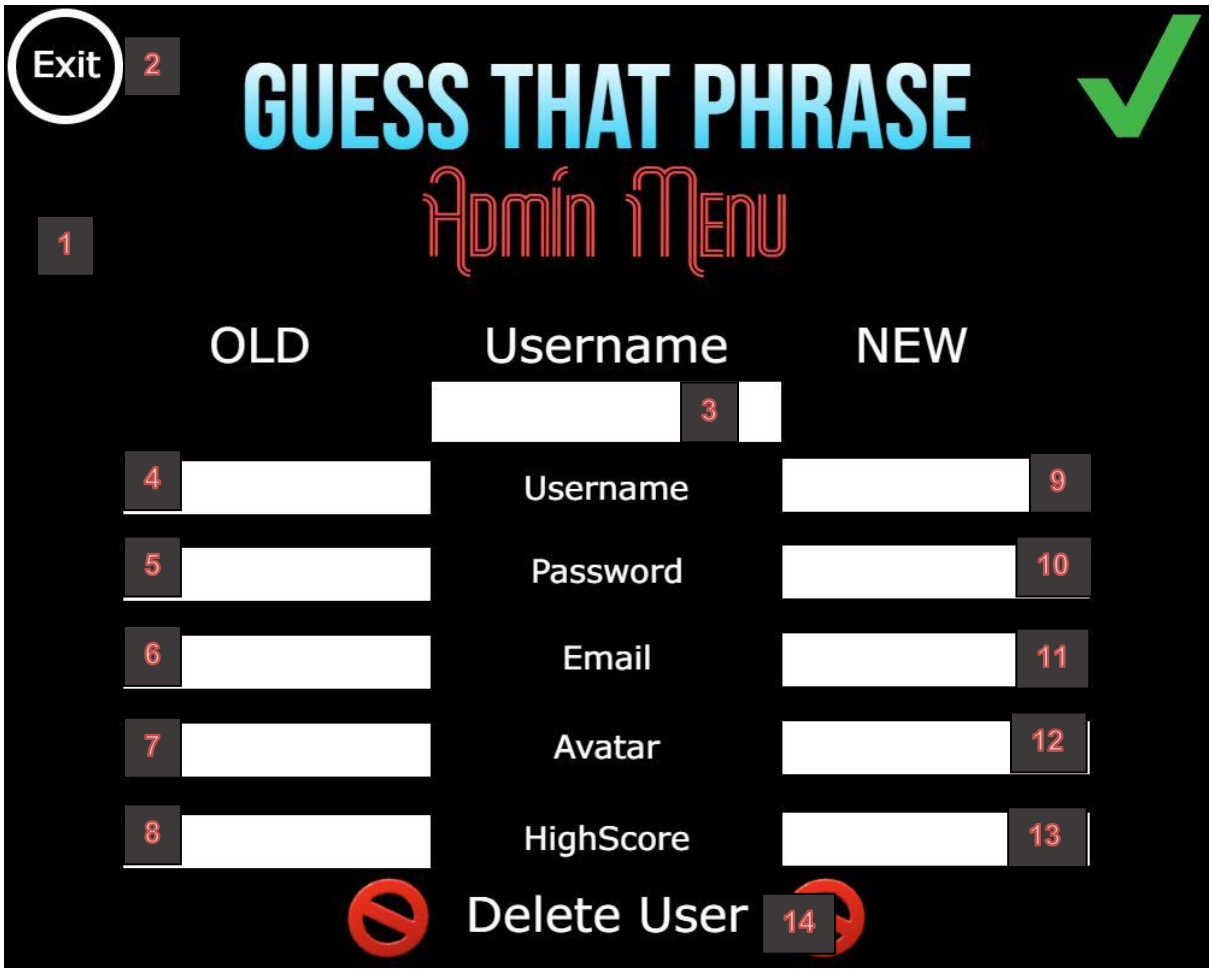


The LeaderBoard form serves as a way for users to view their score along with the other users within the application. The users with the highest score are placed atop the table with the lower scoring users being at the bottom. The Leader Board will display the top 3 users with their scores being taken from the hard version of Guess That Phrase as it will be the most competitive and require the most skill to score highly on.

Component	Properties
1.Form7- Leader Board	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Text: Null Image:CustomLeader.jpeg
2.LeaderBoardInfolbl	Size: 300, 260 Back Colour(RGB): Control Image: CustomLeaderInfo.jpeg
3.LeaderBoardExitlbl	Size: 100, 100

	Back Colour(RGB): Transparent Text: “”
LeaderBoardHelplbl Appears on control 2 Click()	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
4.LeaderBoardGif	Size: 500, 500 Back Colour(RGB): Control Image: CustomLeaderGif.gif
5.FirstPlaceLbl	Size:220, 60 Colour: Active Caption Font: Microsoft YaHei UI Font size : 27.75pt
6.SecondPlaceLbl	Size:220, 60 Colour: Active Caption Font: Microsoft YaHei UI Font size : 27.75pt
7.ThirdPlaceLbl	Size:220, 60 Colour: Active Caption Font: Microsoft YaHei UI Font size : 27.75pt
8.FirstPlaceNumberLbl	Size :50, 50 Colour : Gold Font: Microsoft YaHei UI Font size : 27.75pt
9.SecondPlaceNumberLbl	Size :50, 50 Colour ; Silver Font: Microsoft YaHei UI Font size : 27.75pt
10.ThirdPlaceNumberLbl	Size :50, 50 Colour : Saddle Brown Font: Microsoft YaHei UI Font size : 27.75pt

Admin Form



Based on the peer input and self-review that was conducted in the initial storyboard designs, I concluded that only the admin user would be allowed to change aspects of the user object. The original process was decided to be a security risk due to a potential attacker being able to fully shut out the user from accessing their account by altering their login details.

Component	Properties
1.Form8- AminPage	Size: 1000, 800 FormBorderStyle: None Back Colour(RGB): Null Text: Null Image:CustomAdminPage.jpeg
2.AdminPageExitlbl	Size: 100, 100 Back Colour(RGB): Transparent Text: “”
3.userDropDown	Size: 300, 50 Back colour: Control Font: Microsoft Sans Serif

	Font size ; 20.25pt
4.oldUserNamelbl	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
5.oldPasswordlbl	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
6.oldEmaillbl	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
7.oldAvatarlbl	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
8.oldScorelbl	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
9.newUserNamelbl	Size: 250, 40 Text: “Cannot update Username” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
10.newPasswordTB	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
11.newEmailTB	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
12.newAvatrTB	Size: 250, 40 Text: “”

	Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
13.newScoreTB	Size: 250, 40 Text: “” Back colour: Control Font: Microsoft Sans Serif Font size ; 20.25pt
14.deletelbl	Size: 250, 40 Text: “Delete User” Font: Microsoft Sans Serif Font size ; 20.25pt

Test Plan

The following table denotes the tests carried out on the ‘Guess That Phrase’ application. These tests are in the form of unit tests and link tests which test individual elements of the application and the navigation between them.

Test NO & Test Type	Test	Test Data	Expected outcome	Actual outcome
			Splash Screen	
1.1 UNIT	Run the application	N/A	Splash screen should open and loading gif should play for 6 seconds	Function worked as planned, however 6 seconds felt to long, reduced to 5 seconds
1.2 UNIT	Wait for gif to conclude after 6 seconds	N/A	Splash screen form hide and Login form should open	Function work as designed
			Login Screen	
2.1 UNIT	Login in with a pre registered username with a incorrect password	PreRegUser123 InvalidPass321	Message box should appear with “Password incorrect” text. Main Menu form should open	Function worked as intended
2.2 UNIT	Login in with a pre registered username with a correct password	PreRegUser123 ValidPas35@@	Message box should appear with “Login Successful” text. Main Menu form should open	Function worked as planned, however some minor spelling corrections needed
2.3 LINK	Click the Sign in options label	N/A	Login Form should close and Register Form should open	Function worked as intended
2.4 UNIT	Click the forgot Password button and enter username	Username123	The admin should receive a forgotten password request with the username	Function worked as intended
2.5 UNIT	Login with admin credentials	admin1234 AA!!11adminpass	Login Form should close and AdminScreen should open	Function worked as intended

			Register Screen	
3.1 UNIT	Register with an invalid username	bob123	Message box should appear with “Username does not meet the requirements” or	Function worked as intended, with username taken text only appearing with all other fields filled appropriately
		TakenName123	Message box should appear with ”Username is taken”	
3.2 UNIT	Register with an invalid password	invalidPass	Message box should appear with “Password does not meet the requirements”	Function worked as intended
3.3 UNIT	Register with valid credentials	FreeName123 email@ ValidPas35@@ ValidPas35@@	Message box should appear with “Success please login”. Credentials written to database and login form open	Function worked as planned, with an Excel file being used instead of a plain text file for more convenient storage
3.4 LINK	Click on arrow buttons to control avatar select	N/A	Avatar label should update according to which button is pressed	The left and right avatar buttons do not cycle the avatar picture correctly the order seems to be partly random
3.5 LINK	Click the already have an account label	N/A	Register Form should close and Login Form should open	Function worked as intended
			Main Menu Screen	
4.1 LINK	Click the Play button	N/A	Main Menu Game Info label will be visible along with play buttons	Function worked as planned, small adjustments to make sure buttons correctly overlap labels
4.2 LINK	Click the Easy Mode label	N/A	Menu form will close and the Game form will open	Function worked as planned
4.3 LINK	Click the Hard Mode label	N/A	Menu form will close and the GameHard	Function worked as intended

			form will open	
4.4 LINK	Click the Bonus label	N/A	Menu form will close and the BonusGames form will open	Function worked as intended
4.5 LINK	Click the Leader Board button	N/A	Menu form will close and the Leader Board form will open	Function worked as intended
4.6	Click the Options button	N/A	Options label will turn visible	Function worked as intended
4.7 LINK	Click the Exit button	N/A	Menu form will close Login Form will open	Function worked as intended
4.8 LINK	Click the Help button	N/A	Help label will turn visible	Function worked as planned small adjustments to make ensure correct overlap between labels
			GameEasy Screen	
5.1 UNIT	GameEasy Form load	N/A	Game start sequence timer should start and traffic light labels should play for 5 seconds and GameEasyReadylbl should display for 5 seconds	Function worked as intended
5.2 UNIT	GameEasy start sequence timer stop	N/A	GameEasyReadylbl should not be visible, gameEasyQlbl should randomly be displayed	Function worked as intended
5.3 UNIT	submitGameEasy pressed wrong answer in EasyPhraseAnsTB	WrongANS	An incorrect X should appear on the screen and the question label should change with no change in score, AnsTB should clear	Function worked as intended
5.4 UNIT	submitGameEasy pressed right	Data can vary possible	An correct tick should appear on	If a space is included in the answer text it will return an

	answer in EasyPhraseAnsTB	examples: Flag ship Football world cup	the screen and the question label should change, while incrementing the score tb, AnsTB should clear	incorrect value
5.5 LINK	GameExitlbl pressed	N/A	Game form should close and Main Menu form should open	Function worked as intended
5.6 LINK	GameHelplbl pressed GameHelplbl pressed again	N/A	GameInfolbl should become visible(on second click it should no longer be visible)	Function worked as intended
5.7 UNIT	Clocklbl reaches finish position	N/A	GameEasyGameOverlbl should be visible along with Play again label and Main Menu label	Function worked as intended
5.9 LINK	Play again label pressed	N/A	Game Form should reload	Function worked as intended
5.9 LINK	Main Menu label pressed	N/A	Game Form should close and Main Menu Form should open	Function worked as intended
			GameHard Screen	
6.1 UNIT	GameHard form load	N/A	Game start sequence timer should start and traffic light labels should play for 5 seconds and GameHardReadylbl should display for 5 seconds	Function worked as intended
6.2 UNIT	GameHard start sequence timer stop	N/A	GameHardReadylbl should not be visible, gameHardQlbl should randomly be displayed	Function worked as intended

6.3 UNIT	submitGameHard pressed wrong answer in EasyPhraseAnsTB	WrongANS	An incorrect X should appear on the screen and the question label should change with no change in score. Clocklbl should continue to move left	Function worked as intended
6.4 UNIT	submitGameHard pressed correct answer in HardPhraseAnsTB	Data can vary possible examples: Walking dead Strike while the iron is hot	An correct tick should appear on the screen and the question label should change, while incrementing the score tb. Clocklbl should jump right by 150 pixels	The clock label seems to increment off the screen if the user gets multiple correct answers consecutively. A 150 pixel increment seems to be too great reducing hard levels difficulty
6.5 LINK	GameHardExitlbl pressed	N/A	GameHard form should close and Main Menu form should open	Function worked as intended
6.6 LINK	GameHardHelplbl pressed GameHelplbl pressed again	N/A	GameHardInfolbl should become visible while stopping the timer(on second click it should no longer be visible) timer should stop	Label does not fully appear when called and timer is not restarted when label is off screen
6.7 UNIT	Clocklbl reaches finish position	N/A	GameHardGameOverlbl should be visible along with Play again label and Main Menu label	Function worked as intended
6.8 LINK	Play again label pressed	N/A	GameHard Form should reload	Function worked as intended
6.9 UNIT	Main Menu label pressed	N/A	GameHard Form should close and Main Menu Form should open	Function worked as intended
			BonusGames Screen	
7.1	Bonus Game	N/A	Random spanish	Function worked as intended

UNIT	form load		question should appear in BomusGameQuestionTB with english translation	
7.2 UNIT	Drag a Drag label into each picture box	N/A (Press and hold with drag input required)	All Bonus Game picture boxes can receive an image from each Drag labels	Function worked as intended
7.3 UNIT	Bonus Games submit label pressed with correct value in each picture box	Data can vary in this example	BonusGameCorrectlbl should be visible	The answer is always wrong regardless of if the labels are in the correct order
7.4 UNIT	Bonus Games submit label pressed with incorrect value in each picture box	Data can vary in this example	BonusGameIncorrectlbl should be visible	Function worked as intended
7.5 LINK	Bonus Games help label pressed	N/A	BonusGameInfoilbl should be visible	Function worked as intended
7.6 LINK	Bonus Games Exit label pressed	N/A	Bonus Game Form should close and main menu form should load	Function worked as intended
			Leader Board Screen	
8.1 UNIT	LeaderBoard Form load	N/A	LeaderBoard gif should play and top 3 highscores should be displayed in order	The Leader Board screen seems to display the top scoring user in the number 3 position
8.2 LINK	LeaderBoard help label pressed	N/A	LeaderBoardInfolbl should be visible	Function worked as intended
8.3 LINK	LeaderBoard exit label pressed	N/A	LeaderBoard Form should close and main menu form	Function worked as intended

			should open	
			Admin Screen	
9.1 UNIT	Dropdown menu open	N/A	Contains every username registered in the database	Function worked as intended
9.2 UNIT	Email notification about password updates	“QQ00@@ssdtest”	Email is sent about password change to respective email	The system threw an error during this function reason unclear
9.3 UNIT	Dropdown item selected	“EmailTest”	Fills labels with users corresponding Database data	Function worked as intended
9.4 UNIT	Update textboxes with new data	Score value to 100	User information updated in the Database	Function worked as intended
9.5 LINK	AdminExitlbl pressed	N/A	User is taken to login form	Function worked as intended

Test Plan Results

Test NO.	Was the test successful?	Corrective action
	Splash Screen	
1.1	Yes	N/A
1.2	Yes	N/A
	Login Screen	
2.1	Yes	N/A
2.2	Yes	N/A
2.3	Yes	N/A
2.4	Yes	N/A
2.5	Yes	N/A
	Register Screen	
3.1	Yes	N/A
3.2	Yes	N/A
3.3	Yes	N/A
3.4	No	The previous switch statements may have been overcomplicated, thus if statements were used alongside rechecking the label names to ensure they were being made visible correctly
3.5	Yes	N/A
	Main Menu Screen	
4.1	Yes	N/A
4.2	Yes	N/A
4.3	Yes	N/A
4.4	Yes	N/A
4.5	Yes	N/A
4.6	Yes	N/A
4.7	Yes	N/A

4.8	Yes	N/A
	GameEasy Screen	
5.1	Yes	N/A
5.2	Yes	N/A
5.3	Yes	N/A
5.4	No	Due to a misunderstanding of the String.Trim method the program did not remove space characters between substrings, thus it could not relate to the key in the question dictionary. Rectified by using String.Replace “ ” with “”.
5.5	Yes	N/A
5.6	Yes	N/A
5.7	Yes	N/A
5.8	Yes	N/A
5.9	Yes	N/A
	GameHard Screen	
6.1	Yes	N/A
6.2	Yes	N/A
6.3	Yes	N/A
6.4	No	To prevent the clock label from incrementing off screen a new int xMax was set to determine the maximum x position that the clock label can move to. The 150 pixel increment was reduced to a more conservative 100 pixel movement to ensure thorough challenge
6.5	Yes	N/A
6.6	No	Help label was not appearing due to overlapping labels, with the timer function being disabled rather than stopped preventing the .Start() method from functioning as needed
6.7	Yes	N/A

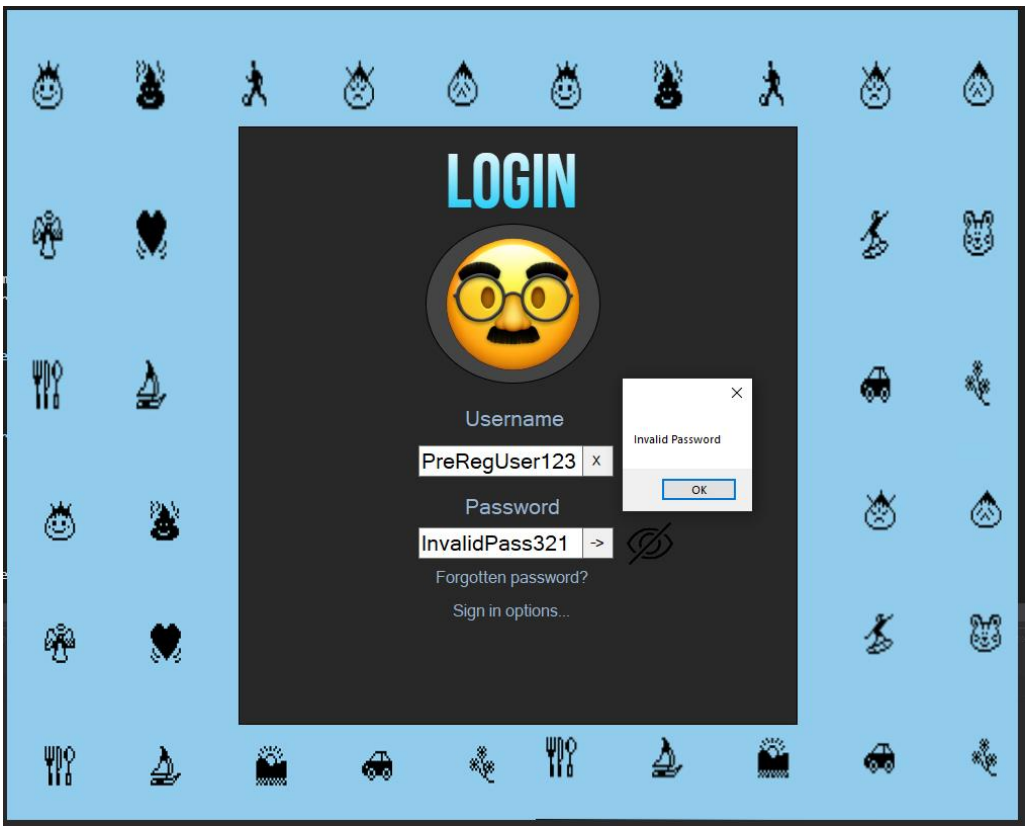
6.8	Yes	N/A
6.9	Yes	N/A
	BonusGames Screen	
7.1	Yes	N/A
7.2	No	Due to an oversight the image details string never matched the value in the question dictionary as there were no commas at the end of the final substring in the value element of the dictionary
7.3	Yes	N/A
7.4	Yes	N/A
7.5	Yes	N/A
7.6	Yes	N/A
	LeaderBoard Screen	
8.1	No	Due to a mistake in label load order the information of each label was ranked wrong. With the top scoring user being place in the bottom position. Solved with a revising of how information is passed into each label and a more robust getLeaderBoardinfo() Method that was not responding due to “,,,,” instead of “” in database.csv file.
8.2	Yes	N/A
8.3	Yes	N/A
	Admin Screen	
9.1	Yes	N/A
9.2	No	The method used to send the Admin email left the email without encryption this caused google mail to flag the email as an unsafe object causing the method to return false.
9.3	Yes	N/A
9.4	Yes	N/A
9.5	Yes	N/A

Evidence of testing

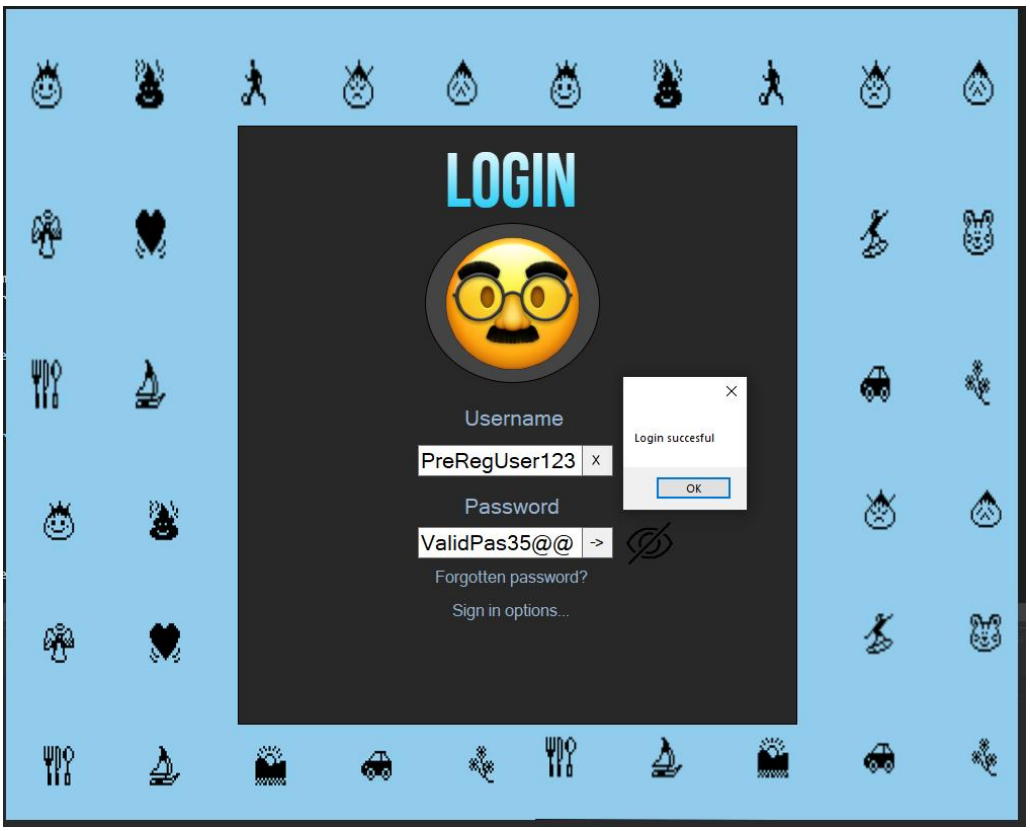
Testing evidence will be split into three sections: Data capture(), Link() and User Acceptance().

Data Capture Evidence

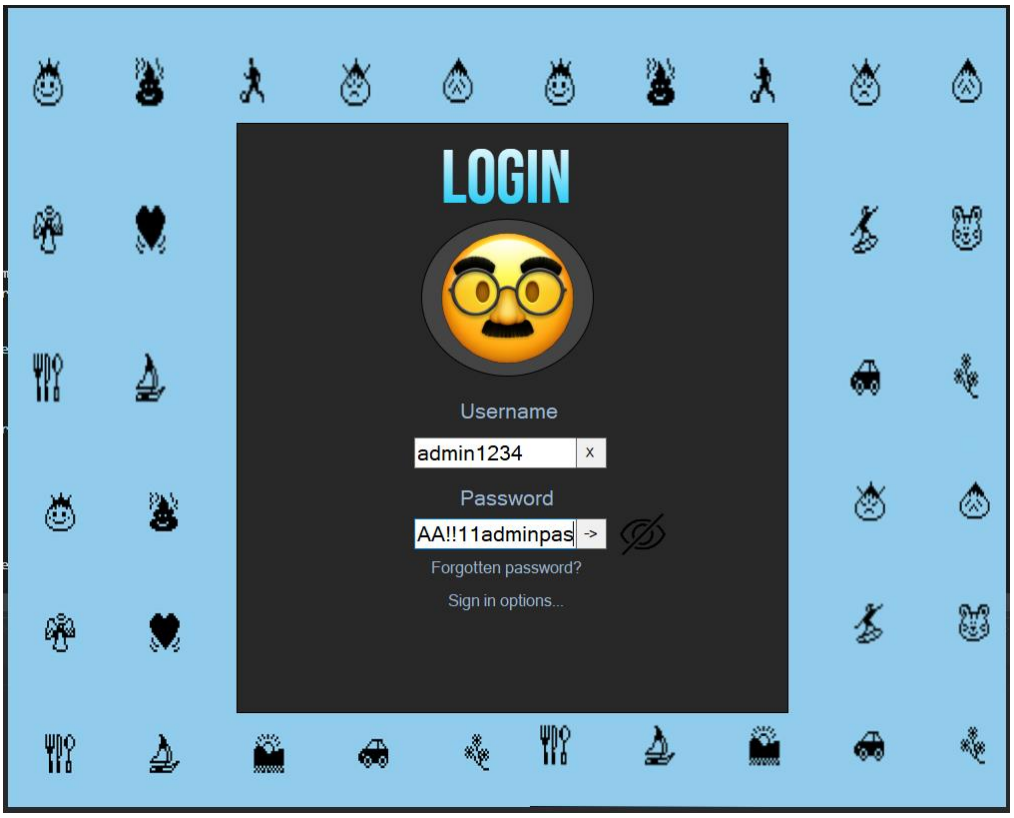
Test 2.1



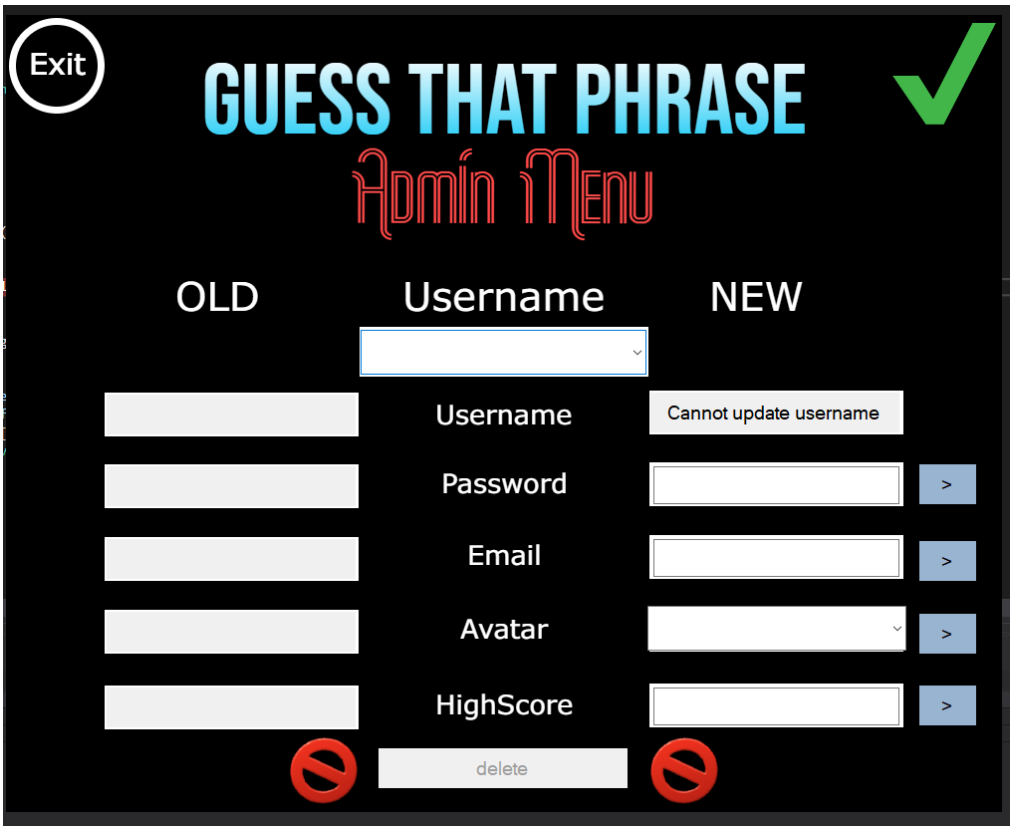
Test 2.2



Test 2.5



On Submitlbl.Click()



Tests 3.1

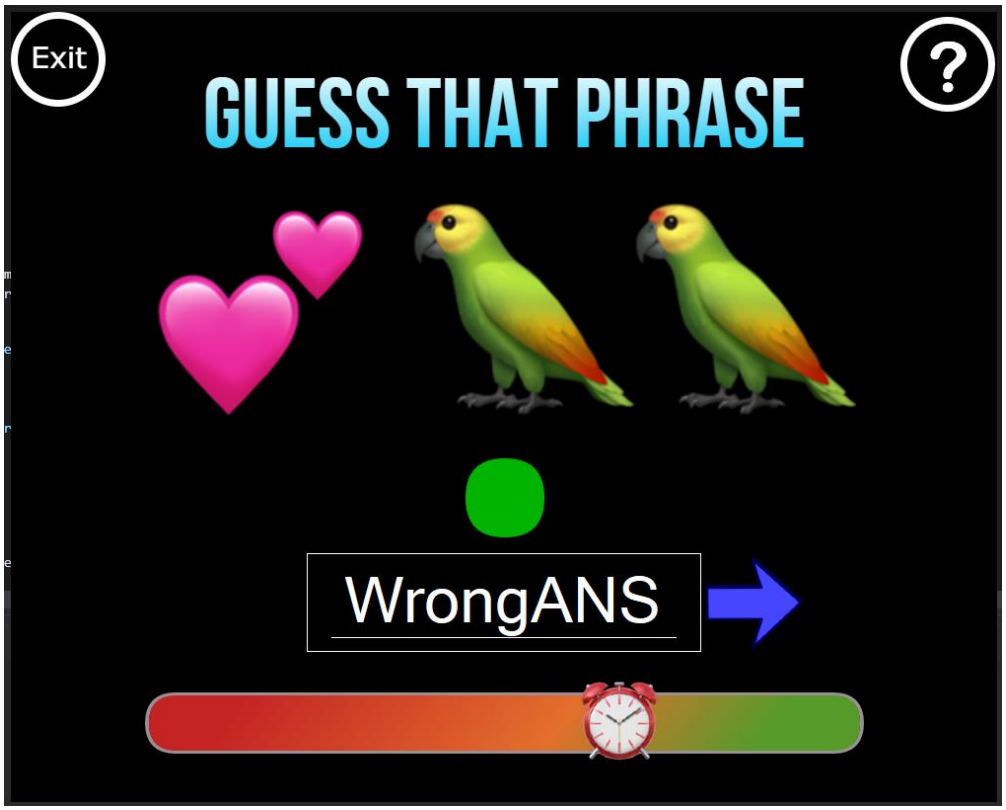

REGISTER
Username
TakenName123
Email Address
email@
Password
ValidPas35@@
Confirm Password
ValidPas35@@
Already have an account?
Username is taken
OK


REGISTER
Username
bob123
Email Address
Username does not meet requirements
OK
Already have an account?

