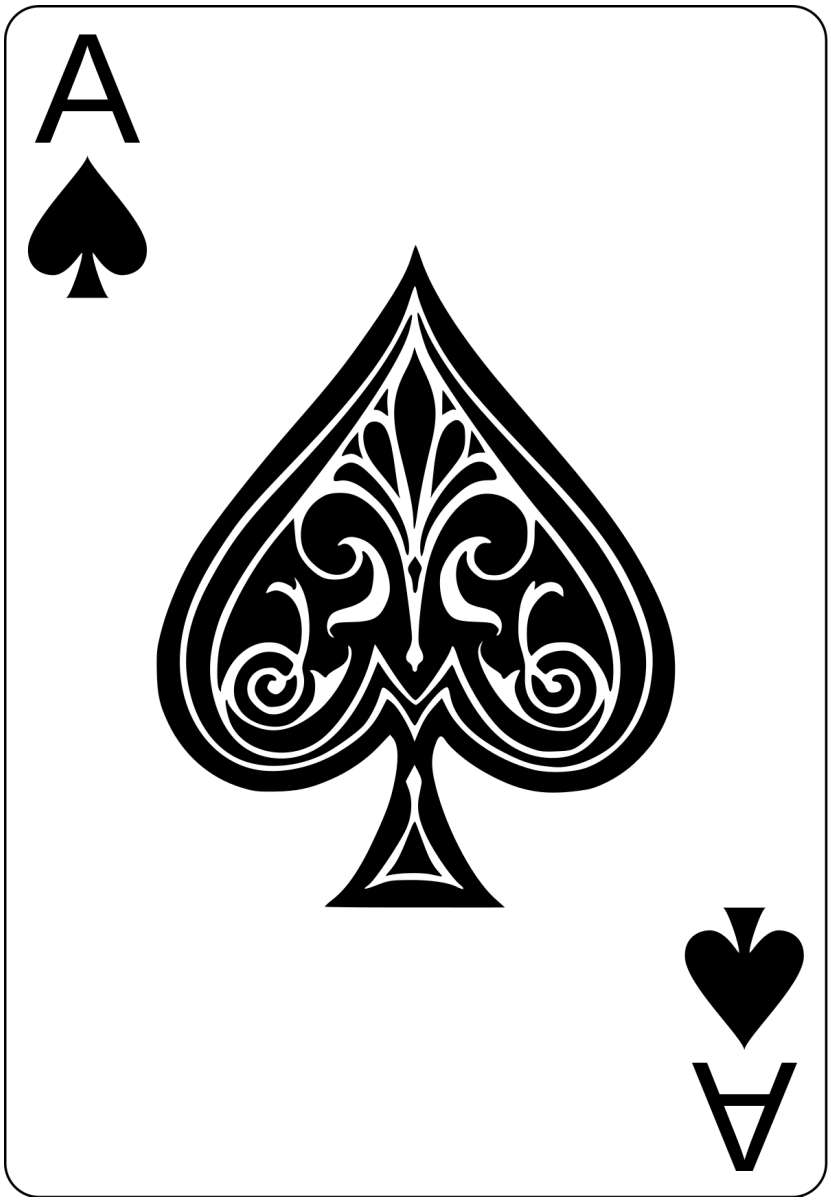
BlackJack

Advanced Higher Computing Science Project

By Abhishek Shadakshari



Candidate Checklist AH Project

SQA instruction document

Stage	Marks	Completed (Date)
Project Ideas		
Analysis	**10**	
Description of the problem	2	09/21
UML use case diagram	2	09/21
Requirements specification	4	09/21
Project plan	2	09/21
Design	**20**	
Project design	15	10/21
User-interface design	5	10/21
Implementation	**30**	
Implementation	21	03/22
Research and development of new skills and/or knowledge	4	02/22
Log of ongoing testing	5	02/22
Testing	**15**	
Final test plan	6	03/22
Requirements testing	6	03/22
Testing with personas and test cases	3	03/22

Evaluation	5	
Evaluation report	5	04/22

Project Diary

Keep this diary up to date with all the tasks for your project (good idea to update it at the start/end of every week). Keep a note of any **resources** you make use of or **errors** you encounter as you'll need a comprehensive list to complete the latter stages of your project.

Date	Task worked on/errors encountered	Notes/Solution to errors/Resources used
12/9	Finished rough sketches/wire frame diagrams of the layout and the design of the forms and database. Includes the main screen with all the cards, pop ups asking the players what they would like to do in certain situations, and the database plan including the fields and the specifications for each field.	Draw.io
18/9	Finished a user survey that gets feedback on the design of the website while also having spaces for suggestions and ways to improve the website. Also using diagrams.net in order to create a more detailed design of the software which will help users answer questions from the user survey.	Google Forms didn't know what sort of questions to ask in a user survey.
24/9	User survey has been sent out to classmates with copies of my high fidelity design and I have received feedback on how to improve the design and the program as a whole.	Google Forms, Peers
30/9	Started setting up a gantt chart and setting up dates for starting and completion of different aspects of the	Google Sheets