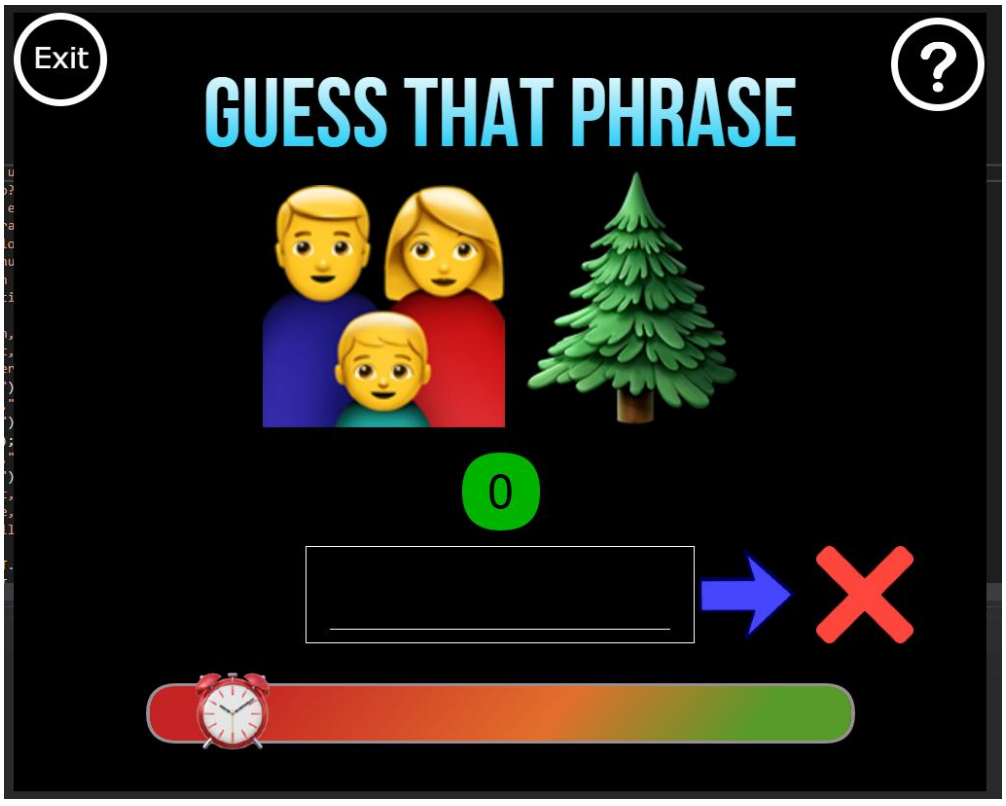
Test 3.2

Test 3.3

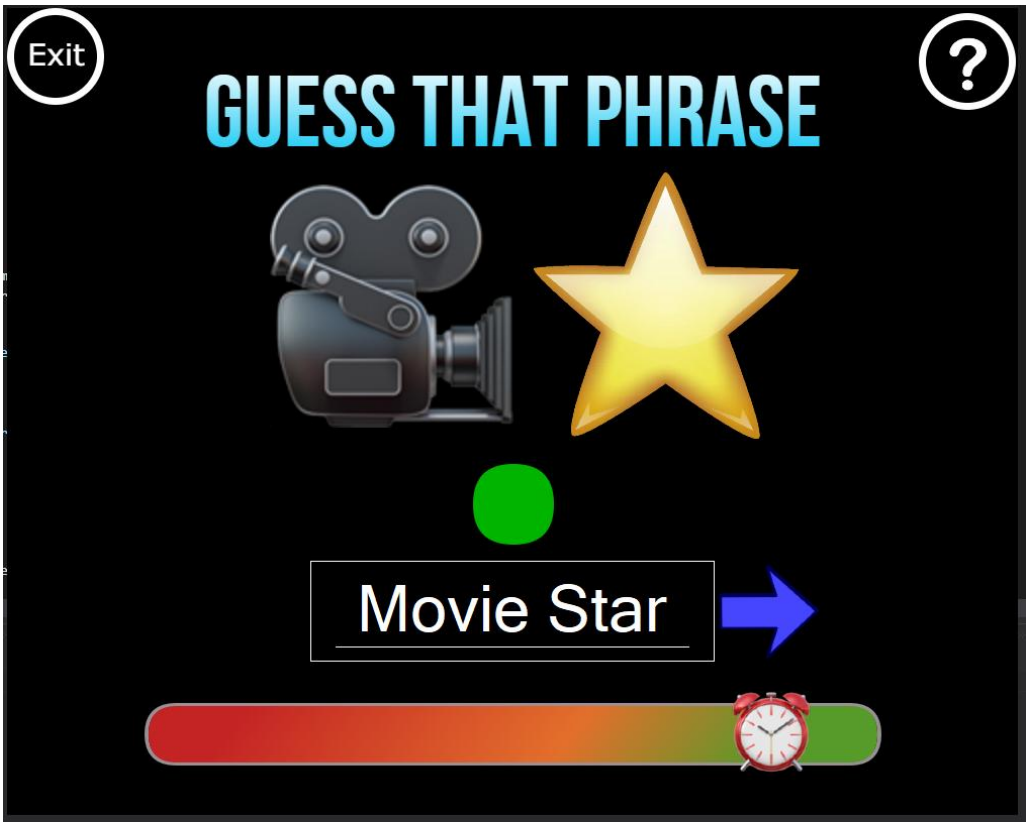
Test 5.3



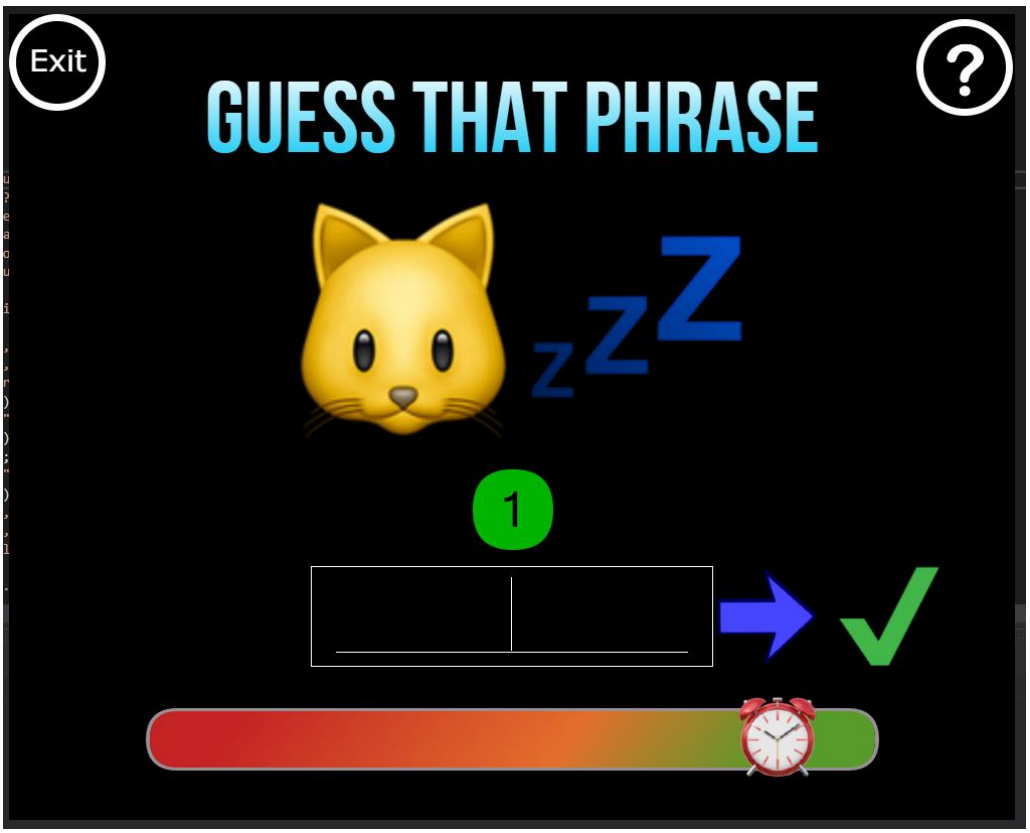
On SubmitGame.Click()



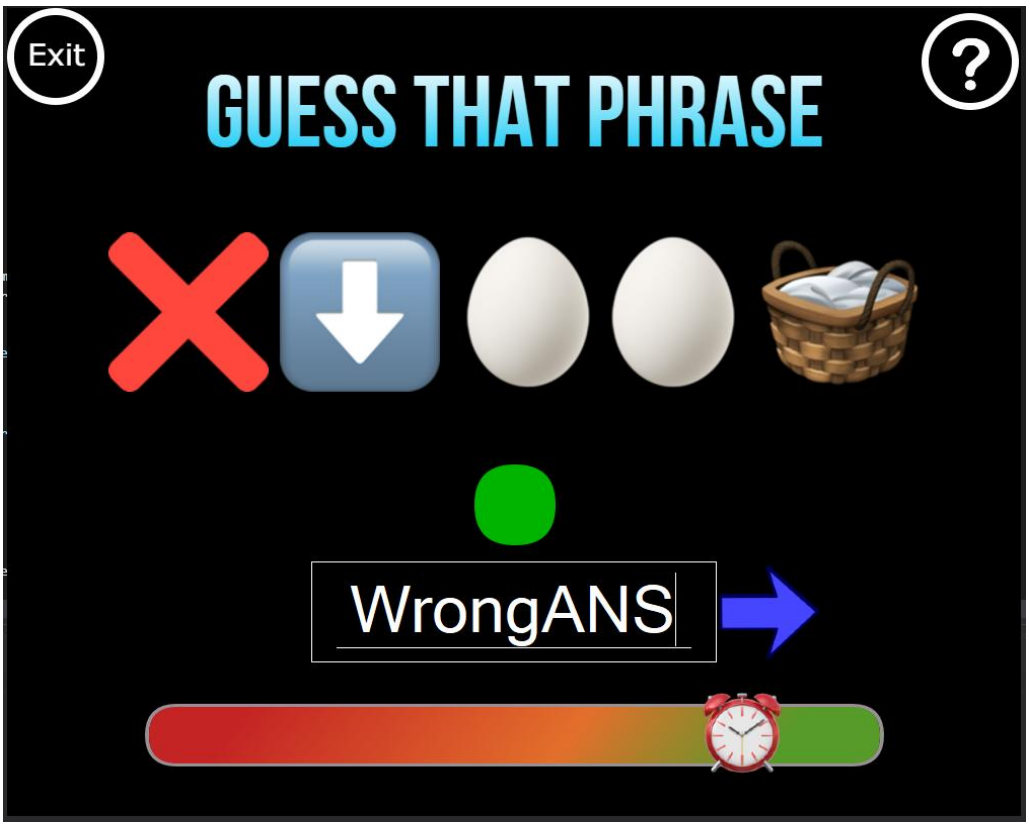
Test 5.4



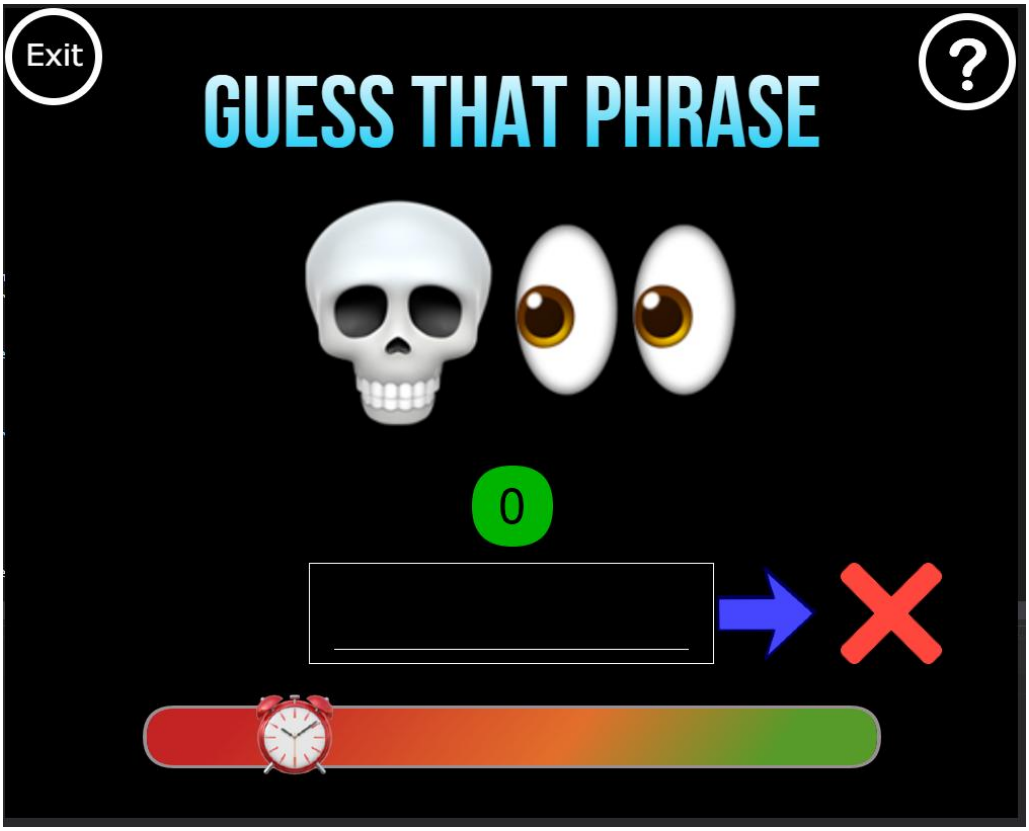
On SubmitGame.Click()



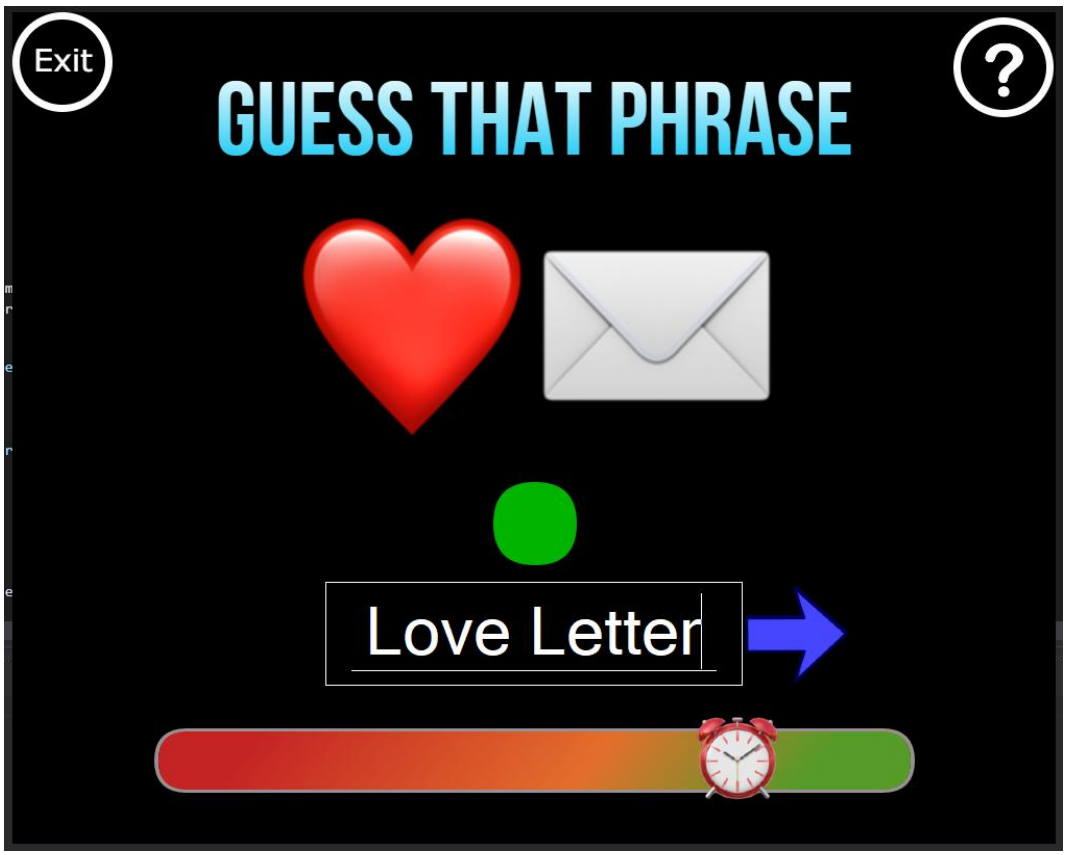
Test 6.3



On SubmitGameHard.Click()

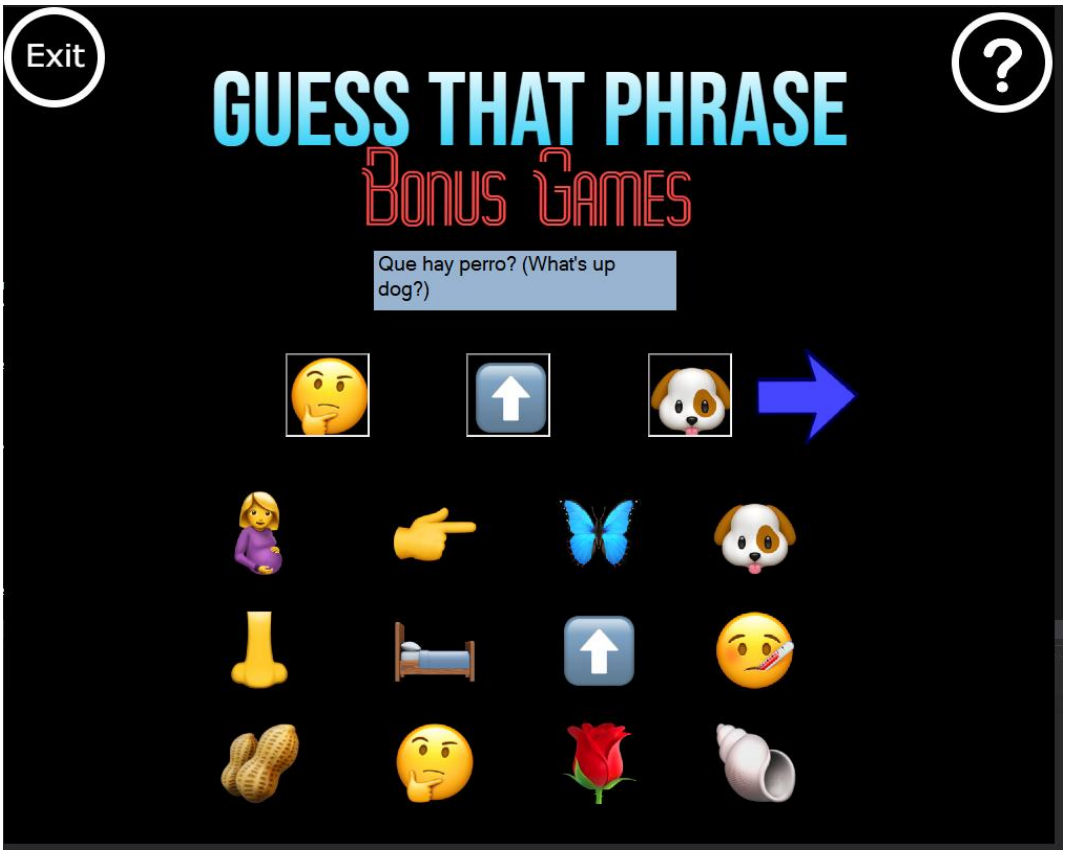


Test 6.4 (Corrective action taken problem resolved)

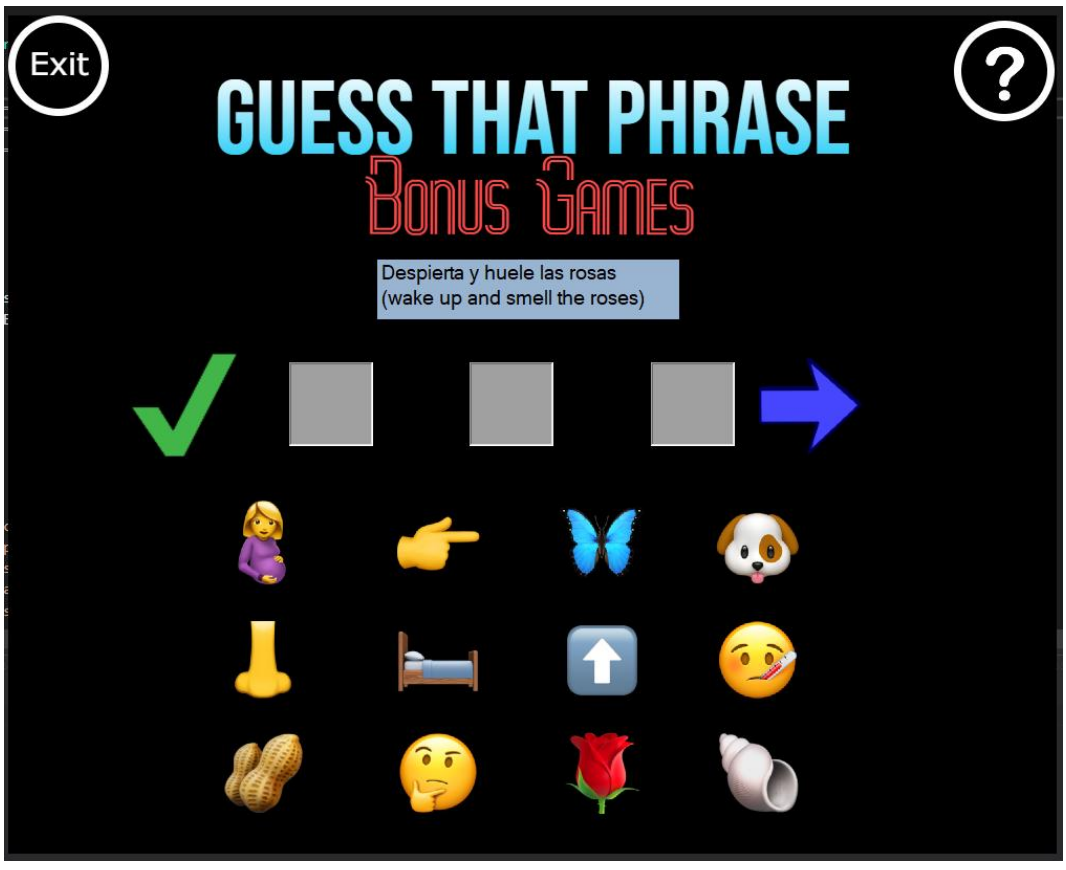


On SubmitGameHard.Click()

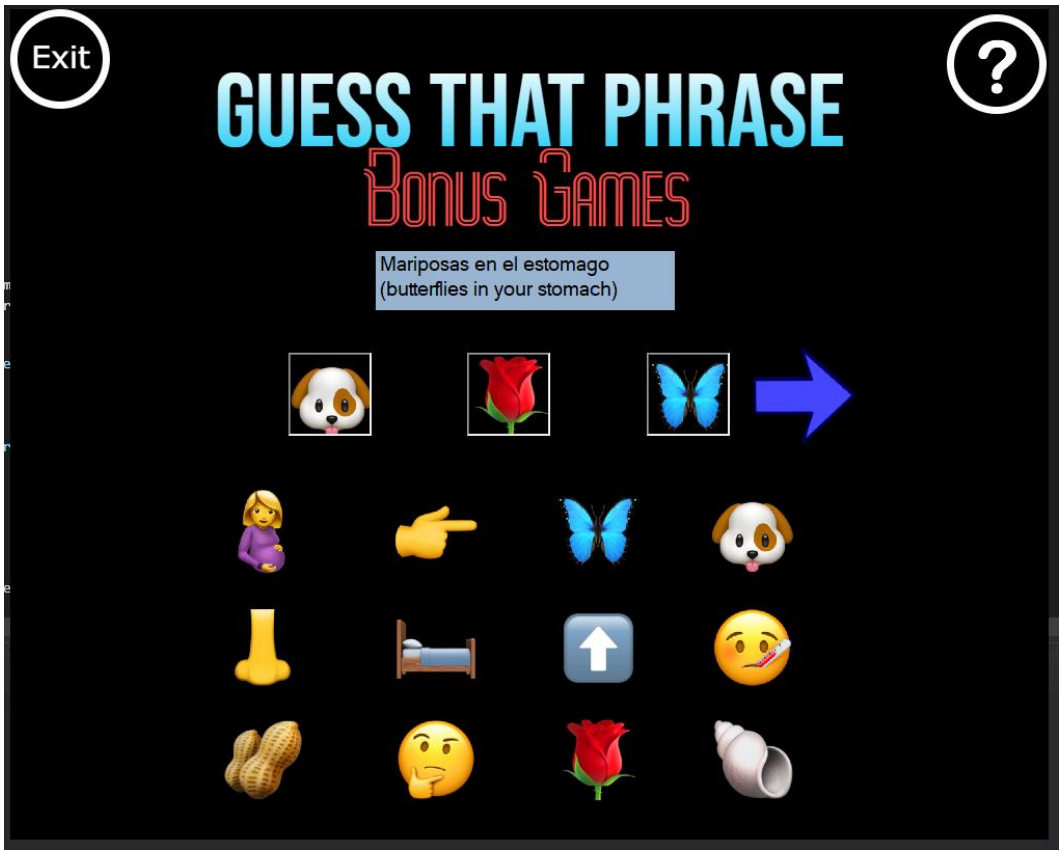
Test 7.2 and 7.3



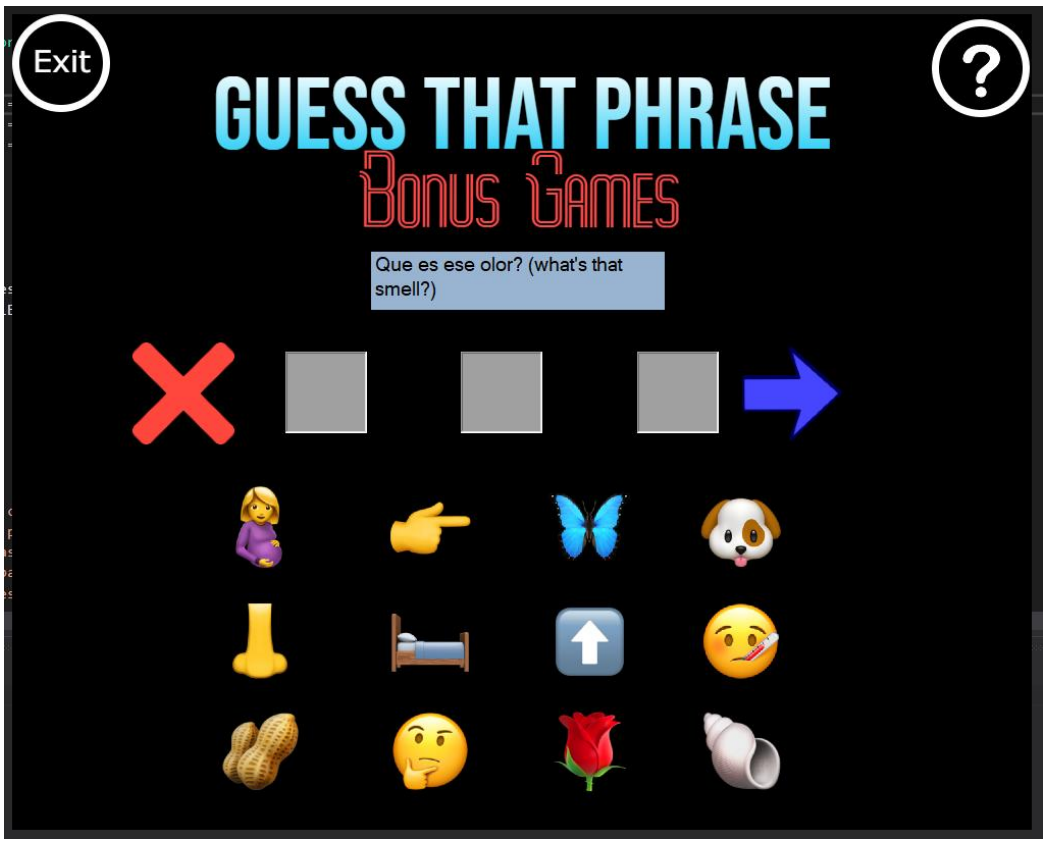
On SubmitGameBonus.Click()



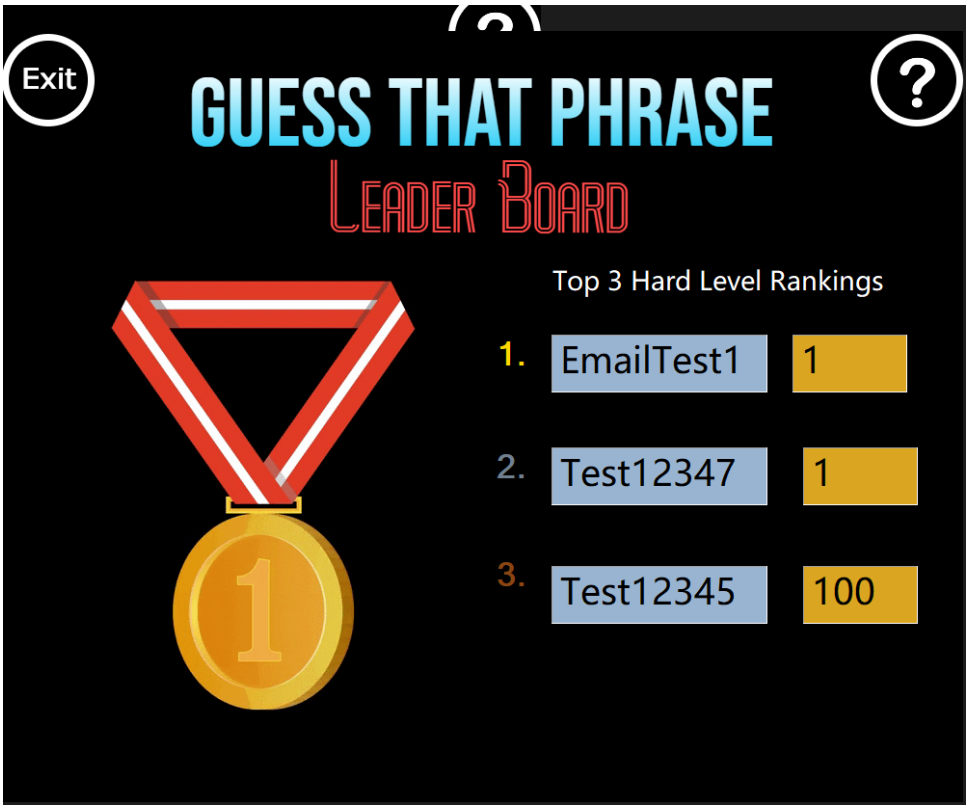
Test 7.4



On SubmitGameBonus.Click()



Test 8.1 (Corrective action taken function works as intended)

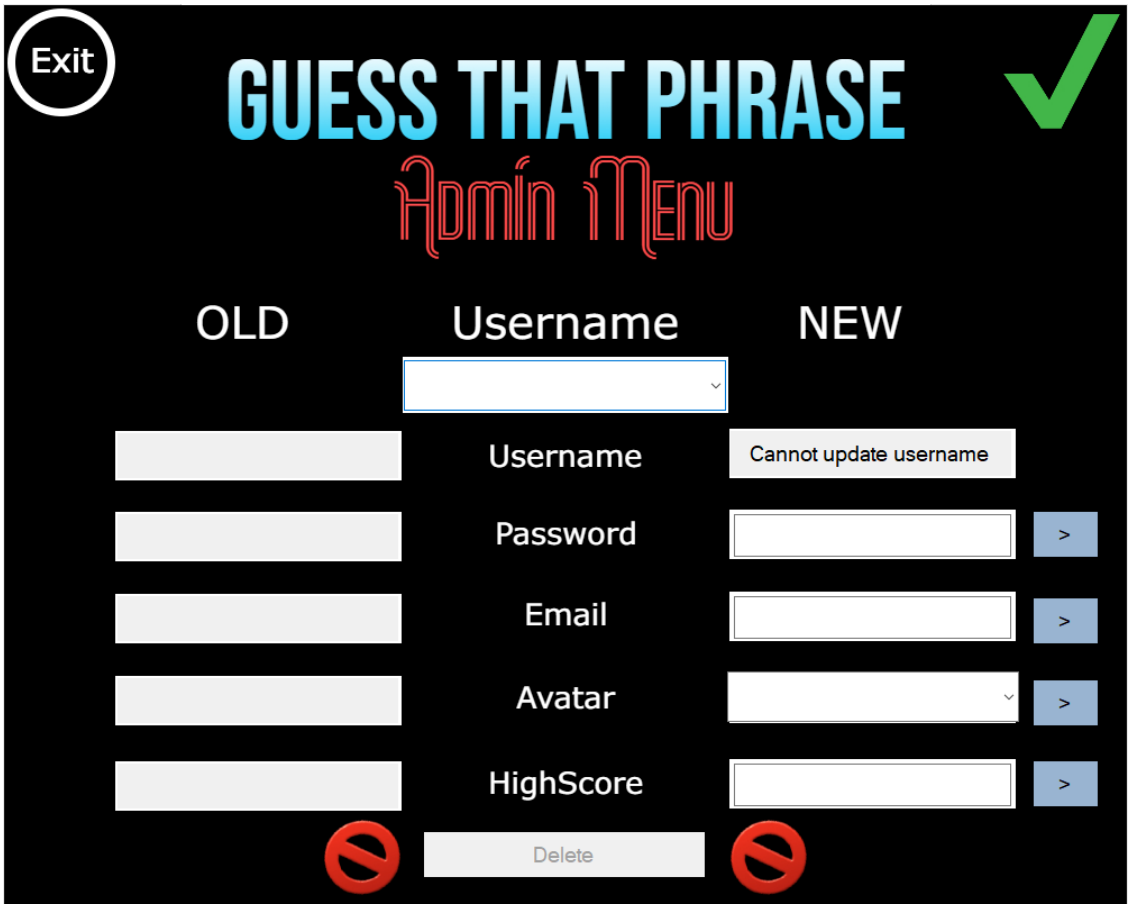


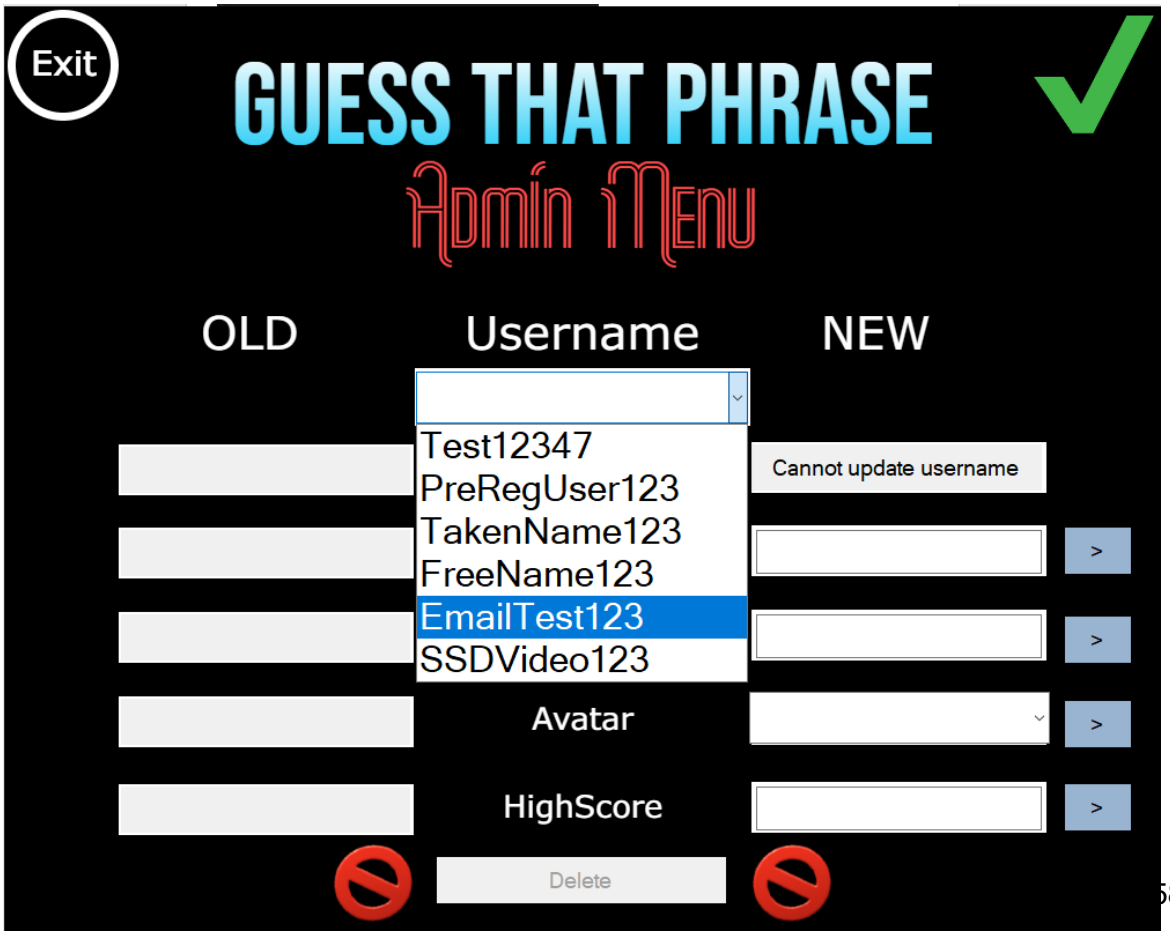
Pictured to below is the method held in the applications Database class was causing the above Leader Board error as pictured above due to a misinterpretation of the OrderByDescending() method when used in conjunction with a list this has now been resolved on the leader board screen by altering label positions and enhancing the robustness of the code held in the Leader Boards form class. The Leader Board form now works fully as intended with scores only being taken form the Game Hard level.

```
public List<User> getLeaderboardInfo()
{
    int count = 0;
    List<User> users = new List<User>();
    //Search the csv file
    try
    {
        string[] lines = System.IO.File.ReadAllLines("database.csv");


        for (int i = 0; i < lines.Length; i++)
        {
            if(lines[i] != ",,,,")//neede due to deleted user method not deleting commas(intented)
            {
                string[] fields = lines[i].Split(',');
                User user = new User(fields[0], fields[1], fields[2]);
                user.setHighScore(Convert.ToInt32(fields[4]));
                users.Add(user);
            }

        }
        //sort the user list by descending
        var sortedUsers = users.OrderByDescending(u => u.getHighScore());
        List<User> highestScoringUsers = new List<User>(3);

        //only bring back top 3
        for(int i = 0; i<sortedUsers.Count(); i++)
        {
            count++;
            if (count <= 3)
            {
                highestScoringUsers.Add(sortedUsers.ElementAt(i));
            }
            else
            {
                break;
            }
        }

        return highestScoringUsers;
    }
    catch (Exception ex1)
    {
        return users;
        throw new ApplicationException("This program did not work :", ex1);
    }

}
```

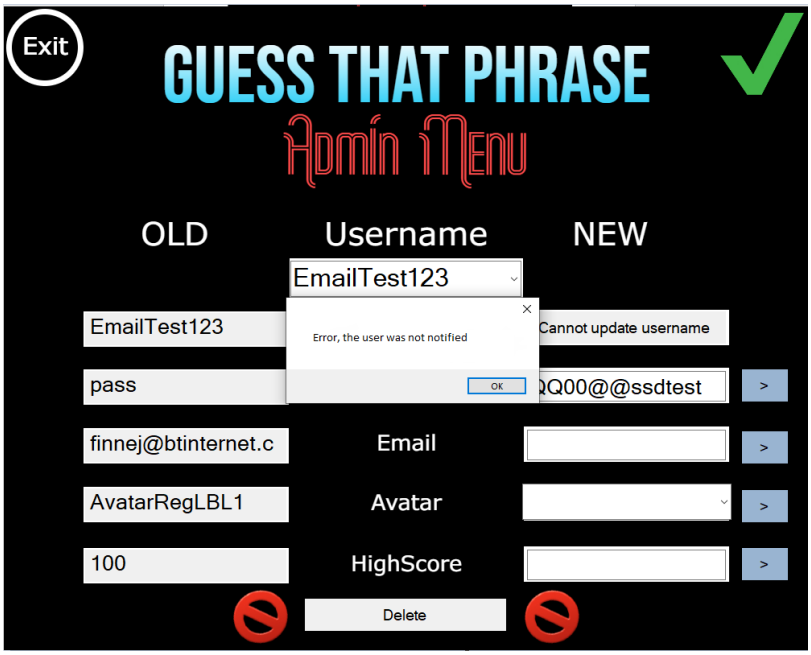
```
private void LeaderBoard_Load(object sender, EventArgs e)
{
    //create a new database object  to bring leaderboard info back username and score
    DataBase db = new DataBase();
    List<User> highestScoringUsers = db.getLeaderboardInfo();

    //put the names and higscores in descending order if they exist
    if (highestScoringUsers.Count() ==0)
    {

    }
    else if(highestScoringUsers.Count() ==1){
        FirstPlaceLblName.Text = highestScoringUsers.ElementAt(0).getUsername();
        FirstPlaceLblScore.Text = highestScoringUsers.ElementAt(0).getHighScore().ToString();
    }
    else if(highestScoringUsers.Count() == 2){
        FirstPlaceLblName.Text = highestScoringUsers.ElementAt(0).getUsername();
        FirstPlaceLblScore.Text = highestScoringUsers.ElementAt(0).getHighScore().ToString();
        SecondPlaceLblName.Text = highestScoringUsers.ElementAt(1).getUsername();
        SecondPlaceLblScore.Text = highestScoringUsers.ElementAt(1).getHighScore().ToString();
    }
    else if (highestScoringUsers.Count() == 3)
    {
        FirstPlaceLblName.Text = highestScoringUsers.ElementAt(0).getUsername();
        FirstPlaceLblScore.Text = highestScoringUsers.ElementAt(0).getHighScore().ToString();
        SecondPlaceLblName.Text = highestScoringUsers.ElementAt(1).getUsername();
        SecondPlaceLblScore.Text = highestScoringUsers.ElementAt(1).getHighScore().ToString();
        ThirdPlaceLblName.Text = highestScoringUsers.ElementAt(2).getUsername();
        ThirdPlaceLblScore.Text = highestScoringUsers.ElementAt(2).getHighScore().ToString();
    }


}
```

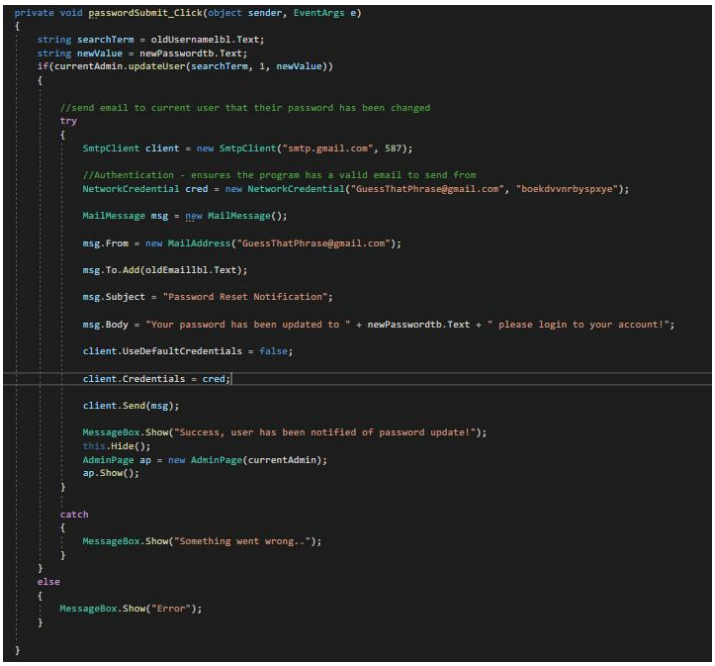
Test 9.1


Exit
GUESS THAT PHRASE
Admin Menu
OLD
Username
NEW
Username
Cannot update username
Password
Email
Avatar
HighScore
Delete


Exit
GUESS THAT PHRASE
Admin Menu
OLD
Username
NEW
Test12347
PreRegUser123
TakenName123
FreeName123
EmailTest123
SSDVideo123
Cannot update username
Avatar
HighScore
Delete

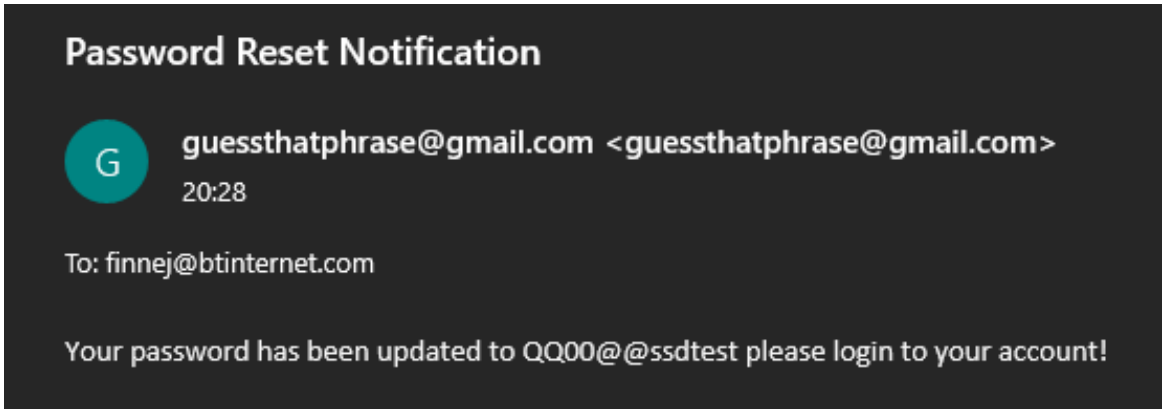
Test 9.2 (Corrective action taken issue resolved as pictured)



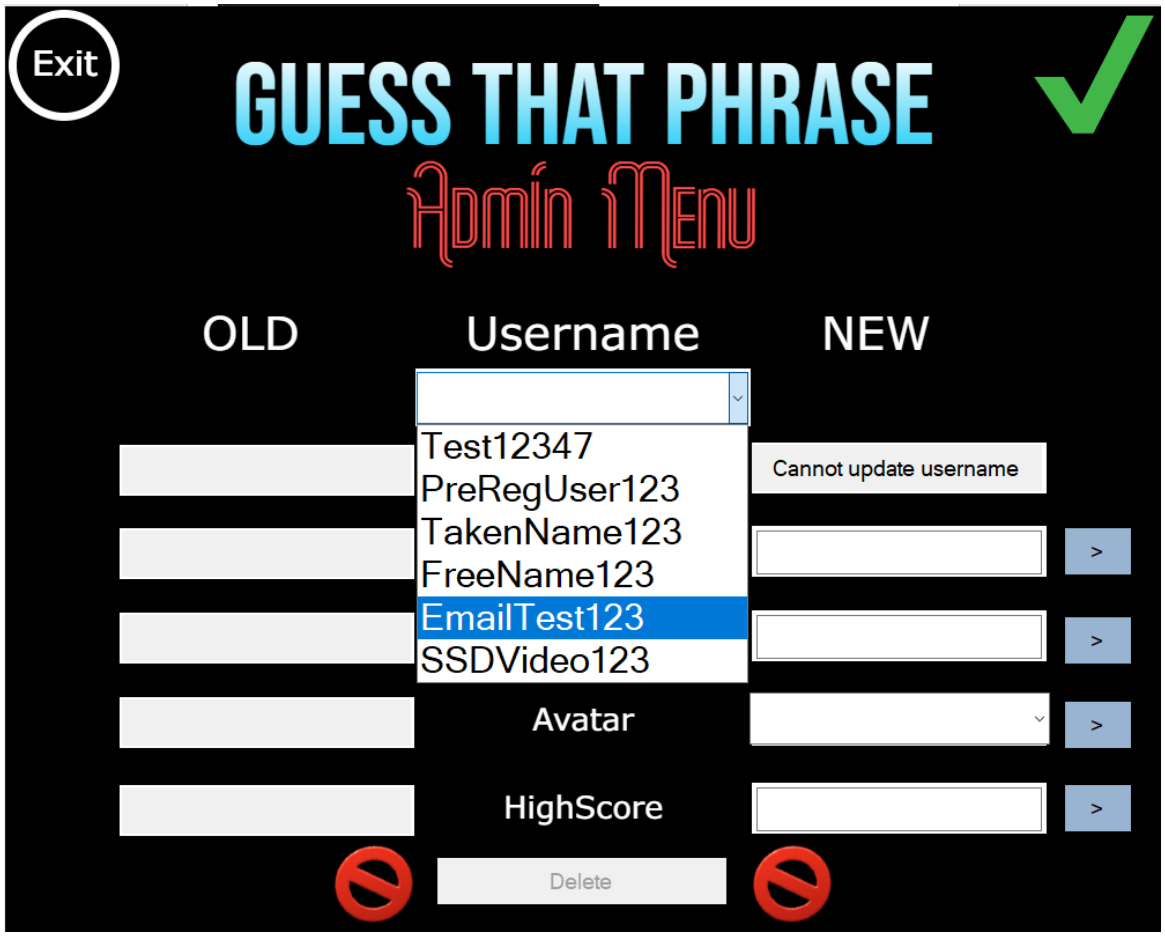


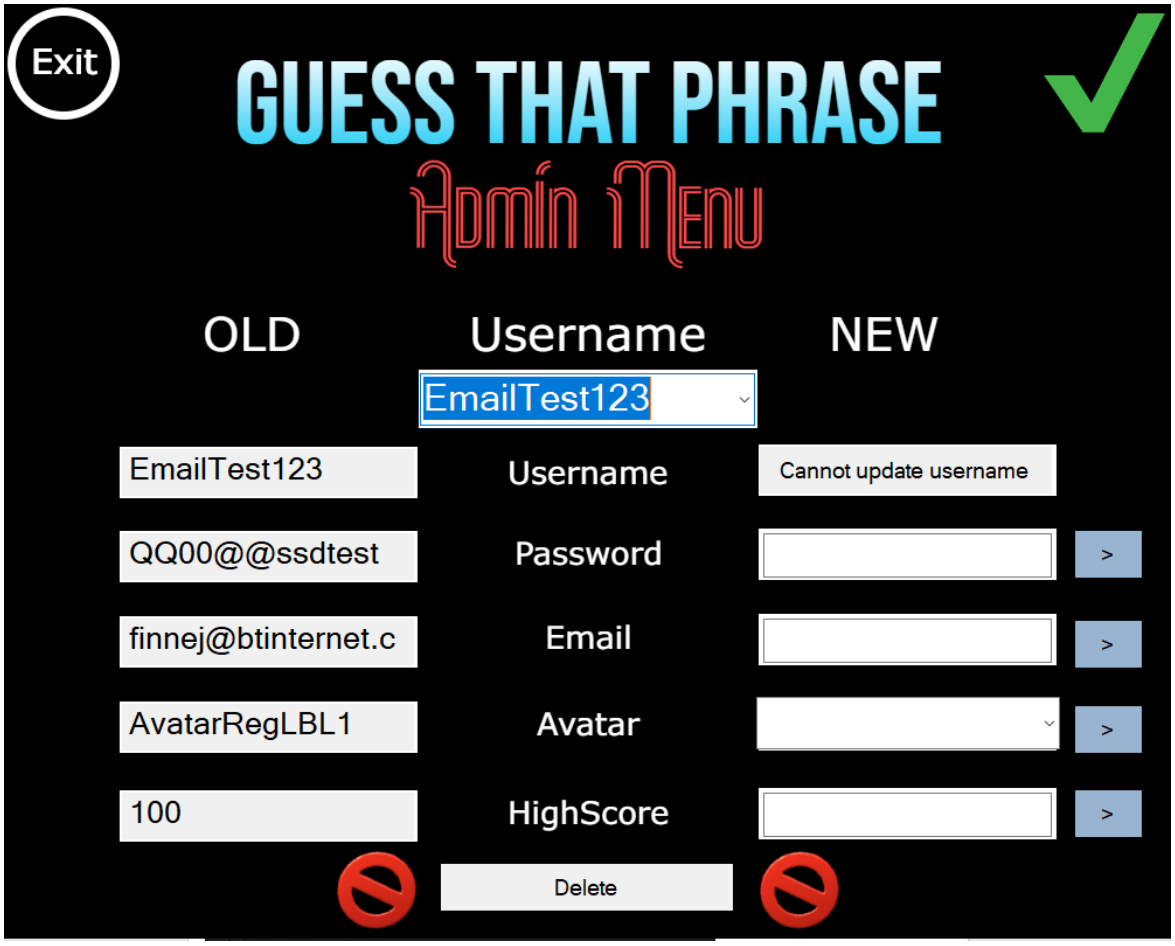
```
private void passwordSubmit_Click(object sender, EventArgs e)
{
    string searchTerm = oldUsernamelbl.Text;
    string newValue = newPasswordtb.Text;
    if(currentAdmin.updateUser(searchTerm, 1, newValue))
    {

        //send email to current user that their password has been changed
        try
        {
            SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

            //Authentication - ensures the program has a valid email to send from
            NetworkCredential cred = new NetworkCredential("GuessThatPhrase@gmail.com", "boekdvvnrbyspxye");

            MailMessage msg = new MailMessage();

            msg.From = new MailAddress("GuessThatPhrase@gmail.com");

            msg.To.Add(oldEmaillbl.Text);

            msg.Subject = "Password Reset Notification";

            msg.Body = "Your password has been updated to " + newPasswordtb.Text + " please login to your account!";

            client.UseDefaultCredentials = false;

            client.Credentials = cred;

            //enabling SSL (Secure sockets layer, encryption) this is required by most email providers to send mail
            client.EnableSsl = true;

            client.Send(msg);

            MessageBox.Show("Success, user has been notified of password update!");
            this.Hide();
            AdminPage ap = new AdminPage(currentAdmin);
            ap.Show();
        }

        catch
        {
            MessageBox.Show("Something went wrong..");
        }
    }
    else
    {
        MessageBox.Show("Error");
    }

}
```

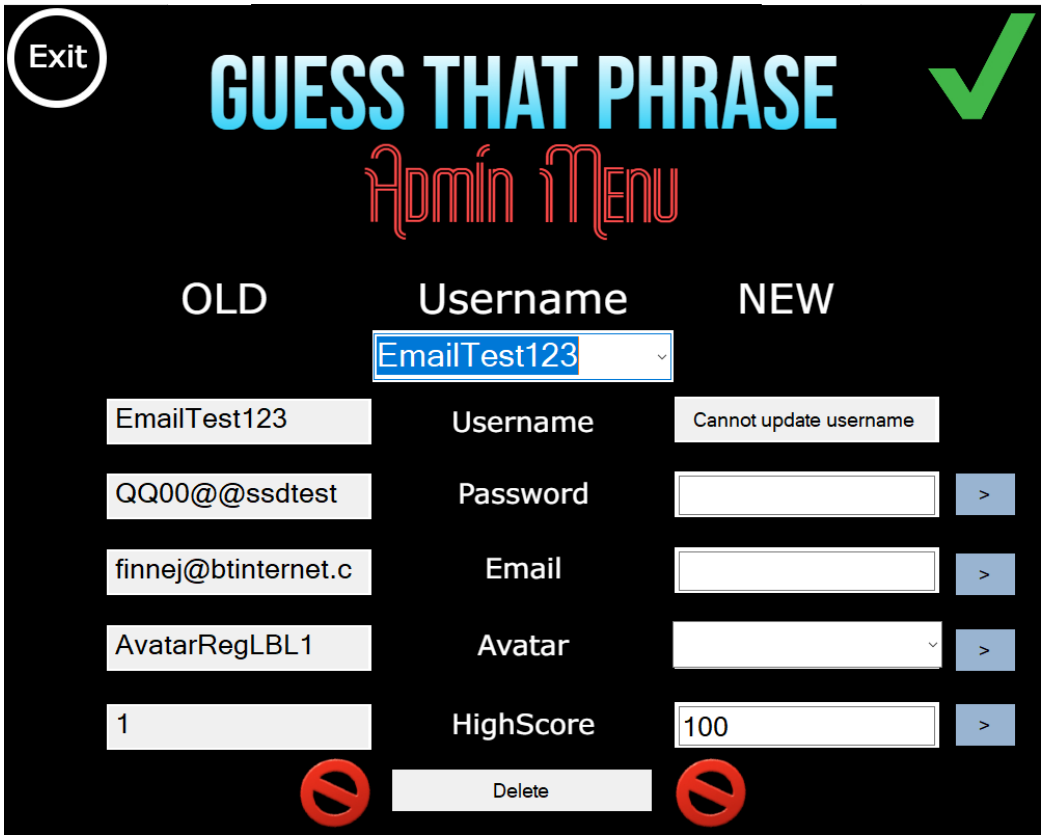
The error was found to be with the application not being a third party service with unauthorised access to the Google Mail Server. It was found due to looking through google mails Terms and Conditions, privacy statements, that all “all objects sent and received must be encrypted and have authorised access to Gmail Network”. This was surprising due to both email accounts having the minimum security and privacy settings active (done in order to pinpoint the specific error in test 9.2). In light of this, I then sought out the solution to encrypt the email and allow data transfer form the Guess That Phrase application over the internet into the destination email. It was found then that the encoding required by google mail was the SSL (Secure sockets layer, encryption) which was required by most major mail provides to sent mail. Thus the SmtpClient object was encrypted to this level.



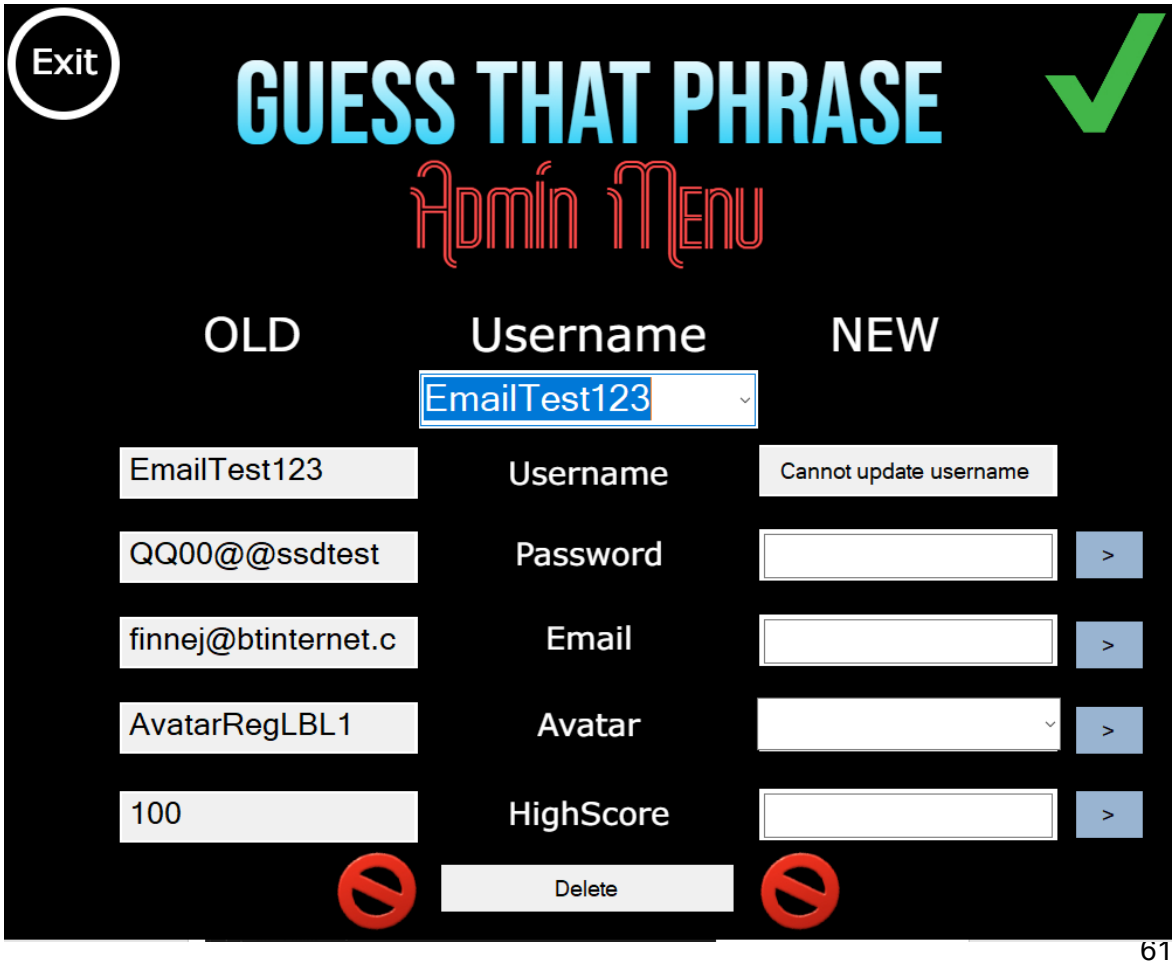
Test 9.3


Exit
GUESS THAT PHRASE
Admin Menu
OLD
Username
NEW
Test12347
PreRegUser123
TakenName123
FreeName123
EmailTest123
SSDVideo123
Cannot update username
Avatar
HighScore
Delete


Exit
GUESS THAT PHRASE
Admin Menu
OLD
Username
NEW
EmailTest123
EmailTest123
Username
Cannot update username
QQ00@@ssdtest
Password
finnej@btinternet.c
Email
AvatarRegLBL1
Avatar
100
HighScore
Delete



On ChangeScore Click()the users score was updated according to the textboxes value

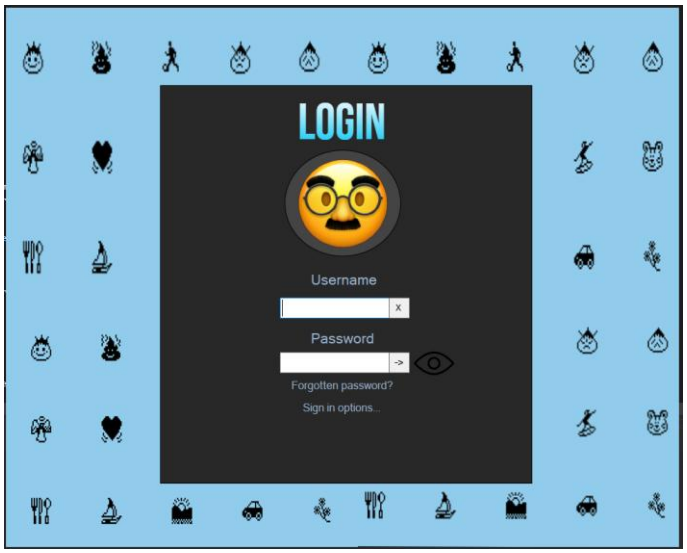


Link Testing Evidence

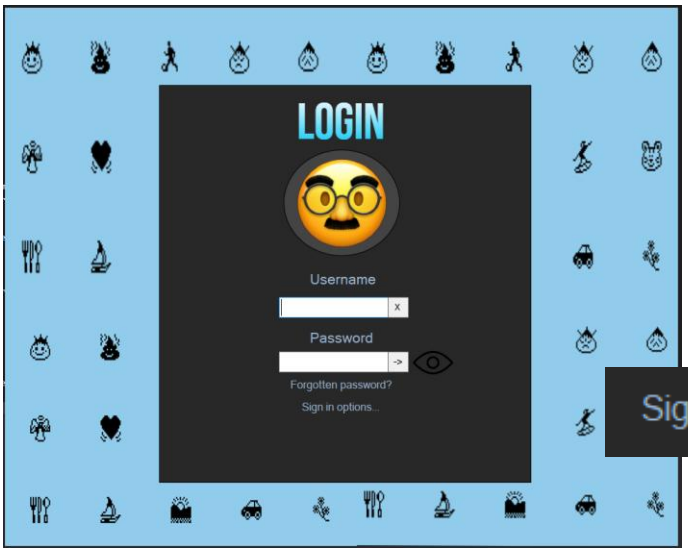
This will include a screenshot of the origin form. the function/control used to get to the destination form and the destination form

Tests 1.1 - 1.2

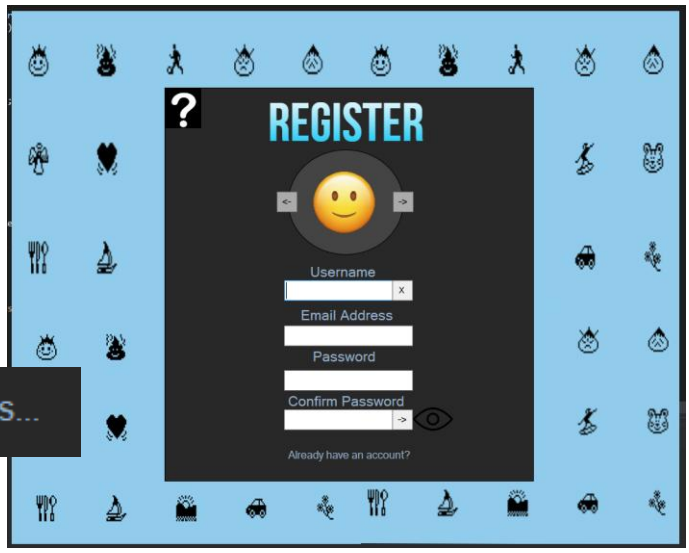
N/A



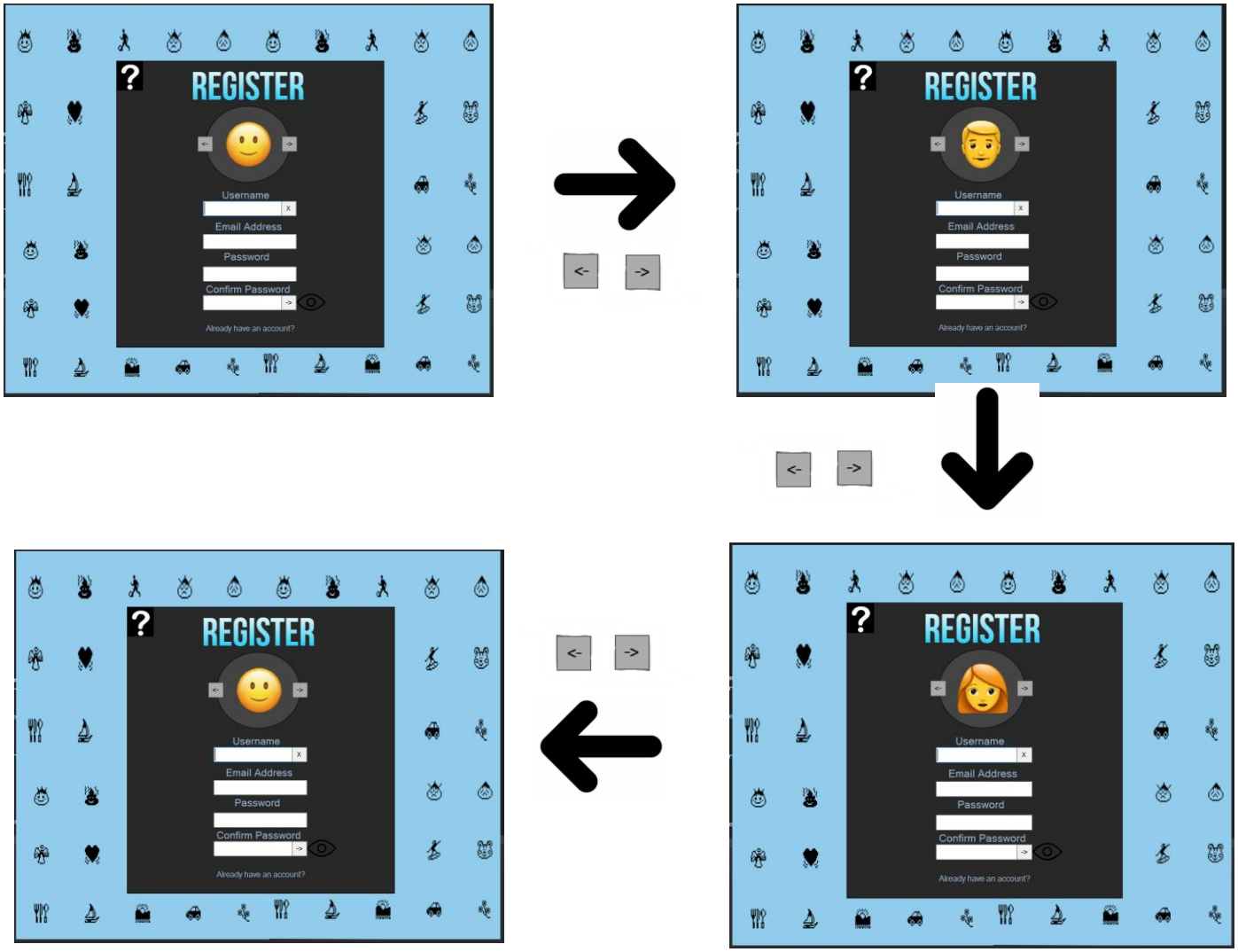
Test 2.2



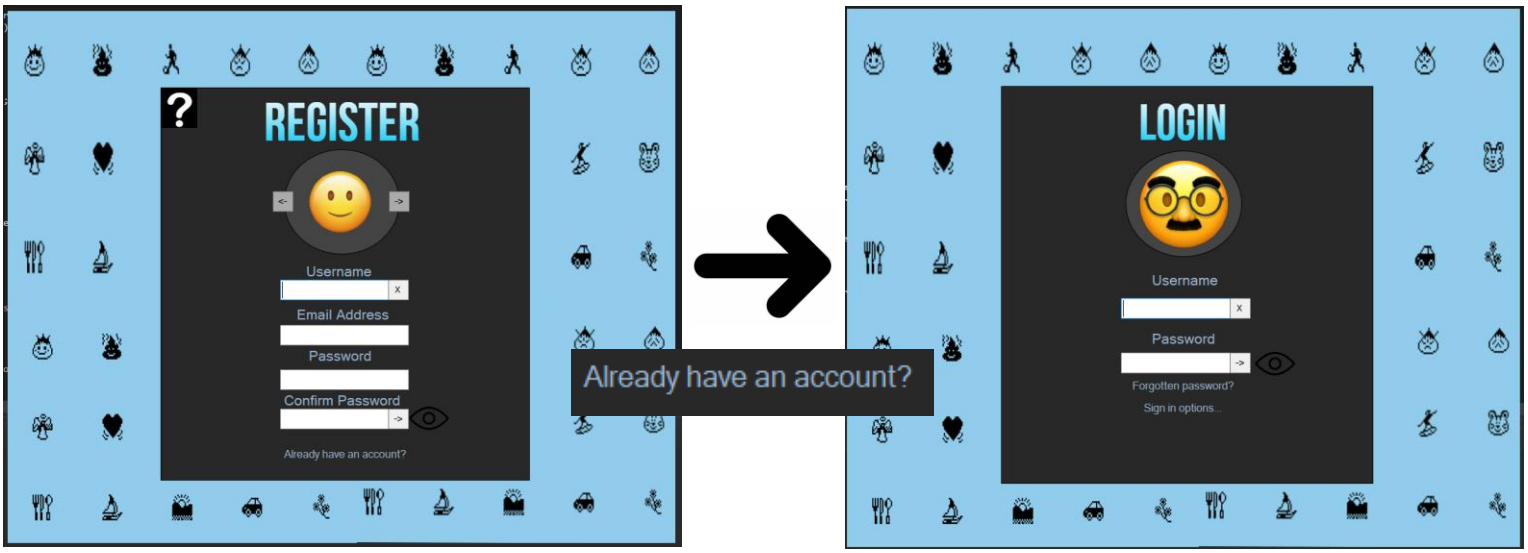
Sign in options...

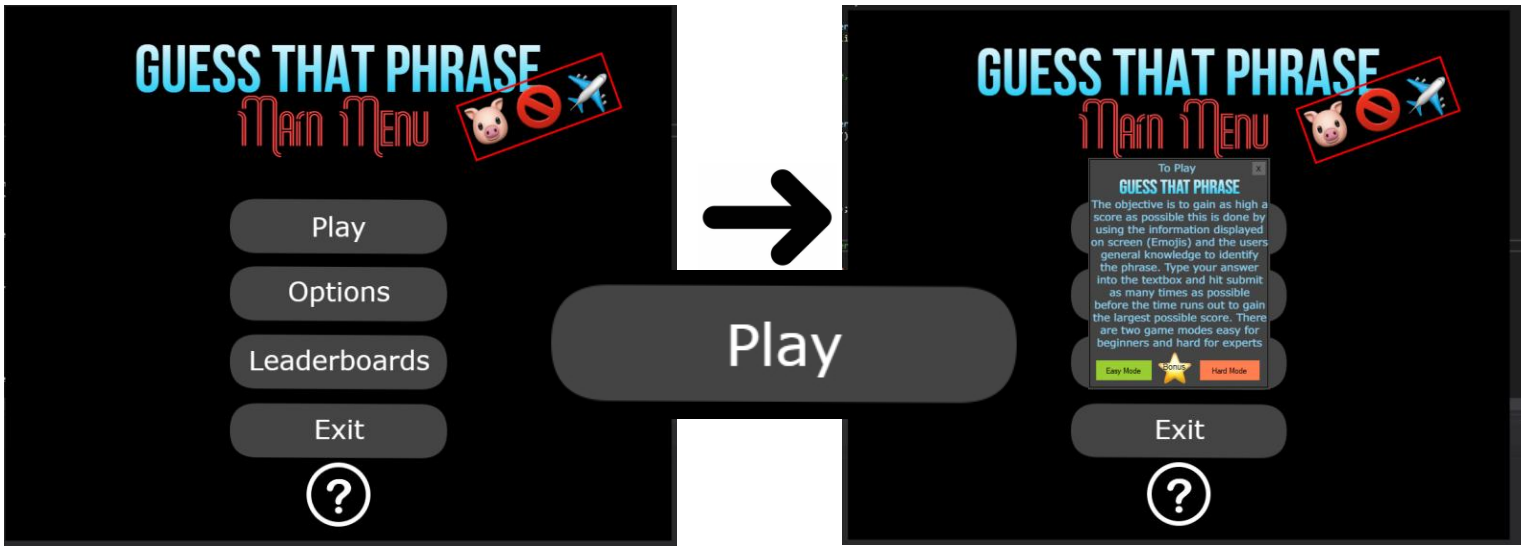


Test 3.4



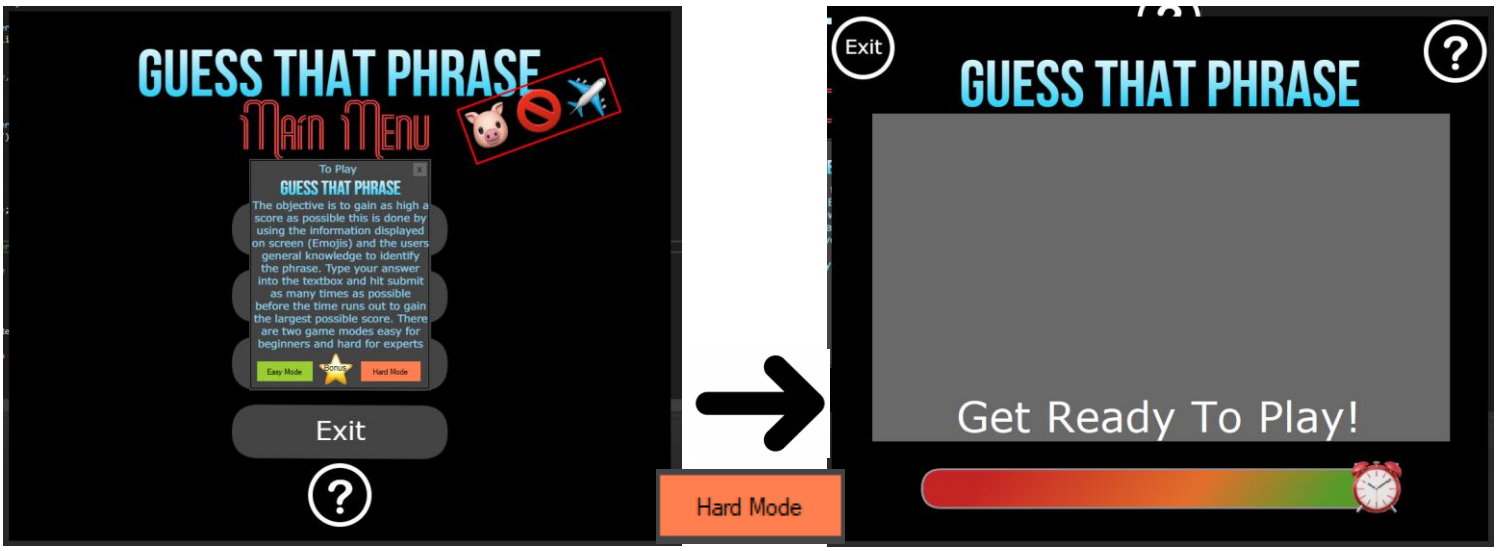
Test 3.5



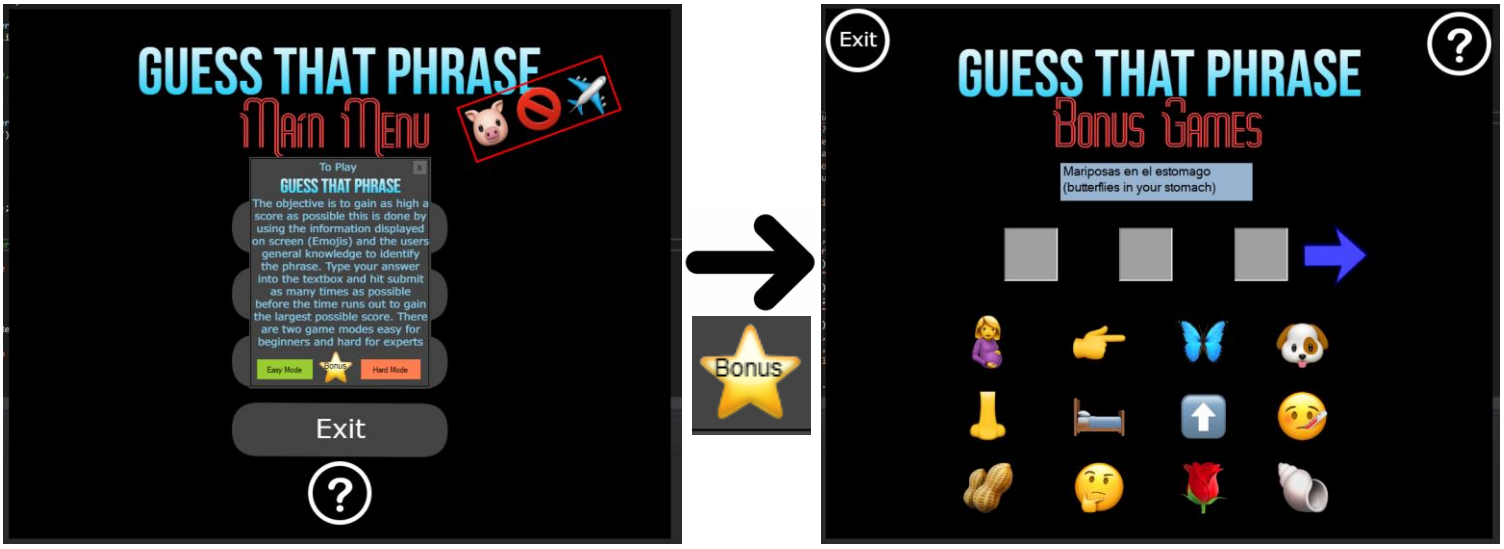


Test 4.2

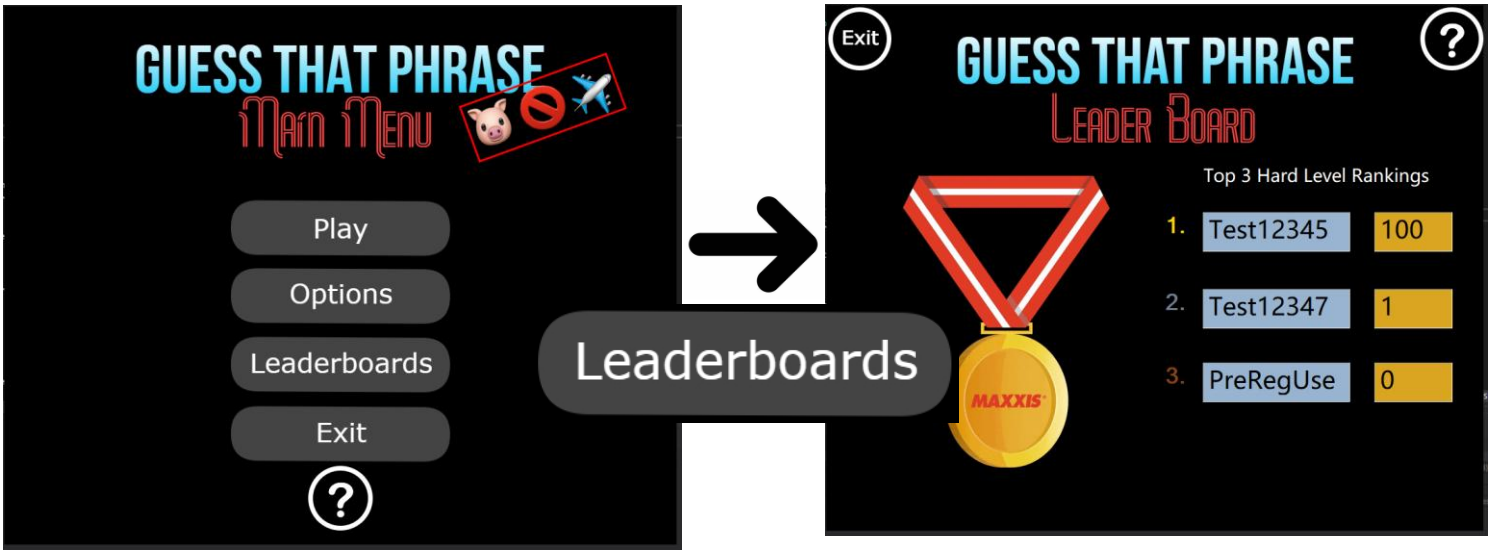
Test 4.3



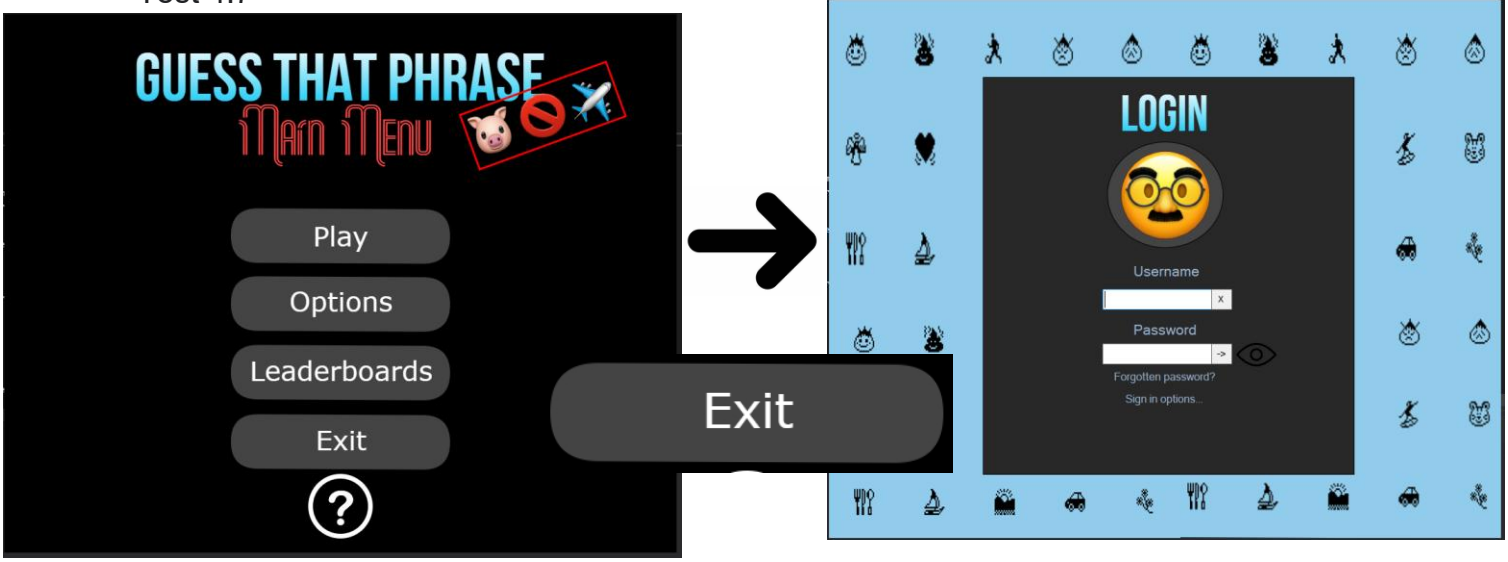
Test 4.4

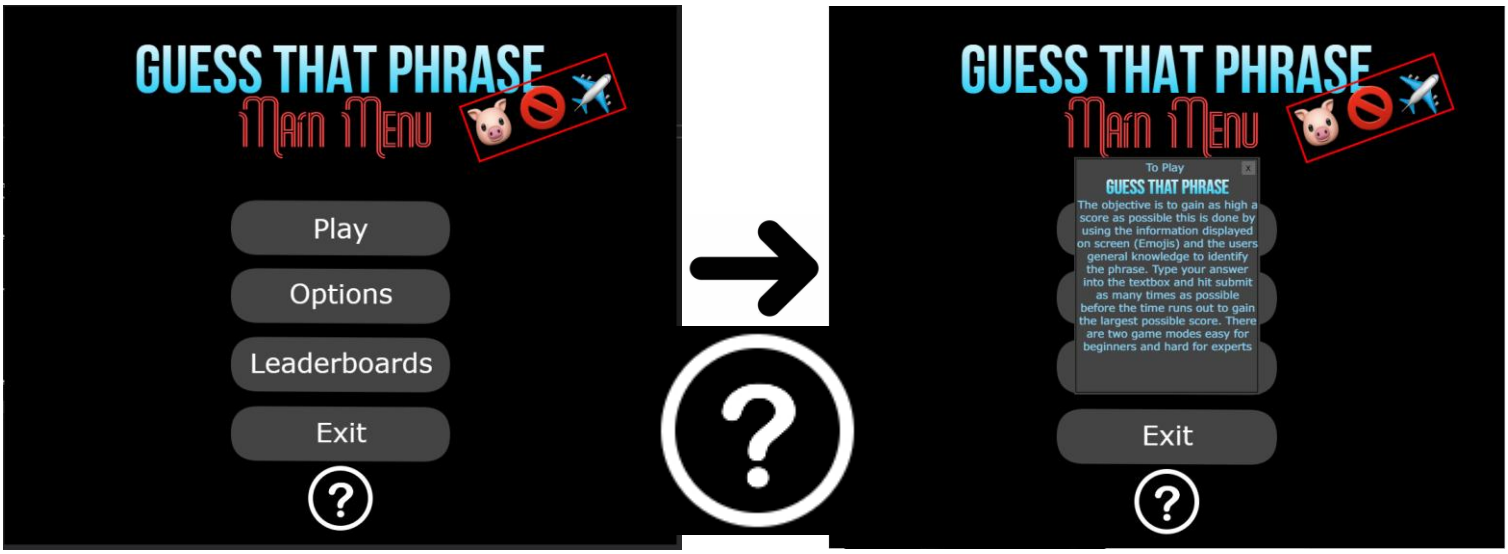


Test 4.5

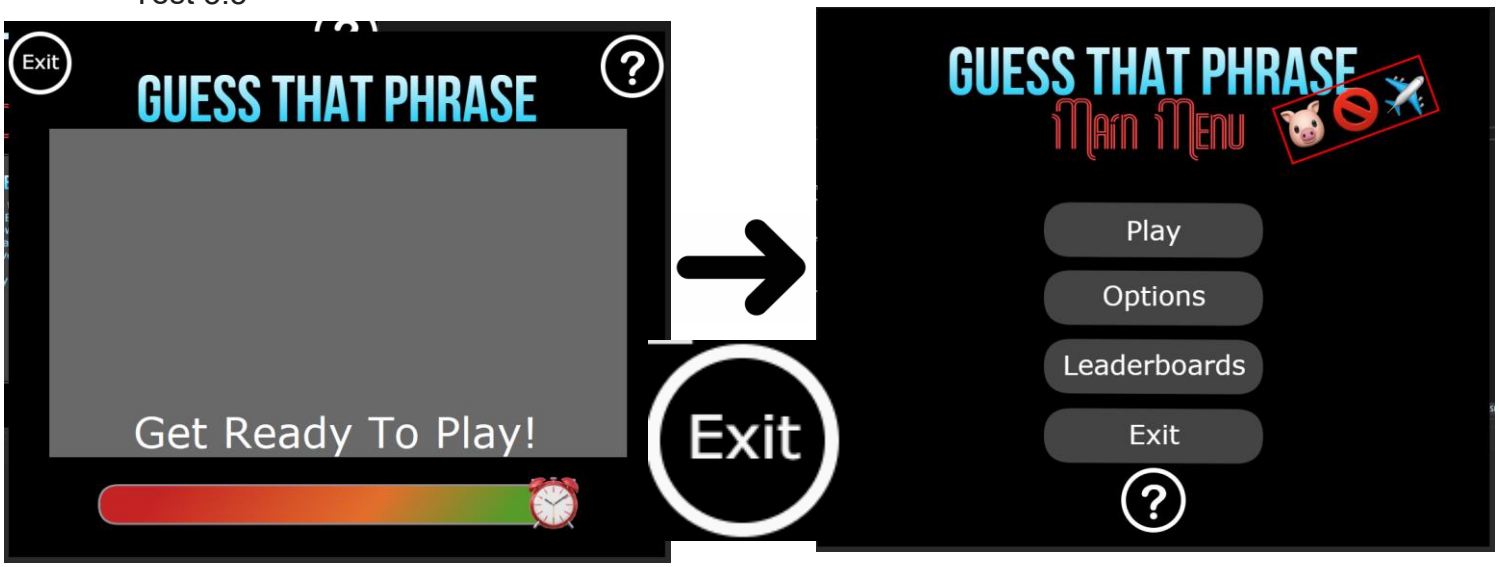


Test 4.7

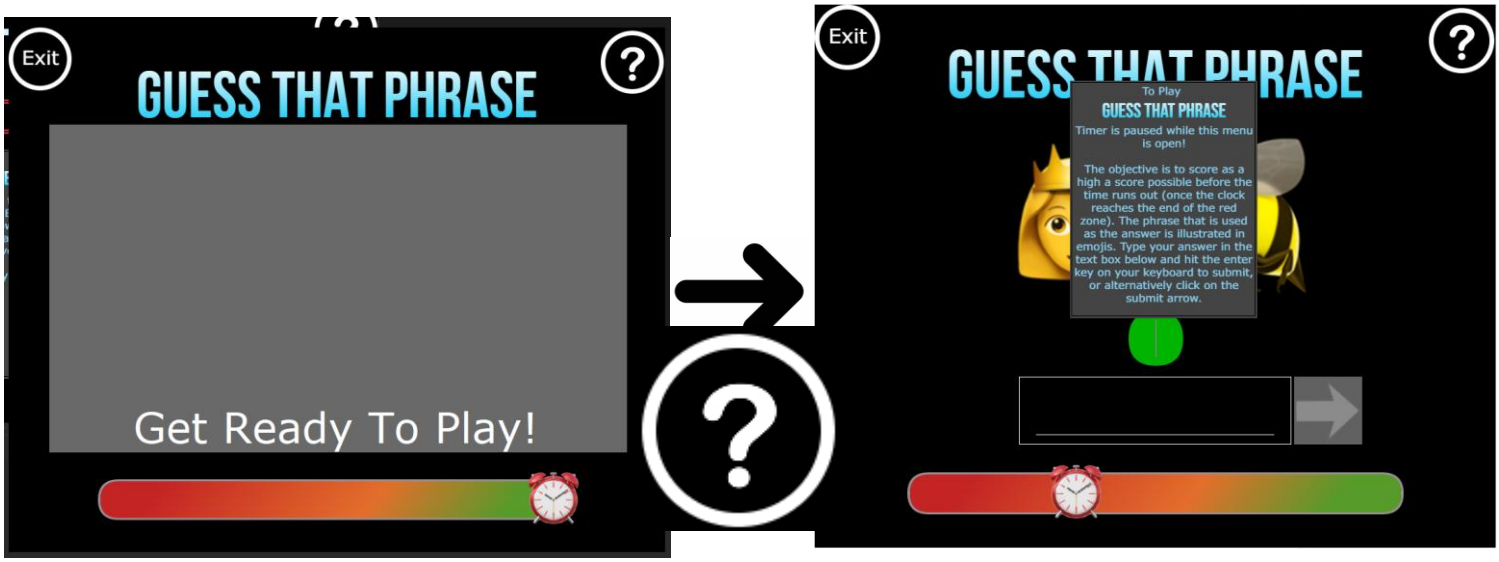




Test 5.5

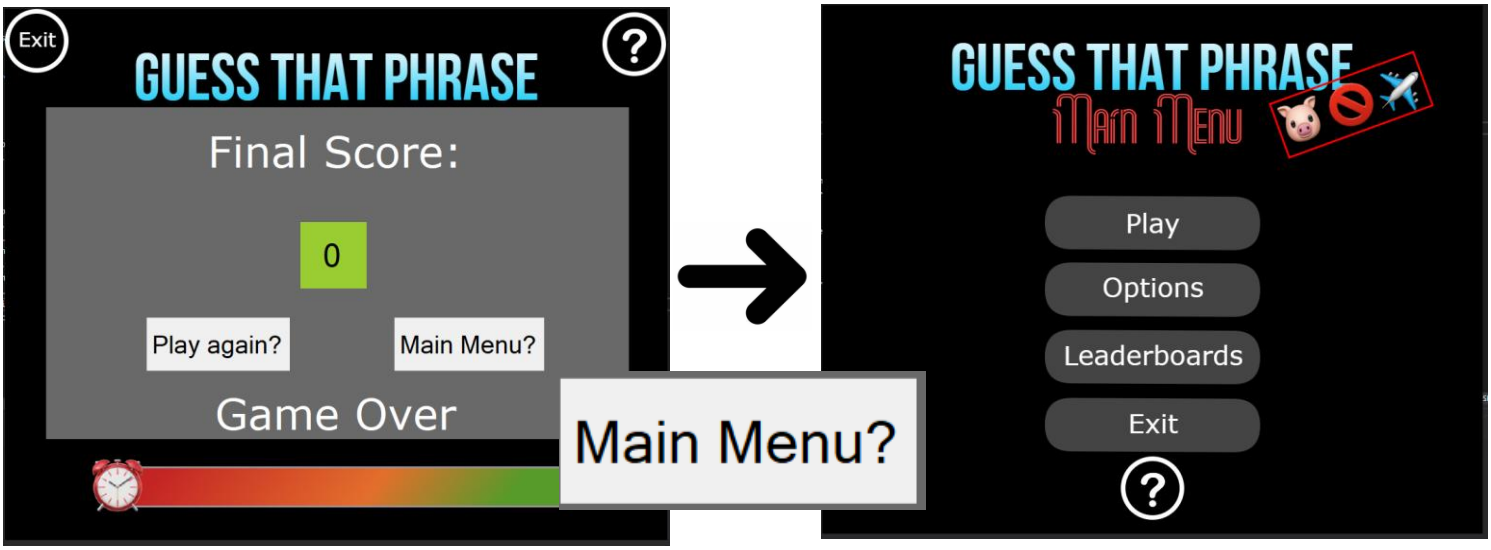


Test 5.6

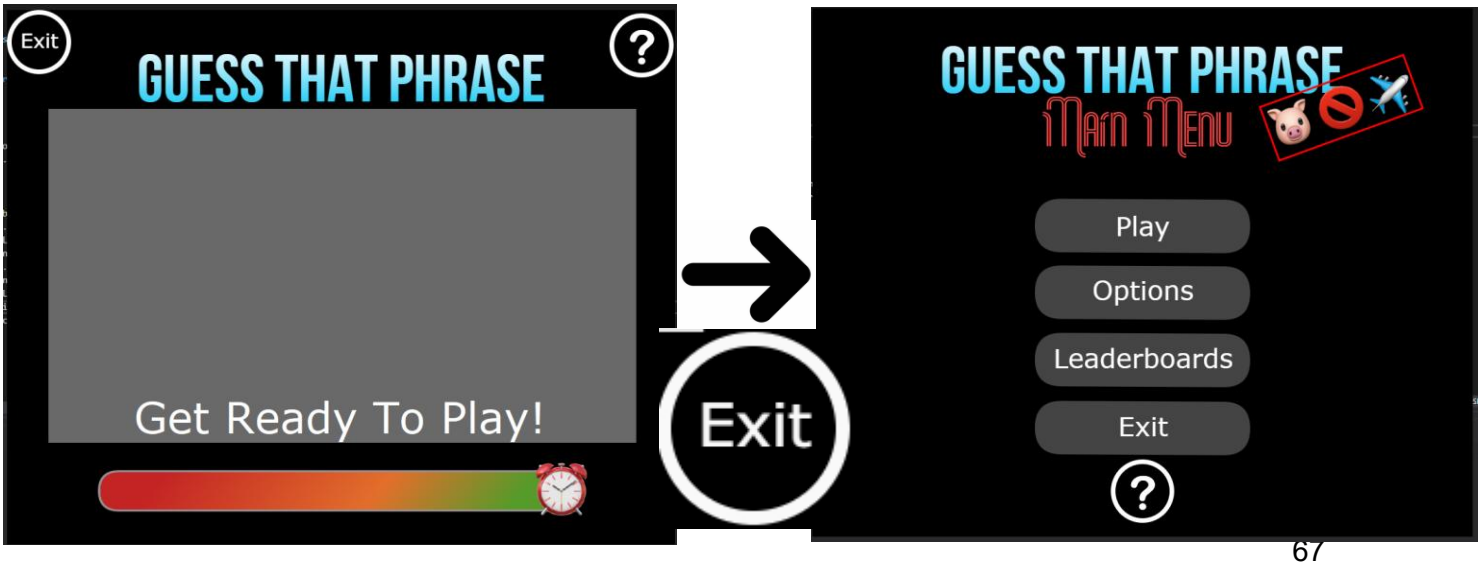


Test 5.8

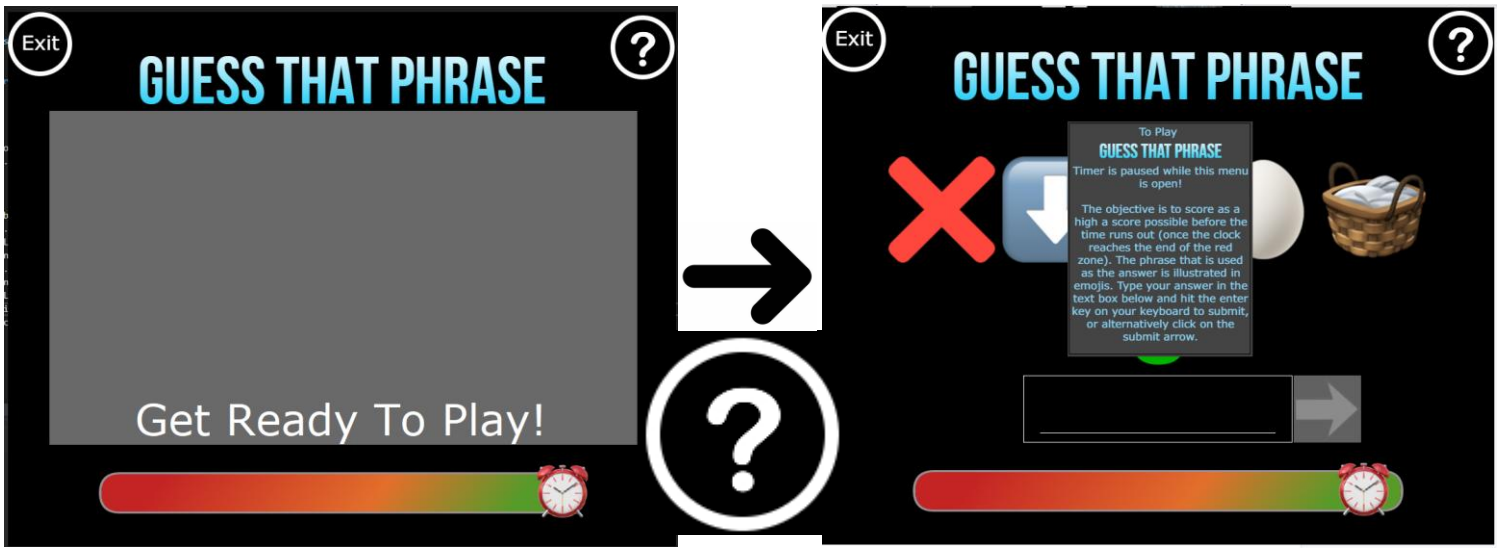
Test 5.9



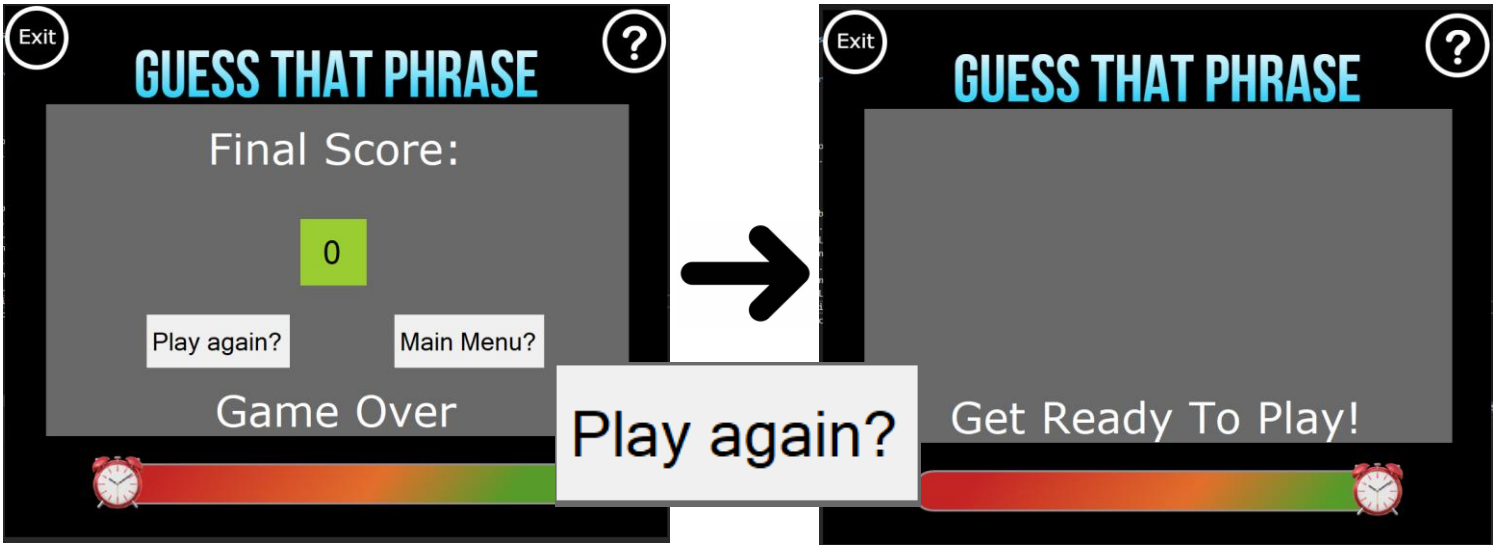
Test 6.5



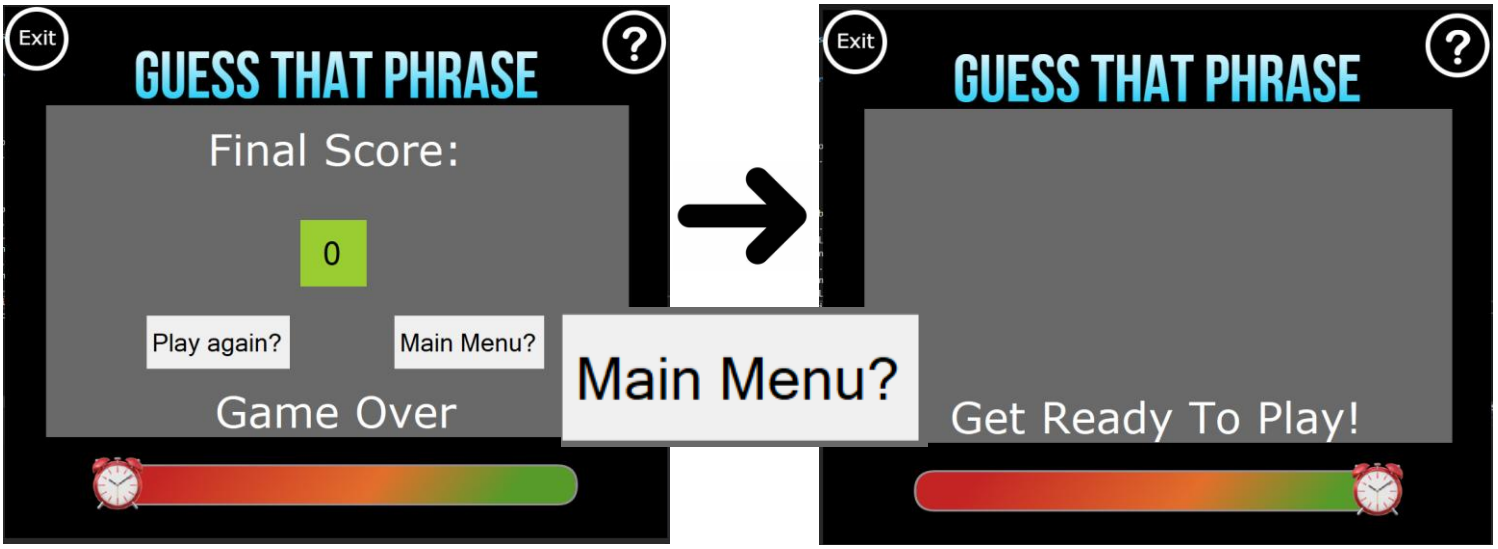
Test 6.6



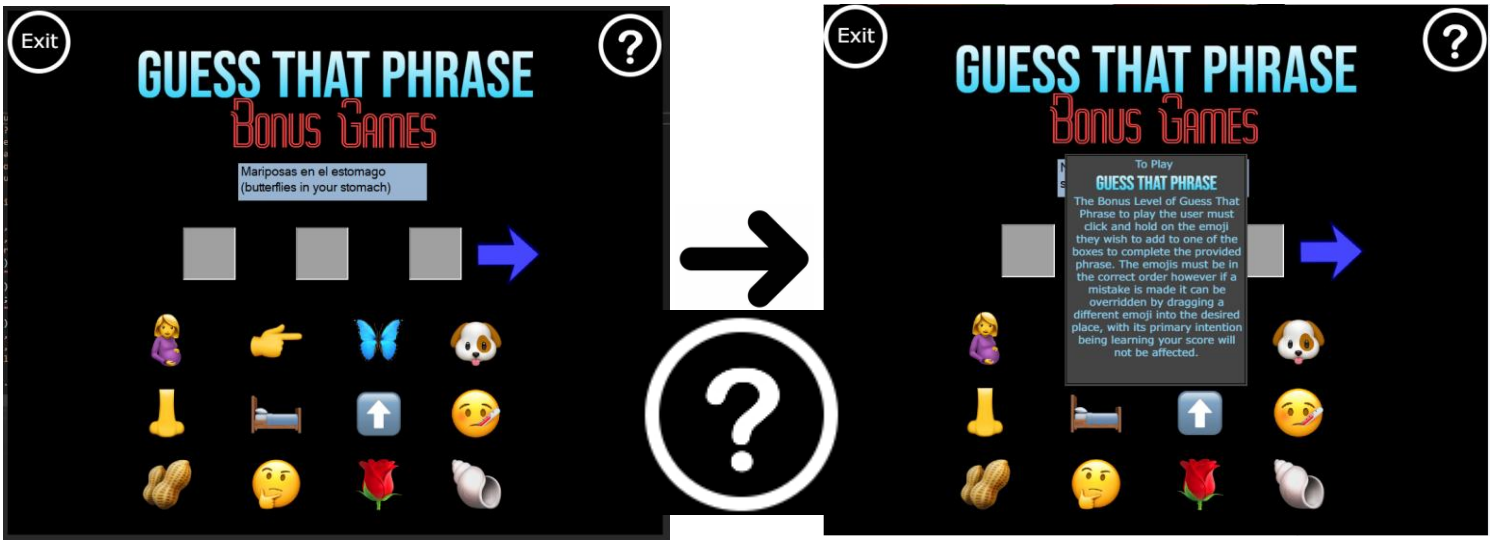
Test 6.8



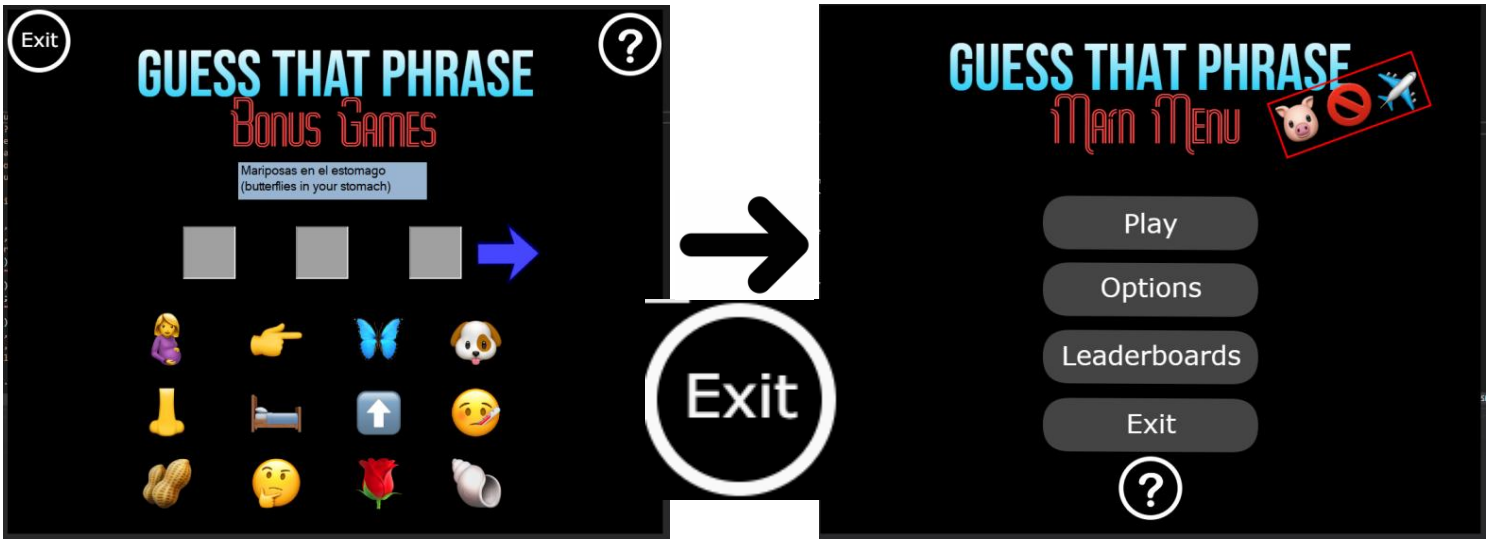
Test 6.9



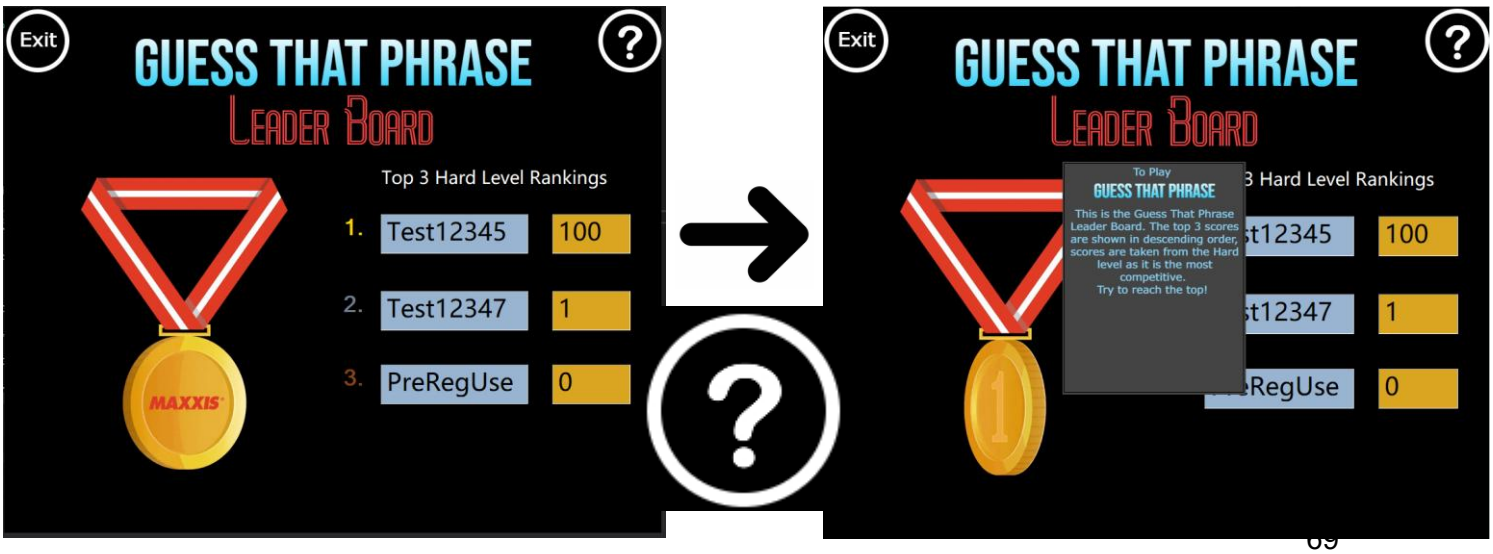
Test 7.5



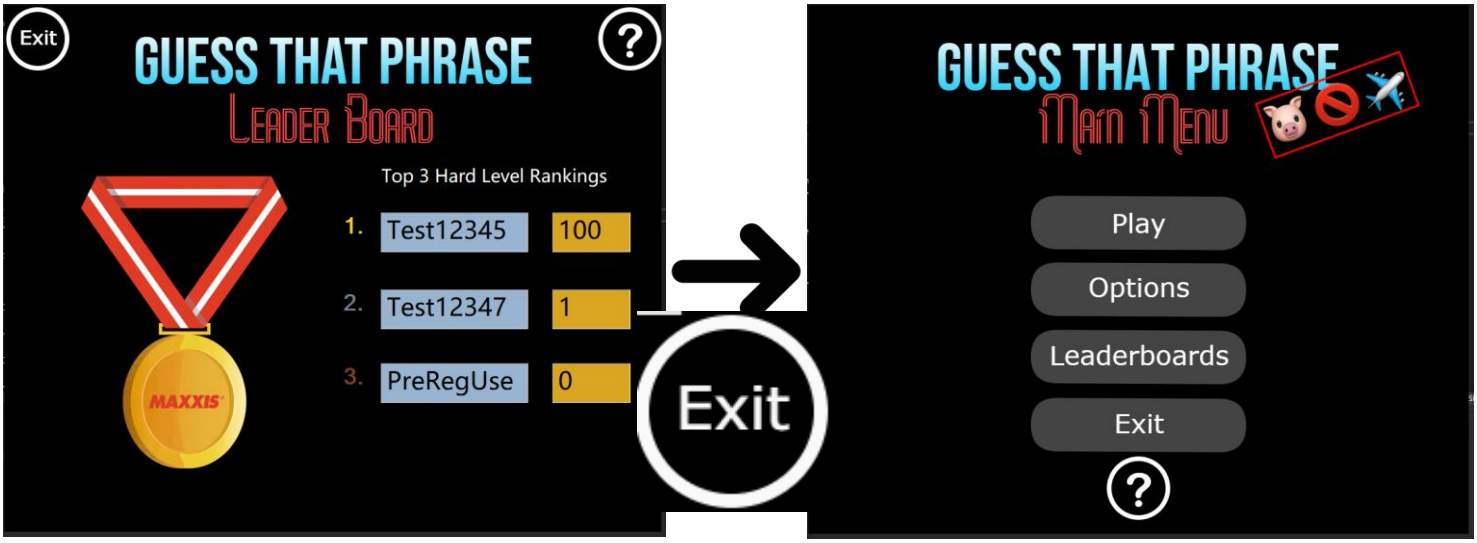
Test 7.6



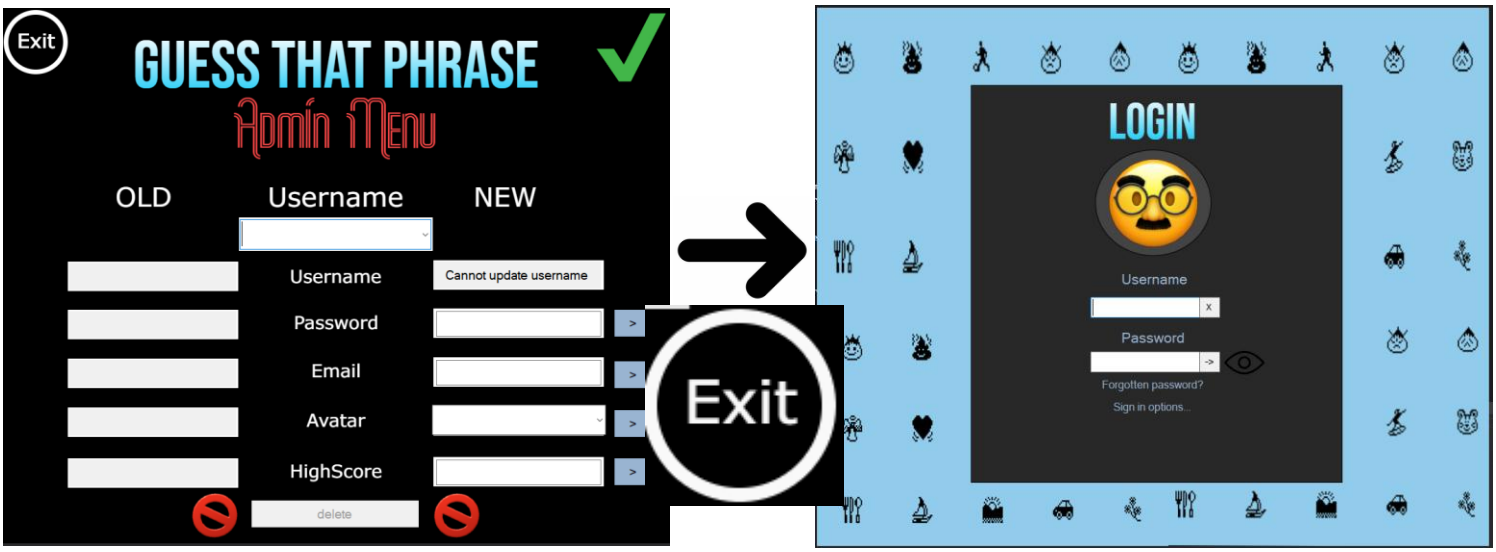
Test 8.2



Test 8.3

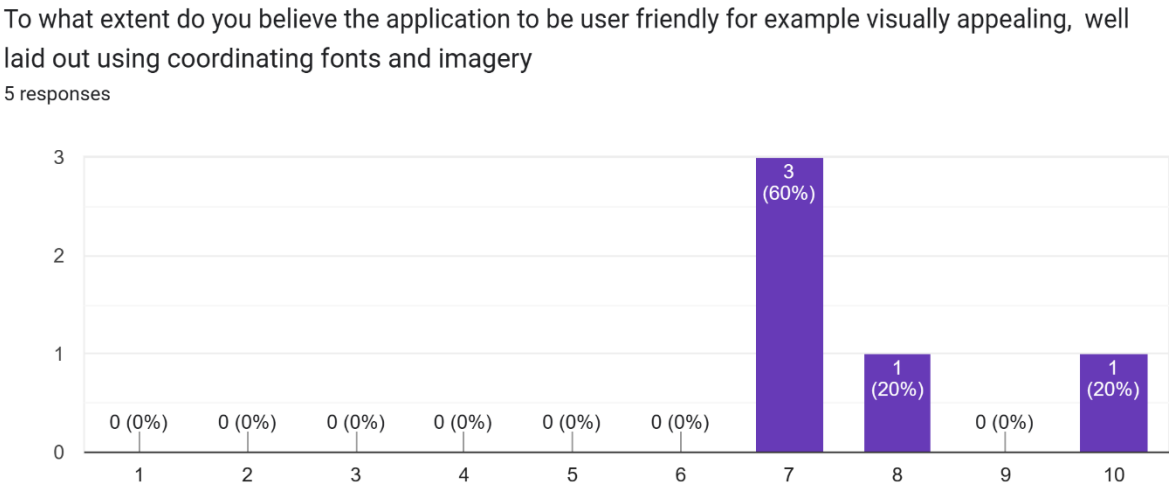


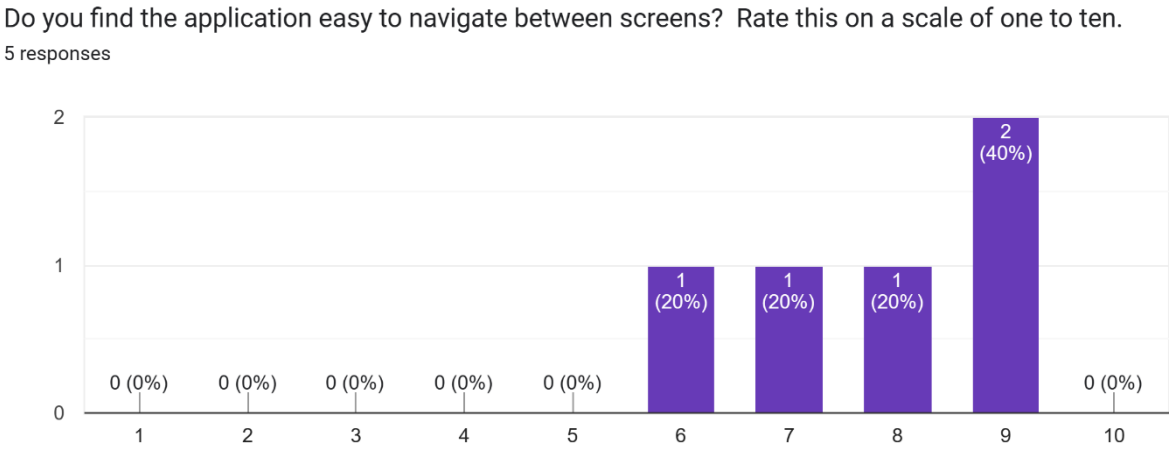
Test 9.5



User Acceptance Testing Data

This segment of testing is based on my user’s thoughts and opinions that were captured through the use of a google form. The evidence is illustrated using the display functions of google forms.



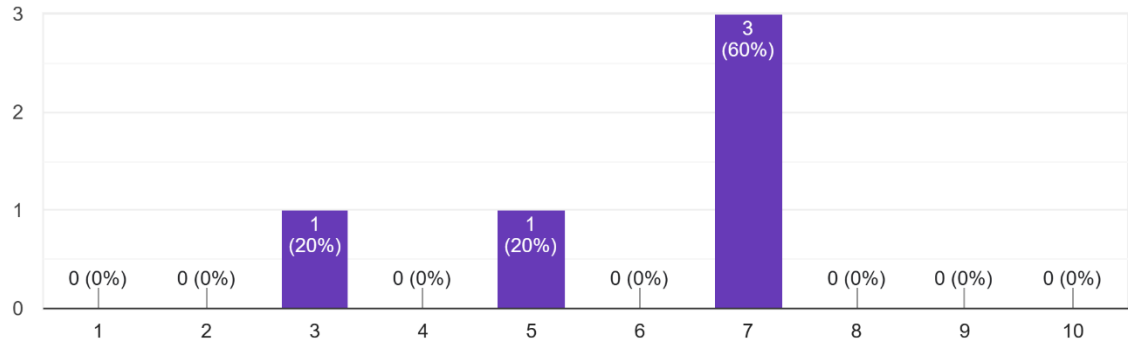


Did you feel the applications questions difficulty was appropriate in each game mode? (5 responses)

Yes but some questions on the hard form were really difficult!
The bonus level questions could have been harder
Some hard level questions were too hard
Some were too easy
I think yes but the easy mode questions were too easy

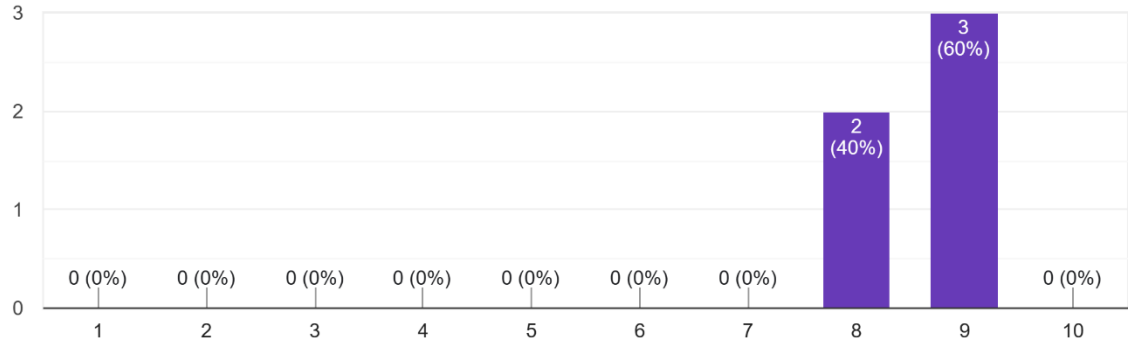
Did you think the time out bar lasted long enough on each screen? Rate this on a scale of one to ten.

5 responses



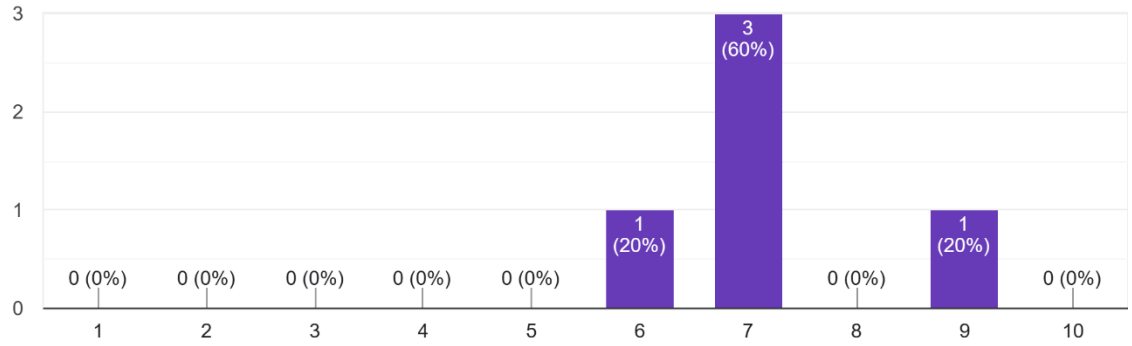
Did you feel the application had enough question variety?

5 responses



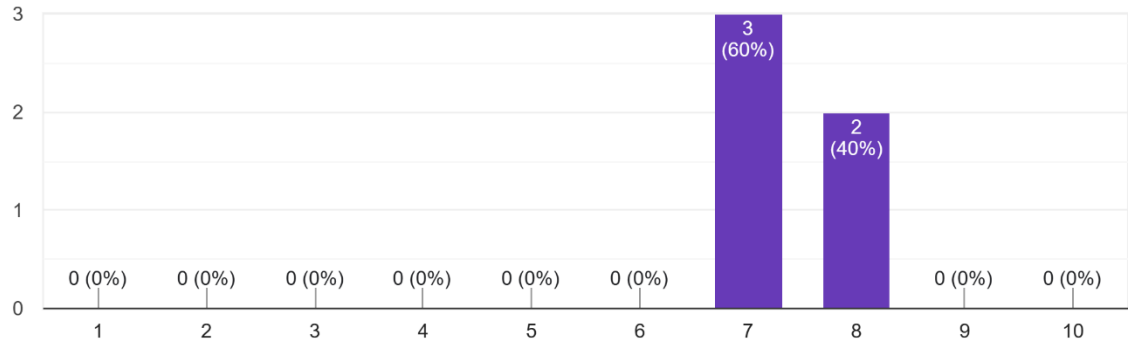
On a scale of 1 to 10 how well was the application optimized to run on school grade PCs

5 responses



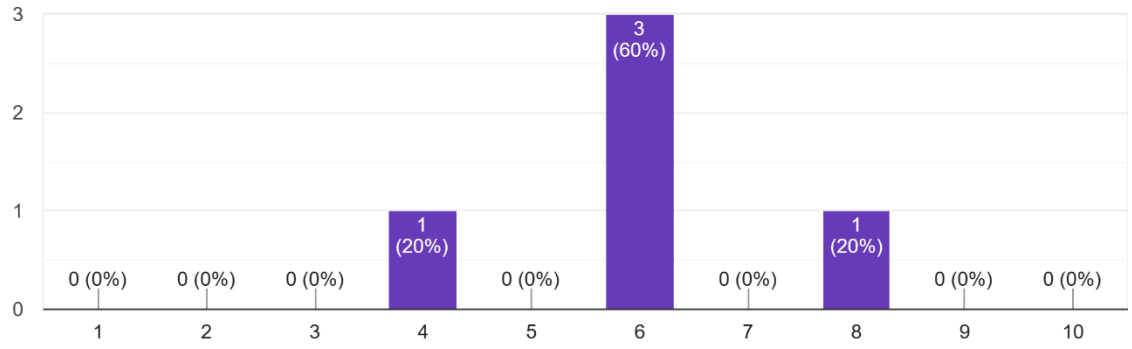
How easy was the registering of your account on this application i.e. creating a strong password and a valid username? Rate this on a scale of one to ten.

5 responses



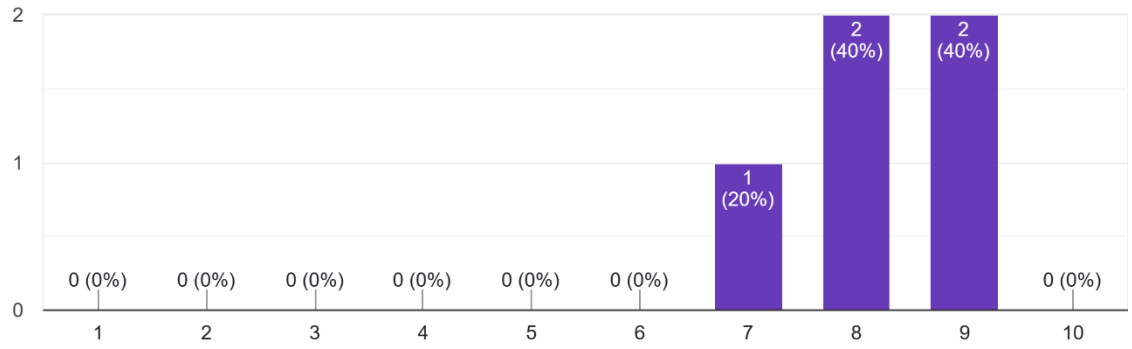
How useful were the information pop ups at each stage of the application? Rate this on a scale of one to ten.

5 responses



Did you feel the application had enough unique question types? Rate this on a scale of one to ten.

5 responses



If you could change one thing about the Guess That Phrase application what would it be and why? (5 responses)

N/A
I would like to be taught more about the emoji history as I read about in the background of the application
I would add more questions to the bonus games mode as I enjoyed it the most
I would add more themes to the question like football and sport
I would make the easy mode phrases longer and make the timers slower

Overall, based on the information provided by the five Users captured above, it is evident that the majority of results where positive. However, it must be noted that the Users found the “Information pop ups”, to be unhelpful and possibly an inconvenience to access the necessary information. This issue may be resolved by reducing the word count of each ‘pop up’ and delivering them more sparsely throughout the application.

It also must be recognised that, the Users felt that they had too little time to complete the questions they were provided with on both levels. With the lowest score recorded in the google form residing in the answer to this question, it is clear that this is a major issue within the application. Thus, a possible response may be to add more game modes such as a medium difficultly section paired with an ‘insane’ difficulty section which would reduce the question count in each portion of the application to twenty five allowing for a greater likelihood of the User gaining satisfaction by completing an entire question set (it should be known that the original design of the application would never have seen this happen in order to test the Users memory), however this may inflate leader board scores.

It should also be recognised where the application succeed, most notably in question variety and the visual appeal of the application (a ten was awarded here). This illustrates that a strong User interest was gained in these sections, which may be matched in the future iterations of the information pop ups, navigation between screens and question types. Which would allow for a greater overall User experience in a future Guess That Phrase application.

Implementation

This section outlines the entirety of the implementation of my application, including the use of external libraries and classes, methods and properties used within each form.

Splashscreen

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {

            timer1.Start();
            this.Hide();
            Welcome w = new Welcome();
            w.Show();
            timer1.Stop();

        }
    }
}
```

Login

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Mail;
```

```
namespace GuessThatWord
{
    public partial class Welcome : Form
    {
        public Welcome()
        {
            InitializeComponent();

            forgottenPasswordSubmitLBL.Visible = false;
            forgottenPasswordTB.Visible = false;
            forgottenPassLbl1.Visible = false;
            label1.Visible = false;
        }

        //Method to allow user to travel to register forn
        private void loginOptionsLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            Register r = new Register();
            r.Show();
        }

        //Method to validate a useres Login and bring them to the Main Menu
        private void submitLoginLBL_Click(object sender, EventArgs e)
        {
            //Take the text the user has entered and assign variables
            string username = uNameLoginTB.Text;
            string password = pWordLoginTB.Text;
            //Create validation and DataBase objects
            Validation v = new Validation(username, password, "", "");
            DataBase db = new DataBase(username, password, "", "", 0);

            //Check to see if the user is an admin
            if (v.validateAdmin())
            {
                //Create a now valid admin object
                Admin a = new Admin(username, password, "");

                //when new admin created go to admin form
                this.Hide();
                AdminPage ap = new AdminPage(a);
                ap.Show();
            }

            //Check to see if the user has an account method
            else if (v.validateLogin())
            {
                var s = string.Join(",", db.readRecord(username, 0, "database.csv"));
                if (s == "Record not found")//String that read record method returns
if false
                {
                    MessageBox.Show("Invalid username or password");
                }
                else
                {
                    var result = s.Split(',');
                    var passFromDB = result[1];
                    //Password check
                    if (password == passFromDB)
                    {
                        MessageBox.Show("Login succesful");
                        this.Hide();
```

```
                        //create new user object read email from database into user
object
                        User u = new User(username, passFromDB, "");
                        MainMenu m = new MainMenu(u);
                        m.Show();
                    }
                    else
                    {
                        //Generalised error message to enhance the platforms security
                        MessageBox.Show("Invalid username or password");
                    }
                }
            }
            else
            {
                //Generalised error message to enhance the platforms security
                MessageBox.Show("Invalid username or password");
            }


        }

        //Method to clear fields on login
        private void clearLoginLBL_Click(object sender, EventArgs e)
        {
            uNameLoginTB.Text ="";
            pWordLoginTB.Text ="";

        }

        //Mehtod to show password
        private void openEyelbl_Click(object sender, EventArgs e)
        {
            //Default password char character
            pWordLoginTB.PasswordChar = '\0';
            closedEyelbl.Visible = true;
            openEyelbl.Visible = false;
        }

        //Mehtod to hide password
        private void closedEyelbl_Click(object sender, EventArgs e)
        {
            openEyelbl.Visible = true;
            closedEyelbl.Visible = false;
            pWordLoginTB.PasswordChar = '*';
        }

        private void loginForgotpassLBL_Click(object sender, EventArgs e)
        {

            forgottenPasswordSubmitLBL.Visible = true;
            forgottenPasswordTB.Visible = true;
            label1.Visible = true;
            forgottenPassLbl1.Visible = true;


        }

        private void forgottenPasswordSubmitLBL_Click(object sender, EventArgs e)
        {
            string userRequest = "";
            if (forgottenPasswordTB.Text != "")
```

```
            {
                userRequest = forgottenPasswordTB.Text;
                try
                {
                    SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

                    //Authentication - ensures the program has a valid email to send
from
                    NetworkCredential cred = new
NetworkCredential("GuessThatPhrase@gmail.com", "boekdvvnrbyspxye");

                    MailMessage msg = new MailMessage();

                    msg.From = new MailAddress("GuessThatPhrase@gmail.com");

                    msg.To.Add("GuessThatPhrase@gmail.com");

                    msg.Subject = "Forgotten Password Notification";

                    msg.Body = "Please update the following user's password: " +
userRequest;

                    client.UseDefaultCredentials = false;

                    client.Credentials = cred;

                    //enabling SSL (Secure sockets layer, encryption) this is required
by most email providers to send mail
                    client.EnableSsl = true;

                    client.Send(msg);

                    MessageBox.Show("Forgotten Password Request Received. You will be
contacted within 24 hours");

                    forgottenPasswordSubmitLBL.Visible = false;
                    forgottenPasswordTB.Visible = false;
                    forgottenPassLbl1.Visible = false;
                    label1.Visible = false;
                }

                catch
                {
                    MessageBox.Show("Something went wrong..");
                }
            }
            else
            {
                MessageBox.Show("Please submit a valid request!");
            }



        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Register

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class Register : Form
    {
        public Register()
        {
            InitializeComponent();
        }

        int timesPressed = 0;
        int avatarCount = 0;
        //Method to bring the user to the Login form
        private void RegisterOptionsLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            Welcome w = new Welcome();
            w.Show();
        }

        private void submitRegisterLBL_Click(object sender, EventArgs e)
        {
            //get current avatar
            string avatar;
            if (avatarCount == 0)
            {
                avatar = "AvatarRegLBL1";
            }
            else if (avatarCount == 1)
            {
                avatar = "AvatarRegLBL2";
            }
            else
            {
                avatar = "AvatarRegLBL3";
            }
            //Validate fields on registration form
            string username = uNameRegisterTB.Text;
            string password = pWordRegisterTB.Text;
            string confirmPassword = pWordRegisterConfirmTB.Text;
            string email = eMailRegisterTB.Text;

            //Create a validation object of the useres credentials check password
strength etc.
            Validation v = new Validation(username, password, confirmPassword, email);

            //If successful add new user in datatbase
            if (v.validateUsername() && v.validatePassword() && v.validateEmail())
            {
                //Create user DataBase object
```

```
            DataBase d = new DataBase(username, password, email, avatar, 0);
            bool success = d.addRecord("");

            //If register is vaild user is brought to login form
            if (success)
            {
                MessageBox.Show("Success, please login");
                this.Hide();
                Welcome w = new Welcome();
                w.Show();
            }
            else //only shows if all other fields on register are valid
            {
                MessageBox.Show("Username is taken");

            }

        }
        //Register error handling with field requirements
        if(pWordRegisterTB.Text != pWordRegisterConfirmTB.Text)
        {
            MessageBox.Show("Passwords do not match");

        }
        if (!v.validateUsername())
        {
            MessageBox.Show("Username does not meet requirements");

        }
        if (!v.validateEmail())
        {
            MessageBox.Show("Email does not meet requirements");

        }
        if (!v.validatePassword())
        {
            MessageBox.Show("Password does not meet requirements");

        }
    }
    //Clear fields method for register form
    private void clearRegisterLBL_Click(object sender, EventArgs e)
    {
        uNameRegisterTB.Text = "";
        eMailRegisterTB.Text = "";
        pWordRegisterTB.Text = "";
        pWordRegisterConfirmTB.Text = "";
    }
    //Make help label visible or not method
    private void QuestionMarkRegisterLBL_Click(object sender, EventArgs e)
    {
        timesPressed++;

        if (timesPressed == 2)
        {
            registerHelpScreenLBL.Visible = false;
            timesPressed = 0;
        }
        else
        {
            registerHelpScreenLBL.Visible = true;
            registerHelpScreenLBL.BringToFront();
```

```
    }
}
//Avatar scroll right method
private void LBLScrollRight_Click(object sender, EventArgs e)
{
    avatarCount++;
    if (avatarCount == 0)
    {
        AvatarRegLBL1.Hide();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Show();
    }
    else if (avatarCount == 1)
    {
        AvatarRegLBL1.Hide();
        AvatarRegLBL2.Show();
        AvatarRegLBL3.Hide();
    }
    else if (avatarCount == 2)
    {
        AvatarRegLBL1.Show();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Hide();
    }
    else if (avatarCount == 3)
    {
        AvatarRegLBL1.Hide();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Show();
        avatarCount = 0;
    }

}
//Avatar scroll left method
private void LBLScrollLeft_Click(object sender, EventArgs e)
{
    avatarCount--;
    if (avatarCount == 0)
    {
        AvatarRegLBL1.Hide();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Show();
    }
    else if (avatarCount == 1)
    {
        AvatarRegLBL1.Hide();
        AvatarRegLBL2.Show();
        AvatarRegLBL3.Hide();
    }
    else if (avatarCount == 2)
    {
        AvatarRegLBL1.Show();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Hide();
    }
    else if (avatarCount == -1)
    {
        AvatarRegLBL1.Show();
        AvatarRegLBL2.Hide();
        AvatarRegLBL3.Hide();
        avatarCount = 2;
    }
```

```
        }
        //Show password and comfirm password fields
        private void openEyelbl_Click(object sender, EventArgs e)
        {
            //Default password char character
            pWordRegisterConfirmTB.PasswordChar = '\0';
            pWordRegisterTB.PasswordChar = '\0';
            closedEyelbl.Visible = true;
            openEyelbl.Visible = false;
        }

        //Hide password and comfirm password fields
        private void closedEyelbl_Click(object sender, EventArgs e)
        {
            openEyelbl.Visible = true;
            closedEyelbl.Visible = false;
            pWordRegisterConfirmTB.PasswordChar = '*';
            pWordRegisterTB.PasswordChar = '*';
        }
    }
}
```

Main Menu

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class MainMenu : Form
    {

        User currentUser = null;
        public MainMenu(User u)
        {
            InitializeComponent();
            currentUser = u;
            gif1.Visible = false;
            gif2.Visible = false;
            optionsLbl.Visible = false;
            optionsCheckbox.Visible = false;
            label1.Visible = false;
        }
        //exit the Main Menu to the login
        private void MainMenuExitLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            Welcome ww = new Welcome();
            ww.Show();
        }
        //bring up the play label
        private void MainMenuPlayLBL_Click(object sender, EventArgs e)
        {
```

```
        MainMenuGameInfoLBL.Visible = true;
        MainMenuGameInfoCloseLBL.Visible = true;
        MainMenuGameInfoCloseLBL.BringToFront();
        MainMenuGameInfoEasyLBL.Visible = true;
        MainMenuGameInfoEasyLBL.BringToFront();
        MainMenuGameInfoHardLBL.Visible = true;
        MainMenuGameInfoHardLBL.BringToFront();
        BonusGamesLBL.Visible = true;
        BonusGamesLBL.BringToFront();

    }
    //close the play label
    private void MainMenuGameInfoCloseLBL_Click(object sender, EventArgs e)
    {
        MainMenuGameInfoLBL.Visible = false;
        MainMenuGameInfoCloseLBL.Visible = false;
        MainMenuGameInfoEasyLBL.Visible = false;
        MainMenuGameInfoHardLBL.Visible = false;
        BonusGamesLBL.Visible = false;
    }
    //bring user to game easy
    private void MainMenuGameInfoEasyLBL_Click(object sender, EventArgs e)
    {
        this.Hide();
        Game g = new Game(currentUser);
        g.Show();
    }
    //bring user to game hard
    private void MainMenuGameInfoHardLBL_Click(object sender, EventArgs e)
    {
        this.Hide();
        GameHard gh = new GameHard(currentUser);
        gh.Show();
    }
    //bring up Main Menu info label
    private void MainMenuHelpLBLQMark_Click(object sender, EventArgs e)
    {
        MainMenuGameInfoLBL.Visible = true;
        MainMenuGameInfoCloseLBL.Visible = true;
        MainMenuGameInfoCloseLBL.BringToFront();
    }

    //bring user to bonus games
    private void BonusGamesLBL_Click(object sender, EventArgs e)
    {
        this.Hide();
        BonusGames bg = new BonusGames(currentUser);
        bg.Show();
    }
    //bring user to leader board page
    private void MainMenuLeaderboardsLBL_Click(object sender, EventArgs e)
    {
        this.Hide();
        LeaderBoard lb = new LeaderBoard();
        lb.Show();
    }

    private void MainMenuOptionsLBL_Click(object sender, EventArgs e)
    {
        optionsCheckbox.Visible = true;
        label1.Visible = true;
        optionsLbl.Visible = true;
```

```
        }

        private void optionsCheckbox_CheckedChanged(object sender, EventArgs e)
        {
            if(gif1.Visible == true)
            {
                gif1.Visible = false;
                gif2.Visible = false;
            }
            else
            {
                gif1.Visible = true;
                gif2.Visible = true;
            }

        }

        private void label1_Click(object sender, EventArgs e)
        {
            optionsLbl.Visible = false;
            optionsCheckbox.Visible = false;
            label1.Visible = false;
        }
    }
}
```

Leader board

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class LeaderBoard : Form
    {

        User currentUser = null;
        int timesClicked = 0;

        public LeaderBoard()
        {
            InitializeComponent();
        }

        private void LeaderBoard_Load(object sender, EventArgs e)
        {
            //create a new database object  to bring leaderboard info back username
and score
            DataBase db = new DataBase();
            List<User> highestScoringUsers = db.getLeaderboardInfo();

            //put the names and higscores in descending order if they exist
            if (highestScoringUsers.Count() ==0)
```

```
            {

            }
            else if(highestScoringUsers.Count() ==1){
                FirstPlaceLblName.Text =
highestScoringUsers.ElementAt(0).getUsername();
                FirstPlaceLblScore.Text =
highestScoringUsers.ElementAt(0).getHighScore().ToString();
            }
            else if(highestScoringUsers.Count() == 2){
                FirstPlaceLblName.Text =
highestScoringUsers.ElementAt(0).getUsername();
                FirstPlaceLblScore.Text =
highestScoringUsers.ElementAt(0).getHighScore().ToString();
                SecondPlaceLblName.Text =
highestScoringUsers.ElementAt(1).getUsername();
                SecondPlaceLblScore.Text =
highestScoringUsers.ElementAt(1).getHighScore().ToString();
            }
            else if (highestScoringUsers.Count() == 3)
            {
                FirstPlaceLblName.Text =
highestScoringUsers.ElementAt(0).getUsername();
                FirstPlaceLblScore.Text =
highestScoringUsers.ElementAt(0).getHighScore().ToString();
                SecondPlaceLblName.Text =
highestScoringUsers.ElementAt(1).getUsername();
                SecondPlaceLblScore.Text =
highestScoringUsers.ElementAt(1).getHighScore().ToString();
                ThirdPlaceLblName.Text =
highestScoringUsers.ElementAt(2).getUsername();
                ThirdPlaceLblScore.Text =
highestScoringUsers.ElementAt(2).getHighScore().ToString();
            }



        }
        //bring the user back to the main menu method
        private void LeaderBoardExitlbl_Click(object sender, EventArgs e)
        {
            this.Hide();
            MainMenu m = new MainMenu(currentUser);
            m.Show();
        }
        //leaderboard info label display
        private void LeaderBoardHelplbl_Click(object sender, EventArgs e)
        {
            timesClicked++;
            LeaderBoardInfoLBL.Visible = true;
            if (timesClicked == 2)
            {
                LeaderBoardInfoLBL.Visible = false;
                timesClicked = 0;
            }
        }
    }
}
```

Play Game (Easy Level)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class Game : Form
    {
        Dictionary<Label, string> easyPhrases = new Dictionary<Label, string>();
        int currentScore = 0;
        int wait = 0;
        User currentUser = null;
        int timesClicked = 0;



        public Game(User u)
        {
            InitializeComponent();
            currentUser = u;
            //question dictionary with labels and answers
            easyPhrases.Add(GameEasyQLBL1, "anchorman");
            easyPhrases.Add(GameEasyQLBL2, "applewatch");
            easyPhrases.Add(GameEasyQLBL3, "armchair");
            easyPhrases.Add(GameEasyQLBL4, "bananabread");
            easyPhrases.Add(GameEasyQLBL5, "bananaboat");
            easyPhrases.Add(GameEasyQLBL6, "bedbug");
            easyPhrases.Add(GameEasyQLBL7, "bluemoon");
            easyPhrases.Add(GameEasyQLBL8, "bombshell");
            easyPhrases.Add(GameEasyQLBL9, "bookclub");
            easyPhrases.Add(GameEasyQLBL10, "bookworm");
            easyPhrases.Add(GameEasyQLBL11, "burgerking");
            easyPhrases.Add(GameEasyQLBL12, "captainamerica");
            easyPhrases.Add(GameEasyQLBL13, "catnap");
            easyPhrases.Add(GameEasyQLBL14, "chillpill");
            easyPhrases.Add(GameEasyQLBL15, "couchpotato");
            easyPhrases.Add(GameEasyQLBL16, "crywolf");
            easyPhrases.Add(GameEasyQLBL17, "dancingqueen");
            easyPhrases.Add(GameEasyQLBL18, "dinnerparty");
            easyPhrases.Add(GameEasyQLBL19, "familytree");
            easyPhrases.Add(GameEasyQLBL20, "firetruck");
            easyPhrases.Add(GameEasyQLBL21, "flagship");
            easyPhrases.Add(GameEasyQLBL22, "giftcard");
            easyPhrases.Add(GameEasyQLBL23, "homesick");
            easyPhrases.Add(GameEasyQLBL24, "honeymoon");
            easyPhrases.Add(GameEasyQLBL25, "hotchocolate");
            easyPhrases.Add(GameEasyQLBL26, "hotmeal");
            easyPhrases.Add(GameEasyQLBL27, "iphone");
            easyPhrases.Add(GameEasyQLBL28, "jinglebellrock");
            easyPhrases.Add(GameEasyQLBL29, "kingkong");
            easyPhrases.Add(GameEasyQLBL30, "lighthouse");
            easyPhrases.Add(GameEasyQLBL31, "lockup");
```

```
    easyPhrases.Add(GameEasyQLBL32, "lovebirds");
    easyPhrases.Add(GameEasyQLBL33, "meatloaf");
    easyPhrases.Add(GameEasyQLBL34, "moneytalks");
    easyPhrases.Add(GameEasyQLBL35, "moviestar");
    easyPhrases.Add(GameEasyQLBL36, "ninjaturtle");
    easyPhrases.Add(GameEasyQLBL37, "paycheque");
    easyPhrases.Add(GameEasyQLBL38, "pieceofcake");
    easyPhrases.Add(GameEasyQLBL39, "piggybank");
    easyPhrases.Add(GameEasyQLBL40, "punchbag");
    easyPhrases.Add(GameEasyQLBL41, "queenbee");
    easyPhrases.Add(GameEasyQLBL42, "rocktheboat");
    easyPhrases.Add(GameEasyQLBL43, "sleepwalking");
    easyPhrases.Add(GameEasyQLBL44, "smartcookie");
    easyPhrases.Add(GameEasyQLBL45, "starfish");
    easyPhrases.Add(GameEasyQLBL46, "sunglasses");
    easyPhrases.Add(GameEasyQLBL47, "surfsup");
    easyPhrases.Add(GameEasyQLBL48, "timeflies");
    easyPhrases.Add(GameEasyQLBL49, "twobirdsonestone");
    easyPhrases.Add(GameEasyQLBL50, "footballworldcup");

    //hide every label
    for (int i=0; i< easyPhrases.Count; i++)
    {
        easyPhrases.ElementAt(i).Key.Visible = false;
    }
    //randomise first question selection
    Random ra = new Random();
    int randQ = ra.Next(0, 50);
    easyPhrases.ElementAt(randQ).Key.Visible = true;
    SubmitRightGameEasyLBL.Visible = false;
    SubmitWrongGameEasyLBL.Visible = false;

    //begin game start sequence
    timer3.Tick += new EventHandler(timer3_Tick);
    timer3.Enabled = true;
    timer3.Start();

    GameEasyTrafficLight1.BringToFront();
    GameEasyTrafficLight2.BringToFront();
    GameEasyTrafficLight3.BringToFront();
    GameEasyTrafficLight4.BringToFront();
}

private void timer3_Tick(object sender, EventArgs e)//game starting sequence
{
    GameEasyTrafficLight1.BringToFront();
    GameEasyTrafficLight2.BringToFront();
    GameEasyTrafficLight3.BringToFront();
    GameEasyTrafficLight4.BringToFront();
    GameEasyScoreDisplayTB.SendToBack();
    label1.Enabled = false;



    wait++;
    if (wait == 1)
    {
        GameEasyScoreDisplayTB.Visible = false;
        GameEasyTrafficLight1.Visible = true;
    }
    else if (wait == 2)
    {
```

```
                GameEasyTrafficLight1.Visible = false;
                GameEasyTrafficLight2.Visible = true;
            }
            else if (wait == 3)
            {
                GameEasyTrafficLight1.Visible = false;
                GameEasyTrafficLight2.Visible = false;
                GameEasyTrafficLight3.Visible = true;
            }
            else if (wait == 4)
            {
                GameEasyTrafficLight1.Visible = false;
                GameEasyTrafficLight2.Visible = false;
                GameEasyTrafficLight3.Visible = false;
                GameEasyTrafficLight4.Visible = true;

            }
            else if (wait == 5)
            {
                GameEasyTrafficLight4.Visible = false;
                GameEasyReadyLBL.Visible = false;
                GameEasyScoreDisplayTB.BringToFront();

                GameEasyScoreDisplayTB.Visible = true;

                timer3.Stop();
                label1.Enabled = true;


                timer2.Tick += new EventHandler(timer2_Tick);
                timer2.Enabled = true;
                timer2.Start();
            }
        }

        private void timer2_Tick(object sender, EventArgs e) //clock moving across the
screen timer sequence
        {

            if (GameInfoLBL.Visible == true)
            {
                timer2.Stop();
            }
            //create an int that continually takes away from the clocks x position
            int step = -1;
            this.GameEasyClockLBL.Location = new
Point(this.GameEasyClockLBL.Location.X + step, this.GameEasyClockLBL.Location.Y);
            if(GameEasyClockLBL.Location.X == 130)
            {
                int finalHighScore = 0;
                timer2.Stop();
                GameEasyGameOverLBL.Visible = true;
                GameEasyGameOverScoreLBL.Visible = true;
                GameEasyGameOverMainMenuLBL.Visible = true;
                GameEasyGameOverPlayAgainLBL.Visible = true;
                GameEasyGameOverScoreLBL.BringToFront();
                GameEasyGameOverPlayAgainLBL.BringToFront();
                GameEasyGameOverMainMenuLBL.BringToFront();
                GameEasyScoreDisplayTB.Visible = false;
                string score = GameEasyScoreDisplayTB.Text;
                label1.Enabled = false;
                //display final highscore
```

```
        if (score.Equals(""))
        {
            finalHighScore = 0;
        }
        else
        {
            finalHighScore = Convert.ToInt32(score);
        }
        GameEasyGameOverScoreLBL.Text =finalHighScore.ToString();
    }
}

//answer checking method
private void SubmitGameEasyLBL_Click(object sender, EventArgs e)
{
    this.currentScore = currentScore;
    string answer = "";
    int currentIndex = 0;
    //check witch question is visible
    for(int i=0; i< easyPhrases.Count; i++)
    {
        if(easyPhrases.ElementAt(i).Key.Visible == true)
        {
            answer = easyPhrases.ElementAt(i).Value;
            currentIndex = i;
        }
    }

   string userAnswer = EasyPhraseAnsTB.Text;
    bool correct = false;
    //string handle the users answer to conform it to the dictionary values
    userAnswer = stringHandling(userAnswer);
    //check if answer is correct
    if (userAnswer == answer)
    {
        EasyPhraseAnsTB.Text = "";
        int randNum = Randomiser();
        SelectPhrase(randNum).Visible = true;
        easyPhrases.ElementAt(currentIndex).Key.Visible = false;
        correct = true;
        currentScore++;
    }
    else
    {
        EasyPhraseAnsTB.Text = "";
        int randNum = Randomiser();
        SelectPhrase(randNum).Visible = true;
        easyPhrases.ElementAt(currentIndex).Key.Visible = false;
        correct = false;
    }
    //display right wrong labels
    if (correct)
    {
        SubmitRightGameEasyLBL.Visible = true;
        SubmitWrongGameEasyLBL.Visible = false;

    }
    else
    {
        SubmitRightGameEasyLBL.Visible = false;
        SubmitWrongGameEasyLBL.Visible = true;
    }
```

```
    GameEasyScoreDisplayTB.Text = currentScore.ToString();

}
//string handling method
private string stringHandling(string userAnswer)
{
    userAnswer = userAnswer.ToLower();
    userAnswer = userAnswer.Replace(" ", "");
    userAnswer = userAnswer.Replace("'", "");
    userAnswer = userAnswer.Replace(",", "");
    userAnswer = userAnswer.Replace("'", "");
    userAnswer = userAnswer.Trim();
    return userAnswer;
}
//question randomiser method
private int Randomiser()
{
    Random r = new Random();
    int genRand = r.Next(0, 50);

    return genRand;
}
//select phrase method
private Label SelectPhrase(int genRand)
{
    Label lbl = easyPhrases.ElementAt(genRand).Key;
    return lbl;
}
//main menu link method
private void GameEasyGameOverMainMenuLBL_Click(object sender, EventArgs e)
{
    this.Hide();
    MainMenu m = new MainMenu(currentUser);
    m.Show();
}
//play again method that reloads form
private void GameEasyGameOverPlayAgainLBL_Click(object sender, EventArgs e)
{
    this.Hide();
    Game gh = new Game(currentUser);
    gh.Show();
}

private void Game_Load(object sender, EventArgs e)
{
    //accidental click
}
//game exit to main menu method
private void gamExitlbl_Click(object sender, EventArgs e)
{
    this.Hide();
    MainMenu m = new MainMenu(currentUser);
    m.Show();
}
//game help label displayer needs to pause game
private void label1_Click(object sender, EventArgs e)
{
    timesClicked++;
    GameInfoLBL.Visible = true;
    EasyPhraseAnsTB.Enabled = false;
```

```
            SubmitGameEasyLBL.Enabled = false;
            if (timesClicked == 2)
            {
                SubmitGameEasyLBL.Enabled = true;
                EasyPhraseAnsTB.Enabled = true;
                GameInfoLBL.Visible = false;
                timesClicked = 0;
                timer2.Start();
            }

        }
        //enter key press to corrolate to submit label click
        private void EasyPhraseAnsTB_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                SubmitGameEasyLBL_Click(sender, e);
            }
        }
    }
}
```

Play Game (Hard Level)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class GameHard : Form
    {
        Dictionary<Label, string> hardPhrases = new Dictionary<Label, string>();
        int currentScore = 0;
        int wait = 0;
        bool correct;
        User currentUser = null;
        int timesClicked = 0;

        public GameHard(User u)
        {
            InitializeComponent();
            currentUser = u;
            //load the question dictionary with label questions that corespond to
string answers
            hardPhrases.Add(GameHardQLBL1, "badapple");
            hardPhrases.Add(GameHardQLBL2, "barkupthewrongtree");
            hardPhrases.Add(GameHardQLBL3, "beataroundthebush");
            hardPhrases.Add(GameHardQLBL4, "breakaleg");
            hardPhrases.Add(GameHardQLBL5, "breakfastclub");
            hardPhrases.Add(GameHardQLBL6, "breaktheice");
            hardPhrases.Add(GameHardQLBL7, "butterflysinstomach");
            hardPhrases.Add(GameHardQLBL8, "captainhook");
            hardPhrases.Add(GameHardQLBL9, "castleintheclouds");
```

```
            hardPhrases.Add(GameHardQLBL10, "coldfeet");
            hardPhrases.Add(GameHardQLBL11, "comparingapplestooranges");
            hardPhrases.Add(GameHardQLBL12, "curveball");
            hardPhrases.Add(GameHardQLBL13, "deathstare");
            hardPhrases.Add(GameHardQLBL14, "dogeatdog");
            hardPhrases.Add(GameHardQLBL15, "dontcountyourchickensbeforetheyhatch");
            hardPhrases.Add(GameHardQLBL16, "dontjudgeabookbyitscover");
            hardPhrases.Add(GameHardQLBL17, "dontputallyoureggsinonebasket");
            hardPhrases.Add(GameHardQLBL18, "dontworrybehappy");
            hardPhrases.Add(GameHardQLBL19, "earlybirdgetstheworm");
            hardPhrases.Add(GameHardQLBL20, "shootingfishinabarrel");
            hardPhrases.Add(GameHardQLBL21, "fortunecookie");
            hardPhrases.Add(GameHardQLBL22, "goodcopbadcop");
            hardPhrases.Add(GameHardQLBL23, "hitthenailonthehead");
            hardPhrases.Add(GameHardQLBL24, "inanutshell");
            hardPhrases.Add(GameHardQLBL25, "ladyluck");
            hardPhrases.Add(GameHardQLBL26, "liarpantsonfire");
            hardPhrases.Add(GameHardQLBL27, "thelionthewitchandthewardrobe");
            hardPhrases.Add(GameHardQLBL28, "loveletter");
            hardPhrases.Add(GameHardQLBL29, "lowkey");
            hardPhrases.Add(GameHardQLBL30, "madcowdisease");
            hardPhrases.Add(GameHardQLBL31, "moneydoesntgrowontrees");
            hardPhrases.Add(GameHardQLBL32, "needleinahaystack");
            hardPhrases.Add(GameHardQLBL33, "nosmokewithoutfire");
            hardPhrases.Add(GameHardQLBL34, "perfectstorm");
            hardPhrases.Add(GameHardQLBL35, "pinsandneedles");
            hardPhrases.Add(GameHardQLBL36, "radiosilence");
            hardPhrases.Add(GameHardQLBL37, "riseandshine");
            hardPhrases.Add(GameHardQLBL38, "rocketscience");
            hardPhrases.Add(GameHardQLBL39, "saveforarainyday");
            hardPhrases.Add(GameHardQLBL40, "seeyoulateraligator");
            hardPhrases.Add(GameHardQLBL41, "shootingstar");
            hardPhrases.Add(GameHardQLBL42, "sickasadog");
            hardPhrases.Add(GameHardQLBL43, "sickleave");
            hardPhrases.Add(GameHardQLBL44, "strikewhiletheironishot");
            hardPhrases.Add(GameHardQLBL45, "thinkoutsidethebox");
            hardPhrases.Add(GameHardQLBL46, "timessquare");
            hardPhrases.Add(GameHardQLBL47, "toolbox");
            hardPhrases.Add(GameHardQLBL48, "wakeupandsmelltheroses");
            hardPhrases.Add(GameHardQLBL49, "walkingdead");
            hardPhrases.Add(GameHardQLBL50, "whatsupdog");

            //set all question labels visible to false
            for (int i = 0; i < hardPhrases.Count; i++)
            {
                hardPhrases.ElementAt(i).Key.Visible = false;
            }
            //randomise first question
            Random ra = new Random();
            int randQ = ra.Next(0, 50);
            hardPhrases.ElementAt(randQ).Key.Visible = true; //TODO randomise this so
it doesnt always start on same one DONE
            SubmitRightGameHardLBL.Visible = false;
            SubmitWrongGameHardLBL.Visible = false;
            //begin start sequence
            timer5.Tick += new EventHandler(timer5_Tick_1);
            timer5.Enabled = true;
            timer5.Start();

            GameHardTrafficLight1.BringToFront();
            GameHardTrafficLight2.BringToFront();
            GameHardTrafficLight3.BringToFront();
```

```
            GameHardTrafficLight4.BringToFront();
        }



        //format the users answer to be validated
        private string stringHandling(string userAnswer)
        {
            userAnswer = userAnswer.ToLower();
            userAnswer = userAnswer.Replace(" ", "");
            userAnswer = userAnswer.Replace("'", "");
            userAnswer = userAnswer.Replace(",", "");
            userAnswer = userAnswer.Replace("'", "");
            userAnswer = userAnswer.Trim();
            return userAnswer;
        }
        //randomiser for random questions
        private int Randomiser()
        {
            Random r = new Random();
            int genRand = r.Next(0, 50);

            return genRand;
        }
        //randomise questions
        private Label SelectPhrase(int genRand)
        {
            Label lbl = hardPhrases.ElementAt(genRand).Key;
            return lbl;
        }

        private void SubmitRightGameHardLBL_Click(object sender, EventArgs e)
        {
            //tick lbl needs no code acciental click
        }

        //Move clock for timer method
        private void timer4_Tick_1(object sender, EventArgs e)
        {
            int step = -1;
            int xMax = 790;
            this.GameHardClockLBL.Location = new
Point(this.GameHardClockLBL.Location.X + step, this.GameHardClockLBL.Location.Y);

            GameHardScoreDisplayTB.Visible = true;

            //Pause game
            if (GameHardInfoLBL.Visible == true)
            {
                GameHardScoreDisplayTB.Visible = false;
                timer4.Stop();
            }
            //timer method that increments timer and decrements timer
            if (correct == true && GameHardClockLBL.Location.X + 100 <= xMax)//xmx
stops clock from moving off screen
            {
                //x max is 790
                this.GameHardClockLBL.Location = new
Point(this.GameHardClockLBL.Location.X + 100, this.GameHardClockLBL.Location.Y);
                correct = false;
            }
            //game over sequence
```

```
            if (GameHardClockLBL.Location.X == 130)
            {
                int finalHighScore = 0;
                timer4.Stop();
                GameHardGameOverLBL.Visible = true;
                GameHardGameOverScoreLBL.Visible = true;
                GameHardGameOverMainMenuLBL.Visible = true;
                GameHardGameOverPlayAgainLBL.Visible = true;
                GameHardGameOverPlayAgainLBL.BringToFront();
                GameHardGameOverMainMenuLBL.BringToFront();
                GameHardGameOverScoreLBL.BringToFront();
                gameHardqMarklbl.Enabled = false;
                GameHardGameOverScoreLBL.Text = GameHardScoreDisplayTB.Text;
                string score = GameHardScoreDisplayTB.Text;
                //final score written to database
                //validation to ensure high score is higher than current score
                if (score.Equals(""))
                {
                    finalHighScore = 0;
                }
                else
                {
                    finalHighScore = Convert.ToInt32(score);
                }
                GameHardGameOverScoreLBL.Text = finalHighScore.ToString();
                currentUser.setHighScore(finalHighScore);
                DataBase db = new DataBase(currentUser.getUsername(),
currentUser.getPassword(), currentUser.getEmail(), currentUser.getAvatar(),
currentUser.getHighScore());
                db.updateHighScore(currentUser.getUsername(), 0, finalHighScore);
            }
        }

        private void timer5_Tick_1(object sender, EventArgs e)//game starting sequence
        {
            GameHardTrafficLight1.BringToFront();
            GameHardTrafficLight2.BringToFront();
            GameHardTrafficLight3.BringToFront();
            GameHardTrafficLight4.BringToFront();
            GameHardScoreDisplayTB.SendToBack();
            gameHardqMarklbl.Enabled = false;

            wait++;
            if (wait == 2)
            {
                GameHardTrafficLight1.BringToFront();
                GameHardTrafficLight2.BringToFront();
                GameHardTrafficLight3.BringToFront();
                GameHardTrafficLight4.BringToFront();
                GameHardScoreDisplayTB.Visible = false;
                GameHardTrafficLight1.Visible = true;
            }
            else if (wait == 3)
            {
                GameHardTrafficLight1.BringToFront();
                GameHardTrafficLight2.BringToFront();
                GameHardTrafficLight3.BringToFront();
                GameHardTrafficLight4.BringToFront();
                GameHardTrafficLight1.Visible = false;
                GameHardTrafficLight2.Visible = true;
            }
            else if (wait == 4)
```

```
    {
        GameHardTrafficLight1.BringToFront();
        GameHardTrafficLight2.BringToFront();
        GameHardTrafficLight3.BringToFront();
        GameHardTrafficLight4.BringToFront();
        GameHardTrafficLight1.Visible = false;
        GameHardTrafficLight2.Visible = false;
        GameHardTrafficLight3.Visible = true;
    }
    else if (wait == 5)
    {
        GameHardTrafficLight1.Visible = false;
        GameHardTrafficLight2.Visible = false;
        GameHardTrafficLight3.Visible = false;
        GameHardTrafficLight4.Visible = true;

    }
    else if (wait == 6)
    {
        GameHardTrafficLight4.Visible = false;
        GameHardReadyLBL.Visible = false;
        GameHardScoreDisplayTB.BringToFront();

        GameHardScoreDisplayTB.Visible = true;

        timer5.Stop();
        gameHardqMarklbl.Enabled = true;

        timer4.Tick += new EventHandler(timer4_Tick_1);
        timer4.Enabled = true;
        timer4.Start();
    }
}
//answer validation on submit click
private void GameHardSubmitLBL_Click(object sender, EventArgs e)
{
    this.currentScore = currentScore;
    string answer = "";
    int currentIndex = 0;
    //for loop to check what question was asked
    for (int i = 0; i < hardPhrases.Count; i++)
    {
        if (hardPhrases.ElementAt(i).Key.Visible == true)
        {
            answer = hardPhrases.ElementAt(i).Value;
            currentIndex = i;
        }
    }
    //answer formatting with string handling
    string userAnswer = HardPhraseAnsTB.Text;
    userAnswer = stringHandling(userAnswer);
    //asnwer validation
    if (userAnswer == answer)
    {
        HardPhraseAnsTB.Text = "";
        int randNum = Randomiser();
        SelectPhrase(randNum).Visible = true;
        hardPhrases.ElementAt(currentIndex).Key.Visible = false;
        correct = true;
        currentScore++;
    }
    else
```

```
            {
                HardPhraseAnsTB.Text = "";
                int randNum = Randomiser();
                SelectPhrase(randNum).Visible = true;
                hardPhrases.ElementAt(currentIndex).Key.Visible = false;
            }
            //show right wrong label answer
            if (correct)
            {
                SubmitRightGameHardLBL.Visible = true;
                SubmitWrongGameHardLBL.Visible = false;

            }
            else
            {
                SubmitRightGameHardLBL.Visible = false;
                SubmitWrongGameHardLBL.Visible = true;
            }

            GameHardScoreDisplayTB.Text = currentScore.ToString();
        }


        private void GameHardSubmitLBL_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
        {
            //control test no code needed
        }
        //reload form method
        private void GameHardGameOverPlayAgainLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            GameHard gh = new GameHard(currentUser);
            gh.Show();
        }
        //bring user to main menu method
        private void GameHardGameOverMainMenuLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            MainMenu m = new MainMenu(currentUser);
            m.Show();
        }
        //pause game and show help label method
        private void gameHardqMarklbl_Click(object sender, EventArgs e)
        {
            timesClicked++;
            GameHardInfoLBL.Visible = true;
            HardPhraseAnsTB.Enabled = false;
            GameHardSubmitLBL.Enabled = false;
            if (timesClicked == 2)
            {
                HardPhraseAnsTB.Enabled = true;
                GameHardSubmitLBL.Enabled = true;
                GameHardInfoLBL.Visible = false;
                timesClicked = 0;
                timer4.Start();
            }
        }
        //bring user to main menu method
        private void gameHardExit_Click(object sender, EventArgs e)
        {
            this.Hide();
```

```
            MainMenu m = new MainMenu(currentUser);
            m.Show();
        }
        //allow enter key to be tyed to submit label click
        private void HardPhraseAnsTB_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                GameHardSubmitLBL_Click(sender, e);
            }
        }
    }
}
```

Play Game (Bonus Level)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GuessThatWord
{
    public partial class BonusGames : Form
    {
        string imageDetails = "";
        static bool picBox1ImageChanged = false;
        static bool picBox2ImageChanged = false;
        static bool picBox3ImageChanged = false;
        static int imageDragged = 0;
        Random rand = new Random();
        int num = 0;
        User currentUser = null;


        Dictionary<string, string> myQuestionDICT = new Dictionary<string, string>();
        Dictionary<Label, string> imageLBLDICT = new Dictionary<Label, string>();
        int timesClicked = 0;


        public BonusGames(User u)
        {
            InitializeComponent();
            currentUser = u;

            myQuestionDICT.Add("Enfermo como un perro (Sick as a dog)",
"sick,point,dog,");
            myQuestionDICT.Add("Que hay perro? (What's up dog?)", "what,up,dog,");
            myQuestionDICT.Add("Mariposas en el estomago (butterflies in your
stomach)", "butterfly,point,woman,");
            myQuestionDICT.Add("En una palabra (in a nutshell)", "point,nut,shell,");
            myQuestionDICT.Add("Que es ese olor? (what's that smell?)",
"what,point,nose,");
            myQuestionDICT.Add("Despierta y huele las rosas (wake up and smell the
roses)", "bed,nose,rose,");
```

```
            myQuestionDICT.Add("Nariz como un perro (nose like a dog)",
"nose,point,dog,");
            myQuestionDICT.Add("Nauseas matutinas (Morning sickness)",
"bed,woman,sick,");

            imageLBLDICT.Add(DragLbl1, "woman,");
            imageLBLDICT.Add(DragLbl2, "point,");
            imageLBLDICT.Add(DragLbl3, "butterfly,");
            imageLBLDICT.Add(DragLbl4, "dog,");
            imageLBLDICT.Add(DragLbl5, "nose,");
            imageLBLDICT.Add(DragLbl6, "bed,");
            imageLBLDICT.Add(DragLbl7, "up,");
            imageLBLDICT.Add(DragLbl8, "sick,");
            imageLBLDICT.Add(DragLbl9, "nut,");
            imageLBLDICT.Add(DragLbl10, "what,");
            imageLBLDICT.Add(DragLbl11, "rose,");
            imageLBLDICT.Add(DragLbl12, "shell,");

            num = rand.Next(0, myQuestionDICT.Count());
            QuestionLbl.Text = myQuestionDICT.Keys.ElementAt(num);

        }

        private void BonusGamesExitLBL_Click(object sender, EventArgs e)
        {
            this.Hide();
            MainMenu m = new MainMenu(currentUser);
            m.Show();
        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void pbAnswer1_Click(object sender, EventArgs e)
        {

        }

        private void BonusGames_Load(object sender, EventArgs e)
        {
            pbAnswer1.AllowDrop = true;
            pbAnswer2.AllowDrop = true;
            pbAnswer3.AllowDrop = true;
        }

        private void DragLbl1_MouseDown(object sender, MouseEventArgs e)
        {
            Bitmap bmp = new Bitmap(DragLbl1.Width, DragLbl1.Height);
            DragLbl1.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
            bmp.MakeTransparent(Color.White);

            Cursor cur = new Cursor(bmp.GetHicon());
            Cursor.Current = cur;

            imageDragged = 1;
        }

        private void DragLbl2_MouseDown(object sender, MouseEventArgs e)
        {
            Bitmap bmp = new Bitmap(DragLbl2.Width, DragLbl2.Height);
```

```
    DragLbl2.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 2;
}

private void DragLbl3_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl3.Width, DragLbl3.Height);
    DragLbl3.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 3;
}

private void DragLbl4_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl4.Width, DragLbl4.Height);
    DragLbl4.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 4;
}

private void DragLbl5_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl5.Width, DragLbl5.Height);
    DragLbl5.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 5;
}

private void DragLbl6_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl6.Width, DragLbl6.Height);
    DragLbl6.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 6;
}

private void DragLbl7_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl7.Width, DragLbl7.Height);
    DragLbl7.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);
```

```
    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 7;
}

private void DragLbl8_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl8.Width, DragLbl8.Height);
    DragLbl8.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 8;
}

private void DragLbl9_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl9.Width, DragLbl9.Height);
    DragLbl9.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 9;
}

private void DragLbl10_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl10.Width, DragLbl10.Height);
    DragLbl10.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 10;
}

private void DragLbl11_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl11.Width, DragLbl11.Height);
    DragLbl11.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
    Cursor.Current = cur;

    imageDragged = 11;
}

private void DragLbl12_MouseDown(object sender, MouseEventArgs e)
{
    Bitmap bmp = new Bitmap(DragLbl12.Width, DragLbl12.Height);
    DragLbl12.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
    bmp.MakeTransparent(Color.White);

    Cursor cur = new Cursor(bmp.GetHicon());
```

```
    Cursor.Current = cur;

    imageDragged = 12;
}

private void pbAnswer1_DragEnter(object sender, DragEventArgs e)
{
    e.Effect = e.AllowedEffect;
    picBox1ImageChanged = true;
}

private void pbAnswer1_DragDrop(object sender, DragEventArgs e)
{
    pbAnswer1.Image = (Bitmap)e.Data.GetData(DataFormats.Bitmap);
}

private void pbAnswer2_DragEnter(object sender, DragEventArgs e)
{
    e.Effect = e.AllowedEffect;
    picBox2ImageChanged = true;
}

private void pbAnswer2_DragDrop(object sender, DragEventArgs e)
{
    pbAnswer2.Image = (Bitmap)e.Data.GetData(DataFormats.Bitmap);
}

private void pbAnswer3_DragEnter(object sender, DragEventArgs e)
{
    e.Effect = e.AllowedEffect;
    picBox3ImageChanged = true;
}

private void pbAnswer3_DragDrop(object sender, DragEventArgs e)
{
    pbAnswer3.Image = (Bitmap)e.Data.GetData(DataFormats.Bitmap);
}

private void pbAnswer1_MouseEnter(object sender, EventArgs e)
{
    if (imageDragged == 1)
    {
        DragLbl1.DoDragDrop(DragLbl1.Image, DragDropEffects.Copy);
        pbAnswer1.Tag = "DragLbl1";
        imageDragged = 0;

    }
    if (imageDragged == 2)
    {
        DragLbl2.DoDragDrop(DragLbl2.Image, DragDropEffects.Copy);
        pbAnswer1.Tag = "DragLbl2";
        imageDragged = 0;


    }
    if (imageDragged == 3)
    {
        DragLbl3.DoDragDrop(DragLbl3.Image, DragDropEffects.Copy);
        pbAnswer1.Tag = "DragLbl3";
        imageDragged = 0;

    }
```

```
if(imageDragged == 4)
{
    DragLbl4.DoDragDrop(DragLbl4.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl4";
    imageDragged = 0;

}
if (imageDragged == 5)
{
    DragLbl5.DoDragDrop(DragLbl5.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl5";
    imageDragged = 0;


}
if (imageDragged == 6)
{
    DragLbl6.DoDragDrop(DragLbl6.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl6";
    imageDragged = 0;

}
if (imageDragged == 7)
{
    DragLbl7.DoDragDrop(DragLbl7.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl7";
    imageDragged = 0;

}
if (imageDragged == 8)
{
    DragLbl8.DoDragDrop(DragLbl8.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl8";
    imageDragged = 0;


}
if (imageDragged == 9)
{
    DragLbl9.DoDragDrop(DragLbl9.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl9";
    imageDragged = 0;

}
if (imageDragged == 10)
{
    DragLbl10.DoDragDrop(DragLbl10.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl10";
    imageDragged = 0;

}
if (imageDragged == 11)
{
    DragLbl11.DoDragDrop(DragLbl11.Image, DragDropEffects.Copy);
    pbAnswer1.Tag = "DragLbl11";
    imageDragged = 0;


}
if (imageDragged == 12)
{
    DragLbl12.DoDragDrop(DragLbl12.Image, DragDropEffects.Copy);
```

```
        pbAnswer1.Tag = "DragLbl12";
        imageDragged = 0;

    }
}

private void pbAnswer2_MouseEnter(object sender, EventArgs e)
{
    if (imageDragged == 1)
    {
        DragLbl1.DoDragDrop(DragLbl1.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl1";
        imageDragged = 0;

    }
    if (imageDragged == 2)
    {
        DragLbl2.DoDragDrop(DragLbl2.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl2";
        imageDragged = 0;


    }
    if (imageDragged == 3)
    {
        DragLbl3.DoDragDrop(DragLbl3.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl3";
        imageDragged = 0;

    }
    if (imageDragged == 4)
    {
        DragLbl4.DoDragDrop(DragLbl4.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl4";
        imageDragged = 0;

    }
    if (imageDragged == 5)
    {
        DragLbl5.DoDragDrop(DragLbl5.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl5";
        imageDragged = 0;


    }
    if (imageDragged == 6)
    {
        DragLbl6.DoDragDrop(DragLbl6.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl6";
        imageDragged = 0;

    }
    if (imageDragged == 7)
    {
        DragLbl7.DoDragDrop(DragLbl7.Image, DragDropEffects.Copy);
        pbAnswer2.Tag = "DragLbl7";
        imageDragged = 0;

    }
    if (imageDragged == 8)
    {
        DragLbl8.DoDragDrop(DragLbl8.Image, DragDropEffects.Copy);
```

```
            pbAnswer2.Tag = "DragLbl8";
            imageDragged = 0;


        }
        if (imageDragged == 9)
        {
            DragLbl9.DoDragDrop(DragLbl9.Image, DragDropEffects.Copy);
            pbAnswer2.Tag = "DragLbl9";
            imageDragged = 0;

        }
        if (imageDragged == 10)
        {
            DragLbl10.DoDragDrop(DragLbl10.Image, DragDropEffects.Copy);
            pbAnswer2.Tag = "DragLbl10";
            imageDragged = 0;

        }
        if (imageDragged == 11)
        {
            DragLbl11.DoDragDrop(DragLbl11.Image, DragDropEffects.Copy);
            pbAnswer2.Tag = "DragLbl11";
            imageDragged = 0;


        }
        if (imageDragged == 12)
        {
            DragLbl12.DoDragDrop(DragLbl12.Image, DragDropEffects.Copy);
            pbAnswer2.Tag = "DragLbl12";
            imageDragged = 0;

        }
    }

    private void pbAnswer3_MouseEnter(object sender, EventArgs e)
    {
        if (imageDragged == 1)
        {
            DragLbl1.DoDragDrop(DragLbl1.Image, DragDropEffects.Copy);
            pbAnswer3.Tag = "DragLbl1";
            imageDragged = 0;

        }
        if (imageDragged == 2)
        {
            DragLbl2.DoDragDrop(DragLbl2.Image, DragDropEffects.Copy);
            pbAnswer3.Tag = "DragLbl2";
            imageDragged = 0;


        }
        if (imageDragged == 3)
        {
            DragLbl3.DoDragDrop(DragLbl3.Image, DragDropEffects.Copy);
            pbAnswer3.Tag = "DragLbl3";
            imageDragged = 0;

        }
        if (imageDragged == 4)
        {
```

```
    DragLbl4.DoDragDrop(DragLbl4.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl4";
    imageDragged = 0;

}
if (imageDragged == 5)
{
    DragLbl5.DoDragDrop(DragLbl5.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl5";
    imageDragged = 0;


}
if (imageDragged == 6)
{
    DragLbl6.DoDragDrop(DragLbl6.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl6";
    imageDragged = 0;

}
if (imageDragged == 7)
{
    DragLbl7.DoDragDrop(DragLbl7.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl7";
    imageDragged = 0;

}
if (imageDragged == 8)
{
    DragLbl8.DoDragDrop(DragLbl8.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl8";
    imageDragged = 0;


}
if (imageDragged == 9)
{
    DragLbl9.DoDragDrop(DragLbl9.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl9";
    imageDragged = 0;

}
if (imageDragged == 10)
{
    DragLbl10.DoDragDrop(DragLbl10.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl10";
    imageDragged = 0;

}
if (imageDragged == 11)
{
    DragLbl11.DoDragDrop(DragLbl11.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl11";
    imageDragged = 0;


}
if (imageDragged == 12)
{
    DragLbl12.DoDragDrop(DragLbl12.Image, DragDropEffects.Copy);
    pbAnswer3.Tag = "DragLbl12";
    imageDragged = 0;
```

```
            }
        }

        private void SubmitLbl_Click(object sender, EventArgs e)
        {
            string currentQuestion = QuestionLbl.Text;
            bool correct = false;
            //check if all picture boxes contain an image
            if (picBox1ImageChanged && picBox2ImageChanged && picBox3ImageChanged)
            {
                //iterate through dictionary imagelbl dictionary to match image to
string
                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer1.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }

                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer2.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }

                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer3.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }
            }

            for (int i = 0; i < myQuestionDICT.Count(); i++)
            {
                if (myQuestionDICT.Values.ElementAt(i) == imageDetails)
                {
                    if (currentQuestion == myQuestionDICT.Keys.ElementAt(i))
                    {
                        correct = true;
                    }
                }

            }

            if (correct)
            {
                CorrectLbl.Visible = true;
                IncorrectLbl.Visible = false;
            }
            else
            {
                CorrectLbl.Visible = false;
                IncorrectLbl.Visible = true;
```

```
            }


            num = rand.Next(0, myQuestionDICT.Count());
            QuestionLbl.Text = myQuestionDICT.Keys.ElementAt(num);

            pbAnswer1.Image = null;
            pbAnswer2.Image = null;
            pbAnswer3.Image = null;
            imageDetails = "";
        }

        private void BonusGamesHelplbl_Click(object sender, EventArgs e)
        {
            timesClicked++;
            BonusGameInfoLBL.Visible = true;
            if (timesClicked == 2)
            {
                BonusGameInfoLBL.Visible = false;
                timesClicked = 0;
            }

        }
    }
}
```

Admin Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Mail;

namespace GuessThatWord
{
    public partial class AdminPage : Form
    {
        public Admin currentAdmin = null;
        public DataBase db = new DataBase();
        public AdminPage(Admin a)
        {
            currentAdmin = a;
            InitializeComponent();
            label1.Enabled = false;
            List<User> users = db.getAllUsers();
            for(int i = 0; i<users.Count(); i++)
            {
                userDropDown.Items.Add(users[i].getUsername());
            }
        }

        private void userDropDown_TextUpdate(object sender, EventArgs e)
        {
```

```
        }

        private void userDropDown_SelectedIndexChanged(object sender, EventArgs e)
        {
            string searchTerm = userDropDown.Text;
            string[] fields = db.readRecord(searchTerm, 0, "database.csv");
            oldUsernamelbl.Text = fields[0];
            oldPasswordlbl.Text = fields[1];
            oldEmaillbl.Text = fields[2];
            oldAvatarlbl.Text = fields[3];
            oldScorelbl.Text = fields[4];
            label1.Enabled = true;
        }
        //methods to individually change aspects of a users info
        private void passwordSubmit_Click(object sender, EventArgs e)
        {
            string searchTerm = oldUsernamelbl.Text;
            string newValue = newPasswordtb.Text;
            if(currentAdmin.updateUser(searchTerm, 1, newValue))
            {

                //send email to current user that their password has been changed
                try
                {
                    SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

                    //Authentication - ensures the program has a valid email to send
from
                    NetworkCredential cred = new
NetworkCredential("GuessThatPhrase@gmail.com", "boekdvvnrbyspxye");

                    MailMessage msg = new MailMessage();

                    msg.From = new MailAddress("GuessThatPhrase@gmail.com");

                    msg.To.Add(oldEmaillbl.Text);

                    msg.Subject = "Password Reset Notification";

                    msg.Body = "Your password has been updated to " +
newPasswordtb.Text + " please login to your account!";

                    client.UseDefaultCredentials = false;

                    client.Credentials = cred;

                    //enabling SSL (Secure sockets layer, encryption) this is required
by most email providers to send mail
                    client.EnableSsl = true;

                    client.Send(msg);

                    MessageBox.Show("Success, user has been notified of password
update!");
                    this.Hide();
                    AdminPage ap = new AdminPage(currentAdmin);
                    ap.Show();
                }

                catch
                {
                    MessageBox.Show("Something went wrong..");
```

```
            }
        }
        else
        {
            MessageBox.Show("Error");
        }

    }

    private void emailSubmit_Click(object sender, EventArgs e)
    {
        string searchTerm = oldUsernamelbl.Text;
        string newValue = newEmailtb.Text;
        if (currentAdmin.updateUser(searchTerm, 2, newValue))
        {
            MessageBox.Show("Success!");
            this.Hide();
            AdminPage ap = new AdminPage(currentAdmin);
            ap.Show();
        }
        else
        {
            MessageBox.Show("Error");
        }
    }

    private void avatarSubmit_Click(object sender, EventArgs e)
    {
        string searchTerm = oldUsernamelbl.Text;
        string newValue = avatarCB.Text;
        if (currentAdmin.updateUser(searchTerm, 3, newValue))
        {
            MessageBox.Show("Success!");
            this.Hide();
            AdminPage ap = new AdminPage(currentAdmin);
            ap.Show();
        }
        else
        {
            MessageBox.Show("Error");
        }
    }

    private void highScoreSubmit_Click(object sender, EventArgs e)
    {
        string searchTerm = oldUsernamelbl.Text;
        string newValue = newScoretb.Text;
        if (currentAdmin.updateUser(searchTerm, 4, newValue))
        {
            MessageBox.Show("Success!");
            this.Hide();
            AdminPage ap = new AdminPage(currentAdmin);
            ap.Show();
        }
        else
        {
            MessageBox.Show("Error");
        }
    }
    //delete user method
    private void label1_Click(object sender, EventArgs e)
    {
```

```
            //method to delete user
            //add in are you sure
            string searchTerm = oldUsernamelbl.Text;
            DialogResult dialogResult = MessageBox.Show("Are you sure you want to
delete this user? Action cannot be undone", "Conformation", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                if (currentAdmin.deleteUser(searchTerm))
                {
                    MessageBox.Show("Success!");
                }
                else
                {
                    MessageBox.Show("Error");
                }
            }
            else if (dialogResult == DialogResult.No)
            {
                MessageBox.Show("Action aborted");
            }

            //reload admin page
            this.Hide();
            AdminPage ap = new AdminPage(currentAdmin);
            ap.Show();
        }
        //admin exit brings to login
        private void adminExitlbl_Click(object sender, EventArgs e)
        {
            this.Hide();
            Welcome w = new Welcome();
            w.Show();
        }
    }
}
```

User Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GuessThatWord
{
    public class User
    {
        private string username;
        private string password;
        private string email;
        private string avatar;
        private int highScore;

        //parameratised constructor for user
        public User(string username, string password, string email)
        {
            this.username = username;
            this.password = password;
            this.email = email;
            this.avatar = "";
```

```
            this.highScore = 0;
        }
        //gets and sets for user information
        public string getUsername()
        {
            return username;
        }
        public void setUsername(string username)
        {
            this.username = username;
        }
        public void setemail(string email)
        {
            this.email = email;
        }
        public void setPassword(string password)
        {
            this.password = password;
        }
        public string getPassword()
        {
            return password;
        }
        public string getEmail()
        {
            return email;
        }
        public string getAvatar()
        {
            return avatar;
        }
        public int getHighScore()
        {
            return highScore;
        }
        public void setAvatar(string avatar)
        {
            this.avatar = avatar;
        }
        public void setHighScore(int highScore)
        {
            this.highScore = highScore;
        }
        //method to add user to csv file from the database class
        public bool addUser()
        {

            DataBase d = new DataBase(username, password, email, avatar, highScore);
            bool success = d.addRecord("");

            if (success)
            {
                return true;
            }
            return false;

        }
    }
}
```

Admin Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GuessThatWord
{
    public class Admin : User
    {
        public DataBase db = null;
        //admin constructer(parameratised)
        public Admin(string username, string password, string email):base(username,
password, email)
        {
            base.setUsername(username);
            base.setPassword(password);
            base.setemail(email);
            db = new DataBase();
        }
        //method to edit user
        public bool updateUser(string user, int positionOfFieldToUpdate, string
newValue)
        {
            try
            {
                db.editRecord(user, positionOfFieldToUpdate, newValue);
                return true;
            }
            catch (Exception ex2)
            {
                Console.WriteLine("This program did not work");
                return false;
                throw new ApplicationException("This program did not work :", ex2);

            }

        }
        //method to delete user
        public bool deleteUser(string username)
        {
            //if the edit record is given a certain number in position of search term
it triggers a delete
            try
            {
                db.editRecord(username, 5, "");
                return true;
            }
            catch (Exception ex2)
            {
                Console.WriteLine("This program did not work");
                return false;
                throw new ApplicationException("This program did not work :", ex2);

            }

        }
    }
}
```

Database Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace GuessThatWord
{
    public class DataBase
    {

        private string usernameRecord;
        private string passwordRecord;
        private string emailRecord;
        private string avatarRecord;
        private int highScoreRecord;
        //States a constant unchanging variable as the filepath is always the same
        private const string Filepath = "database.csv";

        //Default and parameratised DataBase constructor
        public DataBase()
        {

        }
        public DataBase(string record1, string record2, string record3, string
record4, int record5)
        {
            this.usernameRecord = record1;
            this.passwordRecord = record2;
            this.emailRecord = record3;
            this.avatarRecord = record4;
            this.highScoreRecord = record5;

        }

        public bool addRecord(string filePath)
        {
            string f = "database.csv";
            //check file exists if yes add user if not create file
            try
            {
                if (filePath != "")
                {
                    f = filePath;
                }
                string[] result = readRecord(usernameRecord, 0, f);
                if (result[0] == "Record not found")
                {
                    using (System.IO.StreamWriter file = new System.IO.StreamWriter(f,
true))
                    {
                        file.WriteLine(usernameRecord + "," + passwordRecord + "," +
emailRecord + "," + avatarRecord + "," + highScoreRecord);
                        return true;
                    }
                }
                return false;
```

```
            }
            catch (Exception ex)
            {
                throw new ApplicationException("This program did not work :", ex);
            }
        }

        public void updateHighScore(string searchTerm, int positionOfSearchTerm, int
newValue)
        {
            string tempFile = "temp.csv";
            //logic to see if current score is bigger than score in csv

            string[] originalFields = readRecord(searchTerm, positionOfSearchTerm,
Filepath);
            int currentHighScore = Convert.ToInt32(originalFields[4]);
            if (newValue > currentHighScore)
            {
                try
                {
                    string[] lines = File.ReadAllLines(Filepath);

                    for (int i = 0; i < lines.Length; i++)
                    {
                        string[] fields = lines[i].Split(',');

                        if (i < lines.Length)
                        {
                            //add record manually
                            string[] result = readRecord(fields[0], 0, tempFile);
                            if (result[0] == "Record not found")
                            {
                                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(tempFile, true))
                                {
                                    if (fields[0] != searchTerm)
                                    {
                                        file.WriteLine(fields[0] + "," + fields[1] +
"," + fields[2] + "," + fields[3] + "," + fields[4]);
                                    }
                                    else
                                    {

                                        file.WriteLine(fields[0] + "," + fields[1] +
"," + fields[2] + "," + fields[3] + "," + newValue);
                                    }
                                }
                            }
                            Console.WriteLine("Edited");
                        }
                    }

                    File.Delete(Filepath);

                    File.Move(tempFile, Filepath);

                    Console.WriteLine("Reccord edited");
                }
                catch (Exception ex2)
                {
                    Console.WriteLine("This program did not work");
```

```
                    throw new ApplicationException("This program did not work :",
ex2);
                }
            }

        }

        public void editRecord(string searchTerm, int positionOfNewValue, string
newValue)
        {
            string tempFile = "temp.csv";
            //if positionOfNewValue = 5 that triggers the delete user and all fields
are updated to empty

            try
            {
                string[] lines = File.ReadAllLines(Filepath);

                for (int i = 0; i < lines.Length; i++)
                {
                    string[] fields = lines[i].Split(',');

                    if (i < lines.Length)
                    {
                        //add record manually
                        string[] result = readRecord(fields[0], 0, tempFile);
                        if (result[0] == "Record not found")
                        {
                            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(tempFile, true))
                            {
                                if (fields[0] != searchTerm)
                                {
                                    file.WriteLine(fields[0] + "," + fields[1] + "," +
fields[2] + "," + fields[3] + "," + fields[4]);
                                }
                                else
                                {
                                    //Find the position of the new value
                                    switch (positionOfNewValue)
                                    {
                                        case 1: file.WriteLine(fields[0] + "," +
newValue + "," + fields[2] + "," + fields[3] + "," + fields[4]);
                                            break;
                                        case 2: file.WriteLine(fields[0] + "," +
fields[1] + "," + newValue + "," + fields[3] + "," + fields[4]);
                                            break;
                                        case 3: file.WriteLine(fields[0] + "," +
fields[1] + "," + fields[2] + "," + newValue + "," + fields[4]);
                                            break;
                                        case 4: file.WriteLine(fields[0] + "," +
fields[1] + "," + fields[2] + "," + fields[3] + "," + newValue);
                                            break;
                                        case 5: file.WriteLine("" + "," + "" + "," +
"" + "," + "" + "," + "");
                                            break;
                                        default: file.WriteLine(fields[0] + "," +
fields[1] + "," + fields[2] + "," + fields[3] + "," + fields[4]);
                                            break;
                                    }

                                }
```

```
                        }
                    }
                    Console.WriteLine("Edited");
                }
            }

            File.Delete(Filepath);

            File.Move(tempFile, Filepath);

            Console.WriteLine("Reccord edited");
        }
        catch (Exception ex2)
        {
            Console.WriteLine("This program did not work");
            throw new ApplicationException("This program did not work :", ex2);
        }

    }
    //reuseable method to read read the DataBase file and return a users
information
    public string[] readRecord(string searchTerm, int positionOfSearchTerm, string
filePath)
    {
        string[] recordNotFound = { "Record not found" };

        try
        {
            string[] lines = System.IO.File.ReadAllLines(filePath);

            for (int i = 0; i < lines.Length; i++)
            {
                string[] fields = lines[i].Split(',');
                if (recordMatches(searchTerm, fields, positionOfSearchTerm))
                {
                    return fields;
                }
            }
            return recordNotFound;
        }
        catch (Exception ex1)
        {
            return recordNotFound;
            throw new ApplicationException("This program did not work :", ex1);
        }


    }
    //Bring username and high score into a list to sort and return to leader board
    public List<User> getLeaderboardInfo()
    {
        int count = 0;
        List<User> users = new List<User>();
        //Search the csv file
        try
        {
            string[] lines = System.IO.File.ReadAllLines("database.csv");


            for (int i = 0; i < lines.Length; i++)
            {
```

```
                    if(lines[i] != ",,,,")//neede due to deleted user method not
deleting commas(intented)
                    {
                        string[] fields = lines[i].Split(',');
                        User user = new User(fields[0], fields[1], fields[2]);
                        user.setHighScore(Convert.ToInt32(fields[4]));
                        users.Add(user);
                    }

                }
                //sort the user list by descending
                var sortedUsers = users.OrderByDescending(u => u.getHighScore());
                List<User> highestScoringUsers = new List<User>(3);

                //only bring back top 3
                for(int i = 0; i<sortedUsers.Count(); i++)
                {
                    count++;
                    if (count <= 3)
                    {
                        highestScoringUsers.Add(sortedUsers.ElementAt(i));
                    }
                    else
                    {
                        break;
                    }
                }

                return highestScoringUsers;
            }
            catch (Exception ex1)
            {
                return users;
                throw new ApplicationException("This program did not work :", ex1);
            }

        }
        //method to put all users into a list for admin drop down
        public List<User> getAllUsers()
        {
            List<User> users = new List<User>();
            //iterate through csv and add valid fields to list
            try
            {
                string[] lines = System.IO.File.ReadAllLines("database.csv");

                for (int i = 0; i < lines.Length; i++)
                {
                    if (lines[i] != ",,,,") //when information is removed commas
remain
                    {
                        string[] fields = lines[i].Split(',');
                        User user = new User(fields[0], fields[1], fields[2]);
                        user.setAvatar(fields[3]);
                        user.setHighScore(Convert.ToInt32(fields[4]));
                        users.Add(user);
                    }

                }

                return users;
            }
```

```
            catch (Exception ex1)
            {
                return users;
                throw new ApplicationException("This program did not work :", ex1);
            }
        }

        //method to check if the reccord matches the target reccord
        public bool recordMatches(string searchTerm, string[] record, int
positionOfSearchTerm)
        {
            //if the logics search term matches the passed in search term return true
            if (record[positionOfSearchTerm].Equals(searchTerm))
            {
                return true;
            }
            return false;
        }

    }
}
```

Validation Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GuessThatWord
{
    class Validation
    {
        private string username;
        private string password;
        private string confirmPassword;
        private string email;

        //Login details for an admin theese are constant as there is only one admin
        private const string adminUsername = "admin1234";
        private const string adminPassword = "AA!!11adminpass";

        //Parameterised constructor
        public Validation(string username, string password, string confirmPassword,
string email)
        {
            this.username = username;
            this.password = password;
            this.confirmPassword = confirmPassword;
            this.email = email;
        }

        //Username validator to ensure apropiate length andigit count
        public bool validateUsername()
        {
            if (!String.IsNullOrEmpty(username) && username.Length > 8 &&
username.Length < 15)
            {
                int countDigits = 0;
                for (int i= 0;i < username.Length; i++)
```

```
                {
                    if (char.IsDigit(username[i]))
                    {
                        countDigits++;
                    }
                }

                if (countDigits >= 3)
                {
                    return true;
                }
            }
            return false;

        }

        //Password validation method to ensure password complexity
        public bool validatePassword()
        {
            bool validPassword = false;

            if (!String.IsNullOrEmpty(password) && password.Length >= 9)
            {
                int countDigits = 0, countUpper = 0, countPunctuation = 0,
countWhiteSpace = 0;

                for (int i = 0; i < password.Length; i++)
                {

                    if (Char.IsDigit(password[i]))
                    {
                        countDigits++;
                    }

                    if (Char.IsUpper(password[i]))
                    {
                        countUpper++;
                    }

                    if (Char.IsPunctuation(password[i]))
                    {
                        countPunctuation++;
                    }

                    if (Char.IsWhiteSpace(password[i]))
                    {
                        countWhiteSpace++;
                    }
                }
                if (countPunctuation >= 2 && countUpper >= 2 && countDigits >= 2 &&
countWhiteSpace == 0 && password == confirmPassword)
                {
                    validPassword = true;
                }
            }

            else
            {
                validPassword = false;
            }

            return validPassword;
```

```
        }

        //Email validation to ensure user provides a valid email adress
        public bool validateEmail()
        {
            if (!String.IsNullOrEmpty(email))
            {
                int countSymbol = 0;
                for (int i = 0; i < email.Length; i++)
                {
                    if (email[i] == '@')
                    {
                        countSymbol++;
                    }
                }

                if (countSymbol == 1)
                {
                    return true;
                }
            }
            return false;
        }

        //Method to validate fields on login form
        public bool validateLogin()
        {
            if(!String.IsNullOrEmpty(username) && !String.IsNullOrEmpty(password))
            {
                return true;
            }
            return false;
        }

        //Method to check if the login creditionals passed in by the user are admin
credentials
        public bool validateAdmin()
        {

            if(username == adminUsername && password == adminPassword)
            {
                return true;
            }
            return false;
        }
    }
}
```

Key Algorithms

1. Validation.cs ~ validatePassword()

There are many complex algorithms that form the basis of the ‘Guess that Phrase’ application. Starting from the outset, with the login/register feature there is a dedicated Validation.cs class that authorises accounts to be created and users to be logged in. Specifically, the validatePassword() method uses layers of robust string handling to ensure the user has the most secure password possible. This method cycles through each character in the password string and counts a number of string characteristics, such as, digits, upper case characters, punctuation and white space. This method is also multi-functional as it validates that the user has not mistyped their password by checking that the ‘Confirm Password’ field on the registration form matches the original password entered.

Validation.cs ~ validatePassword()

```
    //Password validation method to ensure password complexity
        public bool validatePassword()
        {
            bool validPassword = false;

            if (!String.IsNullOrEmpty(password) && password.Length >= 9)
            {
                int countDigits = 0, countUpper = 0, countPunctuation = 0, countWhiteSpace =
0;

                for (int i = 0; i < password.Length; i++)
                {

                    if (Char.IsDigit(password[i]))
                    {
                        countDigits++;
                    }

                    if (Char.IsUpper(password[i]))
                    {
                        countUpper++;
                    }

                    if (Char.IsPunctuation(password[i]))
                    {
                        countPunctuation++;
                    }

                    if (Char.IsWhiteSpace(password[i]))
                    {
                        countWhiteSpace++;
                    }
                }
                if (countPunctuation >= 2 && countUpper >= 2 && countDigits >= 2 &&
countWhiteSpace == 0 && password == confirmPassword)
                {
                    validPassword = true;
                }
            }

            else
            {
                validPassword = false;
            }

            return validPassword;


        }
```

2. Email Notifications

To make ‘Guess that Phrase’ as similar to a modern commercial application, I have implemented email notifications throughout different stages of the system. Firstly, on the login form, when a user has forgotten their password, they have the ability to request a password reset from an administrator (Figure i). Similarly, when an administrator updates a user’s password (via the admin page), the user will receive an email to the current email address saved to their user object in the database with their updated password (Figure ii).

Figure i

Welcome.cs ~ forgottenPasswordSubmitLBL_Click()

```
private void forgottenPasswordSubmitLBL_Click(object sender, EventArgs e)
        {
            string userRequest = "";
            if (forgottenPasswordTB.Text != "")
            {
                userRequest = forgottenPasswordTB.Text;
                try
                {
                    SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

                    //Authentication - ensures the program has a valid email to send from
                    NetworkCredential cred = new
NetworkCredential("GuessThatPhrase@gmail.com", "boekdvvnrbyspxye");

                    MailMessage msg = new MailMessage();

                    msg.From = new MailAddress("GuessThatPhrase@gmail.com");

                    msg.To.Add("GuessThatPhrase@gmail.com");

                    msg.Subject = "Forgotten Password Notification";

                    msg.Body = "Please update the following user's password: " + userRequest;

                    client.UseDefaultCredentials = false;

                    client.Credentials = cred;

                    //enabling SSL (Secure sockets layer, encryption) this is required by most
email providers to send mail
                    client.EnableSsl = true;

                    client.Send(msg);

                    MessageBox.Show("Forgotten Password Request Received. You will be
contacted within 24 hours");

                    forgottenPasswordSubmitLBL.Visible = false;
                    forgottenPasswordTB.Visible = false;
                    forgottenPassLbl1.Visible = false;
                    label1.Visible = false;
                }

                catch
                {
                    MessageBox.Show("Something went wrong..");
                }
            }
            else
            {
                MessageBox.Show("Please submit a valid request!");
            }
        }
```

Figure ii

AdminPage.cs ~ passwordSubmit_Click()

```
      private void passwordSubmit_Click(object sender, EventArgs e)
        {
            string searchTerm = oldUsernamelbl.Text;
            string newValue = newPasswordtb.Text;
            if(currentAdmin.updateUser(searchTerm, 1, newValue))
            {
                //send email to current user that their password has been changed
                try
                {
                    SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

                    //Authentication - ensures the program has a valid email to send from
                    NetworkCredential cred = new
NetworkCredential("GuessThatPhrase@gmail.com", "boekdvvnrbyspxye");

                    MailMessage msg = new MailMessage();

                    msg.From = new MailAddress("GuessThatPhrase@gmail.com");

                    msg.To.Add(oldEmaillbl.Text);

                    msg.Subject = "Password Reset Notification";

                    msg.Body = "Your password has been updated to " + newPasswordtb.Text + "
please login to your account!";

                    client.UseDefaultCredentials = false;

                    client.Credentials = cred;

                    //enabling SSL (Secure sockets layer, encryption) this is required by most
email providers to send mail
                    client.EnableSsl = true;

                    client.Send(msg);

                    MessageBox.Show("Success, user has been notified of password update!");
                    this.Hide();
                    AdminPage ap = new AdminPage(currentAdmin);
                    ap.Show();
                }

                catch
                {
                    MessageBox.Show("Something went wrong..");
                }
            }
            else
            {
                MessageBox.Show("Error");
            }

        }
```

Both of these code snippets utilise the System.Net and System.Net.Mail libraries to set up a SmtpClient with authorised network credentials. To keep this as professional as possible, I set up a GuessThatPhrase@gmail.com email account to monitor emails received from the application. To bypass the error I discussed in my testing, where the application (being a third party system) did not have access to Google Mail, I researched and found that I had to create an Application Password (‘boekdvvnrbyspxye’) and input this into the network credentials in my code to create the link. From here, a new MailMessage object is created with the sender and receiver emails along with the message subject and body. SSL is then enabled on the client to meet the encryption standards that are required by most email providers. Similarly, Error handling has been incorporated into these sections to catch any exceptions thrown if for some reason the Google Mail server is unavailable.

3. Database ~ overloaded constructors and editRecord(), getLeaderBoardInfo()

The database.csv is the sea of data that the application subsists of. In the same way, the DataBase.cs file is the control centre to reroute this data to the appropriate locations. For simple functions, such as, adding a user(addRecord()), the parameterised constructor can be used to pass in user information directly to database, meaning the method itself only requires one parameter in its method signature (the filepath). For more complex methods such as editing a record, unique parameters are required that may differ between each method call, for example, the position of the value to be edited and its replacement. For this reason, a default constructor was introduced to allow the application of both methods in one class.

DataBase.csv ~ Default and Parameterised Constructors

```
//Default and parameratised DataBase constructor
        public DataBase()
        {

        }
        public DataBase(string record1, string record2, string record3, string record4, int
record5)
        {
            this.usernameRecord = record1;
            this.passwordRecord = record2;
            this.emailRecord = record3;
            this.avatarRecord = record4;
            this.highScoreRecord = record5;

        }
```

The editRecord() method allows the most efficient updates to fields via data transfer between Excel CSV files, through the creation of a temporary file. This method takes

in a searchTerm (the username of the current user object being edited), the position of the field being edited (e.g. high score is at position 4) and the new value. The original database.csv excel file is cycled through and copies each user and their details across to a temporary file, each time checking for the searchTerm and updating this user object accordingly. This is done through the reusable readRecord() method which checks if a user exists in the database. Once this is established, a switch statement is used to edit the field with the new value at the correct position. Once all users, and the updated user are in the temp file, the original database.csv file is deleted, the temporary file is renamed to be ‘database.csv’ and the update is complete. There is also error handling around this method in the form of a try, catch, throw.

DataBase.csv ~ editRecord()

```
    public void editRecord(string searchTerm, int positionOfNewValue, string newValue)
        {
            string tempFile = "temp.csv";
            //if positionOfNewValue = 5 that triggers the delete user and all fields are
updated to empty

            try
            {
                string[] lines = File.ReadAllLines(Filepath);

                for (int i = 0; i < lines.Length; i++)
                {
                    string[] fields = lines[i].Split(',');

                    if (i < lines.Length)
                    {
                        //add record manually
                        string[] result = readRecord(fields[0], 0, tempFile);
                        if (result[0] == "Record not found")
                        {
                            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(tempFile, true))
                            {
                                if (fields[0] != searchTerm)
                                {
                                    file.WriteLine(fields[0] + "," + fields[1] + "," +
fields[2] + "," + fields[3] + "," + fields[4]);
                                }
                                else
                                {
                                    //Find the position of the new value
                                    switch (positionOfNewValue)
                                    {
                                        case 1: file.WriteLine(fields[0] + "," + newValue +
"," + fields[2] + "," + fields[3] + "," + fields[4]);
                                            break;
                                        case 2: file.WriteLine(fields[0] + "," + fields[1] +
"," + newValue + "," + fields[3] + "," + fields[4]);
                                            break;
                                        case 3: file.WriteLine(fields[0] + "," + fields[1] +
"," + fields[2] + "," + newValue + "," + fields[4]);
                                            break;
                                        case 4: file.WriteLine(fields[0] + "," + fields[1] +
"," + fields[2] + "," + fields[3] + "," + newValue);
                                            break;
                                        case 5: file.WriteLine("" + "," + "" + "," + "" + ","
+ "" + "," + "");
                                            break;
                                        default: file.WriteLine(fields[0] + "," + fields[1] +
"," + fields[2] + "," + fields[3] + "," + fields[4]);
                                            break;
                                    }

                                }
                            }
                            Console.WriteLine("Edited");
                        }
                    }

                    File.Delete(Filepath);

                    File.Move(tempFile, Filepath);

                    Console.WriteLine("Reccord edited");
                }
                catch (Exception ex2)
                {
                    Console.WriteLine("This program did not work");
                    throw new ApplicationException("This program did not work :", ex2);
                }

        }
```

The getLeaderboardInfo() method is key as it creates a List of my customised User object, sorts this list by high score using the IOrderedEnumerable extension and limits it to the top 3 users with the highest score. This method has appropriate error handling, in the form of a try, catch, throw.

DataBase.csv ~ getLeaderboardInfo()

```
}
        //Bring username and high score into a list to sort and return to leader board
        public List<User> getLeaderboardInfo()
        {
            int count = 0;
            List<User> users = new List<User>();
            //Search the csv file
            try
            {
                string[] lines = System.IO.File.ReadAllLines("database.csv");


                for (int i = 0; i < lines.Length; i++)
                {
                    if(lines[i] != ",,,,")//neede due to deleted user method not deleting
commas(intented)
                    {
                        string[] fields = lines[i].Split(',');
                        User user = new User(fields[0], fields[1], fields[2]);
                        user.setHighScore(Convert.ToInt32(fields[4]));
                        users.Add(user);
                    }

                }
                //sort the user list by descending
                var sortedUsers = users.OrderByDescending(u => u.getHighScore());
                List<User> highestScoringUsers = new List<User>(3);

                //only bring back top 3
                for(int i = 0; i<sortedUsers.Count(); i++)
                {
                    count++;
                    if (count <= 3)
                    {
                        highestScoringUsers.Add(sortedUsers.ElementAt(i));
                    }
                    else
                    {
                        break;
                    }
                }

                return highestScoringUsers;
            }
            catch (Exception ex1)
            {
                return users;
                throw new ApplicationException("This program did not work :", ex1);
            }

        }
```

4. Game Hard Timer

This method controls the countdown for the hard level in ‘Guess That Phrase,’ where the user is awarded with 5 more seconds to the timer every time they get a question correct. This is illustrated by a clock image which changes location to correspond to time remaining. The location is established by creating a new instance of the Point object with an x and y co-ordinate. It also, responds to the user’s pause request (i.e. the time is frozen). Once the clock reaches the final position and the user’s time has run out, it invokes the game over sequence. This timer has threshold validation, in that the clock will never increment off of the screen.

GameHard.cs ~ timer4_Tick_1()

```
 private void timer4_Tick_1(object sender, EventArgs e)
        {
            int step = -1;
            int xMax = 790;
            this.GameHardClockLBL.Location = new Point(this.GameHardClockLBL.Location.X +
step, this.GameHardClockLBL.Location.Y);

            GameHardScoreDisplayTB.Visible = true;

            //Pause game
            if (GameHardInfoLBL.Visible == true)
            {
                GameHardScoreDisplayTB.Visible = false;
                timer4.Stop();
            }
            //timer method that increments timer and decrements timer
            if (correct == true && GameHardClockLBL.Location.X + 100 <= xMax)//xmx stops clock
from moving off screen
            {
                //x max is 790
                this.GameHardClockLBL.Location = new Point(this.GameHardClockLBL.Location.X +
100, this.GameHardClockLBL.Location.Y);
                correct = false;
            }
            //game over sequence
            if (GameHardClockLBL.Location.X == 130)
            {
                int finalHighScore = 0;
                timer4.Stop();
                GameHardGameOverLBL.Visible = true;
                GameHardGameOverScoreLBL.Visible = true;
                GameHardGameOverMainMenuLBL.Visible = true;
                GameHardGameOverPlayAgainLBL.Visible = true;
                GameHardGameOverPlayAgainLBL.BringToFront();
                GameHardGameOverMainMenuLBL.BringToFront();
                GameHardGameOverScoreLBL.BringToFront();
                gameHardqMarklbl.Enabled = false;
                GameHardGameOverScoreLBL.Text = GameHardScoreDisplayTB.Text;
                string score = GameHardScoreDisplayTB.Text;
                //final score written to database
                //validation to ensure high score is higher than current score
                if (score.Equals(""))
                {
                    finalHighScore = 0;
                }
                else
                {
                    finalHighScore = Convert.ToInt32(score);
                }
                GameHardGameOverScoreLBL.Text = finalHighScore.ToString();
                currentUser.setHighScore(finalHighScore);
                DataBase db = new DataBase(currentUser.getUsername(),
currentUser.getPassword(), currentUser.getEmail(), currentUser.getAvatar(),
currentUser.getHighScore());
                db.updateHighScore(currentUser.getUsername(), 0, finalHighScore);
            }
        }
```

5. Bonus Level (Drag and Drop)

The Bonus level contains 2 dictionaries: one to hold the Label (key) and the string identifier for that label (Value). For example the label containing the dog emoji and the string identifier “dog,”. The second dictionary holds the question phrase (key) and the concatenated string made up of the substrings from the values in the first dictionary (value). For example, (`"Enfermo como un perro (Sick as a dog)", "sick,point,dog,"`). The first complex code snippet is the MouseDown function applied to each target label. This creates a new Bitmap object with the same characteristics of the target label’s image. This bitmap is then casted to the cursor to imitate a drag effect. This offers a new form of game play to the user giving them greater visual cues and higher user experience.

BonusGames.cs ~ DragLbl1_MouseDown()

```
private void DragLbl1_MouseDown(object sender, MouseEventArgs e)
      {
          Bitmap bmp = new Bitmap(DragLbl1.Width, DragLbl1.Height);
          DragLbl1.DrawToBitmap(bmp, new Rectangle(Point.Empty, bmp.Size));
          bmp.MakeTransparent(Color.White);

          Cursor cur = new Cursor(bmp.GetHicon());
          Cursor.Current = cur;

          imageDragged = 1;
      }
```

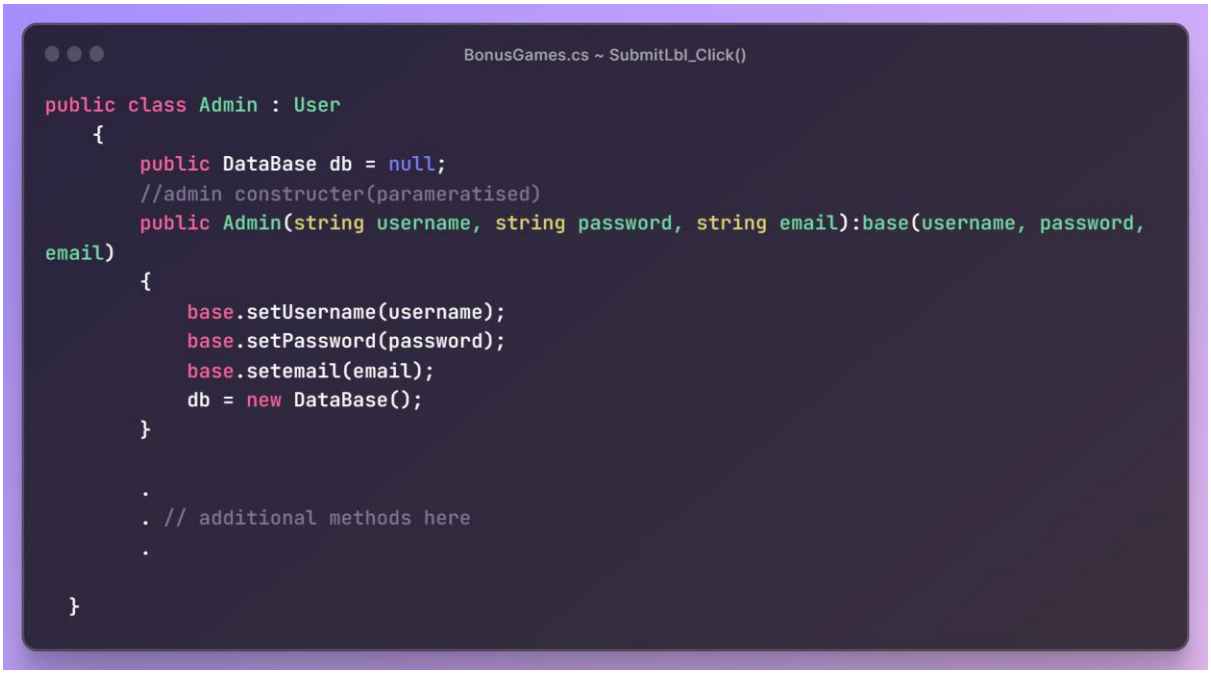
The next complex algorithm within the bonus level is the event that occurs when the answer is submitted (SubmitLbl_Click()). This method first checks that all three of the picture boxes have been populated with an image. When an emoji label is dragged into one of the picture boxes the tag of the picture box is updated to the name of the target emoji label. Due to this, each picture box tag is then checked against the dictionary that holds the emoji labels to find the current emoji label inputted into the current picture box. This is repeated for each of the 3 picture boxes, each having their own for loop to check this. Once the label in each picture box is identified the string value in the dictionary that corresponds to that label (Key in the dictionary) is added to a current string variable. This process builds up a string concatenation based on the 3 emoji labels that the user has entered into the 3 picture boxes, for example, if they entered the bed emoji into picture box 1, the nose emoji into picture box 2 and the rose emoji into picture box 3, the string would be “bed, nose, rose.” This string is then checked against the value of the second dictionary to see if it matches the answer for the current question (key in the second dictionary). If so, the tick label will become visible, indicating the user’s answer was correct. On the other hand, the cross label will show to indicate they have given the wrong answer.

BonusGames.cs ~ SubmitLbl_Click()

```
private void SubmitLbl_Click(object sender, EventArgs e)
        {
            string currentQuestion = QuestionLbl.Text;
            bool correct = false;
            //check if all picture boxes contain an image
            if (picBox1ImageChanged && picBox2ImageChanged && picBox3ImageChanged)
            {
                //iterate through dictionary imagelbl dictionary to match image to string
                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer1.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }

                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer2.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }

                for (int i = 0; i < imageLBLDICT.Keys.Count(); i++)
                {
                    if (pbAnswer3.Tag == imageLBLDICT.Keys.ElementAt(i).Name)
                    {
                        imageDetails += imageLBLDICT.Values.ElementAt(i);

                    }
                }
            }

            for (int i = 0; i < myQuestionDICT.Count(); i++)
            {
                if (myQuestionDICT.Values.ElementAt(i) == imageDetails)
                {
                    if (currentQuestion == myQuestionDICT.Keys.ElementAt(i))
                    {
                        correct = true;
                    }
                }

            }

            if (correct)
            {
                CorrectLbl.Visible = true;
                IncorrectLbl.Visible = false;
            }
            else
            {
                CorrectLbl.Visible = false;
                IncorrectLbl.Visible = true;
            }


            num = rand.Next(0, myQuestionDICT.Count());
            QuestionLbl.Text = myQuestionDICT.Keys.ElementAt(num);

            pbAnswer1.Image = null;
            pbAnswer2.Image = null;
            pbAnswer3.Image = null;
            imageDetails = "";
        }
```

6. Admin, User Inheritance

The Guess That Phrase Application has a User object class which defines the unique characteristics and behaviours of a user. Since an administrator is a type of user, the Admin class inherits this functionality to its own class and extends its unique functions. The administrator requires a username, password and email of which it passes to the base class for initialisation. It also requires a database object to deal with alterations to users of the game.



Evaluation

I thoroughly enjoyed creating the ‘Guess That Phrase’ application and watching a small idea develop into a fully-fledged modern day system. I believe I have implemented, designed and documented each component to the best of my ability given the time frame and I will reflect on this process in this section.

1. Evaluation of User Requirements

User Requirement	Achieved/Not Achieved	Comment
FUNCTIONAL REQUIREMENTS		
UR1 The application must have a splashscreen	ACHIEVED	The application has an initial splash screen that establishes the ‘Guess That Phrase’ logo and brand from the outset of the application. It also has a progress bar which gives a visual indication of the application loading.
UR2 The application must have a login and register screen	ACHIEVED	The application has a login page which allows the user to login with an existing username and password. It also has a password eye for enhanced security and ease of use. The register form allows a user to create a new account with advice given on the requirements of each field available in help labels.
UR3 The application must allow the user to register as new user	ACHIEVED	Once a user has successfully inputted their information to the register form their information is stored in the database.
UR4 The application must have password validation with a strength check and visual indication of strength e.g. strength bar	NOT - ACHIEVED	The application does have thorough password checking and validation. It also outlines a guideline for how strong the password has to be to register an account. For example, a certain length, containing punctuation and uppercase

		characters. However, a visual indication was not displayed as I believed it would clutter the register form and be an eye sore.
UR5 The application must allow the new user to choose an avatar, UserName and Password	ACHIEVED	A user can choose from 3 different avatars on the register page they can also choose their own username as long as it is unique and is not already taken.
UR6 The application must allow a user to login with a saved username and password	ACHIEVED	Once a user has successfully registered an account they can use their existing details to login to the system. If they have forgotten their password there is the option to request a password reset from the administrator.
UR7 The application must allow an admin account to login	ACHIEVED	An admin account credentials have been created in the system and an admin can successfully login to the application.
UR8 The application must have a main menu screen	ACHIEVED	The application has a main menu screen which is a crossroads that allows access to different avenues of the application. These include, the ‘play game’ forms, with access to the easy, bonus and hard levels of the game, the options screen, the leader board and the ability to logout/exit the main menu and return to the login page.
UR9 The application must allow an admin to change account details of users	ACHIEVED	Once an admin is successfully logged into the system using the stored credentials they are brought to the admin page where they can select a user from the

		database and change their password, email, avatar or high score.
UR10 The application must allow an admin user to delete accounts	ACHIEVED	Once an admin is successfully logged into the system using the stored credentials they are brought to the admin page where they can select a user and delete their details from the system.
UR11 The application must have multiple levels of difficulty (Easy, Medium, Hard) for the game with appropriate information provided	NOT - ACHIEVED	The application has an easy and hard level of the game. A medium difficulty level was not implemented due to time constraints and performance concerns. However, a bonus level was implemented which showcase a different form of question type (drag and drop).
UR12 The application must show if the users answer was correct or incorrect	ACHIEVED	On each level of the application a tick label / cross label is displayed when the user inputs a correct/incorrect answer to the question.
UR13 The application must record the users score and other information in a database csv file	ACHIEVED	The application has a database.csv file which stores each user object. This information is split into username, password, email, avatar and high score fields. There are multiple methods implemented to add, edit or delete records from this file.
UR14 The application must have a leader board to compare users score	ACHIEVED	The application has a leader board page which showcases the top three highest ranking users and their respective scores achieved from the hard level of the game.

UR15 The application must have a variety of questions to be asked	ACHIEVED	The easy level of the game contains a dictionary of 50 possible questions that can be asked repeatedly over the given timeframe. The hard level has an additional 50 questions which again can be repeated over the time frame. The bonus level has 8 possible questions that can be asked with no time constraint. Therefore, in total there are 108 unique questions throughout the entirety of the application.
UR16 The application must time the user until the questions are finished	NOT - ACHIEVED	Originally, the score would be based on the users time that they took to complete the questions and the time would increment and finish when they had answered all questions. However, upon reflection I thought the game would be more addictive and fast paced with a timer countdown and unlimited repeating questions.
UR17 The application must be optimised and run well on school grade PCs	ACHIEVED	The application is optimised as much as possible with the constraints of Windows Forms. By rendering my images in Pixlr.com I was able to omit a large amount of components, only adding controls when absolutely necessary. Similarly, in the options page of the game a user has the ability to turn of advanced design settings such as gifs which optimises performance.

UR18 The application must allow a user to log out	ACHIEVED	A user can exit the game from the main menu which brings them back to the login screen therefore, logging their account out of the game.
NON FUNCTIONAL REQUIREMENTS		
UR19 The application must have appealing imagery that fits into the emoji theme throughout.	ACHIEVED	The use of the paint technique in Pixlr.com, the online image designer allowed me to use fonts and colours outside of what Windows Forms provides. Similarly, finding high resolution images of emojis from emojipedia.com and resizing these to fit the forms' dimensions ensured the imagery was engaging and on brand.
UR20 The application must have engaging headings that fit each section of the game with easily readable font.	ACHIEVED	Each heading in the application was clear and corresponded directly to what the form was about. All fonts were readable and sized proportionally.
UR21 The questions must be universal and fit the difficulty level, allowing for speed of entry to make the game addictive and encourage the user to play again.	ACHIEVED	The questions are universal as they are visual and don't have a language. In the future, there would be an option to translate the user input so it could me multilingual. I have made it more inclusive and educational by adding the Spanish element into the bonus level. The easy and hard levels are separated by using 2-3 emoji phrases in the easy level and 3-4 emoji phrases in the hard level.
UR22 The application must be appealing to attract more users to the game to in turn, add more users to the leader board rankings.	ACHIEVED	Overall, the application is appealing in multiple visual aspects and the ability to hit the enter key on the keyboard when submitting questions

		makes the game more fast paced and addictive for users to return to and compete against each other.

2. Evaluation of Time Management

Overall, I believe I managed my time effectively to produce a well-rounded application. Initially, I perhaps spent a lot of time gathering the images of the emojis for each question when this could have been avoided by having a smaller and more simple question set. I could also have employed multiple time management techniques, such as, the Pomodoro technique. This would have been useful as I could have timeboxed each design section to a limit of 25 minutes to ensure each part of the project had equal planning. The best form of time management would be iterations of this technique which would limit external distractions and focus time on developing key areas of the application in depth. I believe I spent more time coding than on my initial storyboarding. Although this allowed me to achieve a large project, it may have limited my ideas for design and caused delays when trying to implement specific features as I had to go back and rethink design options. Personally, I enjoy seeing a section of the application out to completion in one sitting. Therefore, I coded for various blocks of time. With some sections being longer than others, this may have been an ineffective time management technique but it allowed me to better understand my application’s development process.

3. Evaluation of Testing Procedures and the Results Obtained

I employed 3 different testing methodologies: unit testing, link testing and user acceptance testing. This ensured thorough testing of the application from both a programmer’s and user’s perspective. I conducted unit testing after each section of the application was implemented. Once the whole application was up and running, I performed linked testing to test the ease of navigation and access to the system. I then released it to users for acceptance testing. The test plan was very detailed but did not cover every possible way that the application could be used. For this reason, I performed extra tests where necessary to ensure each feature was fully robust. I believe it was better that I performed the testing myself as I have a deeper understanding of the application and possible errors that may arise than a third party could have had. As well as this, I could fix the problems in real time. Due to this approach, I believe I spent a significant amount of time testing and ensuring that the application is as error free as possible. I greatly included the client in my testing and thought of enhancing the user experience as much as possible in the design and implementation phases of the application’s lifecycle.

4. Evaluation of Project Management

Given that I achieved ~90% (19/22 user requirements were met) of what I set out to complete in the initial planning of the application, I believe I managed the project very well. It may be noted that this could have been improved by using tools such as a Gantt chart to graphically represent my progress. I believe my key factors of success are my strong drive for program completion, dedication to increasing user experience and strive to meet all user requirements. By implementing more structured project and time management techniques like the ones I have mentioned I would like to think I would have 100% met the user requirements. Given my experience with developing other applications in my free time I was able to employ strategies that I had learnt to streamline production.

5. Evaluation of Self Management

I felt I upheld myself well throughout this application development process. I worked efficiently and helped others by sharing the tips I learned from sources, such as, Stack Overflow. However, I did not lose focus from my own project and strived to complete it to the best of my ability. On reflection, I would have benefitted from doing online courses in C# and Windows Forms as it was not an application I was familiar with. This would have enabled me to ‘hit the ground running’ and not face any unforeseen setbacks with the technology I was working with. I felt I did significant market research and still believe that ‘Guess That Phrase’ has a unique selling point as there are no other applications that use emojis as question types. I could have done more research into more robust database structures such as SQL Databases as opposed to a csv file.

6. Application Strengths and Weaknesses

I believe my application has multiple selling points from a coding perspective and end-user perspective. My in-depth user acceptance testing reflects the my strengths that the user’s enjoyed and some weaknesses that I improved upon or could add in future iterations of the application. The key selling points are as follows:

- A highly secure login and registration
- A robust and flexible database structure
- A unique and engaging screen design which is suitable for all audiences
- A large variety of challenging questions and educational opportunities
- A competitive leader board
- The ability for a user to alter their design preferences and receive more information via help labels

Some limitations to the game that could be improved upon are:

- A more efficient data storage system, such as a database, to accommodate for a large growth of users on the platform
- Additional question types / expansion packs of emojis

7. Possible Future Improvements

Given the knowledge, skills and understanding gained from developing this application and testing it with end-users I would improve the application in a few ways. Such as, making help labels more concise and readable – perhaps creating a user guide for the user’s to refer to detailed instructions, removing clutter from the forms. As mentioned previously, I would upgrade the database storage solution to ensure longevity of the application as the product matures. Similarly, I would investigate encryption of the database to protect client’s GDPR rights and ensure the data is as secure as possible. I would consider using ‘Unity’ as it also uses C# but has more opportunities for higher resolution graphics and design. I would consider deploying my application to the cloud to utilise the fast developing cloud computing technology and allow multiple users to inhabit the application at one time. This would enable the ability for users to compete against each other directly, in real time. As well as this, cloud deployment would be a step towards making this project cross-inhabitable between mobile app and desktop users.