	design, implementation and testing so that I have a solid idea of what is going to be happening when	
2/9	Finished exercises for structured data types and have made sure that I understand all of them so I will be able to use them for my project, I have decided that I will be using an Array of Records to hold the information about each account that is registered to the program and update the information in the database while the program is running	
3/9	Practising object oriented programming as it will be a key part for my project, the deck of cards will be a super class, with the suits being a subclass of that and each card being a subclass of their respective suits	
9/9	Did some practice uml Use Case tasks to understand what to do, then completed a Use case diagram for a project with the actors being the player, database and dealer(cpu) and completed the uml diagram.	
13/9	Continued setting up a gantt chart to give myself solid goals for when to have everything done for which makes it easier for me as I know when I'm going to have to complete each section of my project.	
20/9	Started pseudocode for main program, made it into sections to show how each aspect of the program will work	Google Docs
24/09	Finished Pseudocode, it is split into 6 sections; 1 - 5 being each button on the main menu that can be pressed, and 6 being the code for the actual game that is being played	Google docs, Was very hard to read even for myself so had to add indentations and spacing between different steps
20/10	Started designing user interface for	

	main menu, includes buttons to take the user to different pages, also added in text to the 'rules' page to explain all the needed rules to players who don't know them already	
27/10	Testing passing data to and from a database, is going smoothly with minimal problems, finished create account form with a syntax issue with microsoft access, solved problem by adding square brackets into the statement which helped	Visual Studio, Microsoft Access
3/11	Finished coding create account form, tested using breakpoints and is working fine, started coding sign in page and also frequently using breakpoints to check for errors and bugs	Visual studio
10/11	Finished coding sign in form, is available to use when checking account details but not when playing the actual game yet as the game is yet to be programmed. Account details form has been completed as well and is able to display the username, account balance, and games played of each player/account that has been registered.	
11/11	Found a group of cards to use in the program, had to crop each individually into separate images and then import them each into a separate picture box in visual basic, adding these into the game form and when the form is open getting all of them to hide when the form is opened was the next step. Changed the max amount of players down to 5 due to the constraints of the school computers.	
22/11	With the game only being played on one screen each player will need to	

	login on the same screen, I have implemented this by including a log in section at the start of the game that takes place after the number of players has been decided. Once one player signs in with the correct credentials it will add their details into an array of records that will be used throughout the game, after this the textboxes will be cleared and another player will sign in. This will repeat until the number of players that were selected have successfully signed into the game. Usage of global variables is quite high.	
24/11	Started programming actual game, using 2D array data structure to hold values for each coordinate for where the cards should be displayed when they are dealt to players and dealer, continued coding game, mostly working on the user interface at the moment.	
6/12	Finished inserting card images and other imports into the game, starting to code the procedures and functions including the options each player gets during, taking advantage of global variables to pass data between two different forms	
13/12	Continuing coding procedures for main game of project, creating a separate form in order to let players choose options during a round (hit, stay, double down) to get more cards or stop playing this round, using a procedure found online to shuffle the deck of cards, it is called **The Standard Fisher-Yates Shuffle.**	Unable to make aces values interchangeable (since it should be able to be used as 1 or 11) haven't found a way to implement this
01/02	Finished prelim, haven't looked at project in a while, doing some testing in order to get back up to date and understand how each section of my	

	project works, added	
07/02	Project didn't meet Ah standards (I had enough data structures and a connection to a database) but there were no standard algorithms. So I implemented a leaderboard in order to add a bubble sort to the project, it displays the ranks, usernames, and balances of the players in order of their balance (highest to lowest).	
09/02	Code hadn't been commented throughout the program, going through now and commenting each section, procedure and function.	
23/02	Starting test plan, with general outline of the testing to be carried out to confirm the fitness for purpose in terms of the End-User and Functional Requirements	
24/02	Finished test plan in terms of requirement testing, and have started carrying out tests to check if each requirement has been fulfilled.	
16/03	Have been carrying out requirement tests (was isolating for a week), and have just completed them, moving on to creating test cases.	
18/03	Test cases have been created, there are 3 which altogether test every major aspect of the program.	
21/03	Test cases have been completed, the test results are ideal and show that the program is usable for the majority of end-users	
23/03	Taking screenshots of the final code and user interface and entering it into the project file	
24/03	Screenshots of User Interface and code have been put into the	

	document, moving on to the evaluation after checking everything is done to a desirable degree.	
18/04	Starting evaluation, commenting on different aspects of the program as well as the testing. Evaluation completed, mentioned each type of testing as well as maintainability	

Analysis of the problem (10 marks)

Before you begin designing and developing a solution, you must analyse the problem that you are going to solve, to ensure that you fully understand every aspect of it.

Description of the problem (2 marks)

Describe your problem. Your description should include:

- an outline of the problem, identifying the Advanced Higher **concepts** and **integration**

The project is a Blackjack game and will allow users to create an account to keep track of how much money they make and how many games they've played. The program will include an array of records to store the data of each account that has been registered on the program, some examples of the fields would be - username, Password, accountBalance, gamesPlayed. This array of records will be updated as the different users interact with the program and the information will be stored in a database. As the program is a card game, the cards will be saved as objects, every card object will be created out of the card class apart from the aces, which will be done as a subclass as it needs extra properties and methods due to the fact it can have 2 values. Each object will be stored in a 1D array which will be the 'deck' of cards and will be randomised at the start of every round. A bubble sort will be used to sort the playerdata into order of account balance from highest to lowest in order to have a leaderboard of all the registered players.

- the **scope** and **boundaries** of the problem, and any **constraints** you identify

Scope
The scope of the program is a single card game (Black Jack) and will use fake currency to place bets on the game while playing, players will be able to create accounts in order to keep track of their progress and the money they win or lose while playing the game, the game will be played on a single screen with each player having a pop up input in order to input the values they need to into the program.

Boundaries
The boundaries of the program are that there will be a max player count of 5 in a game , only one game can be played at one time and we'll be using fake currency.

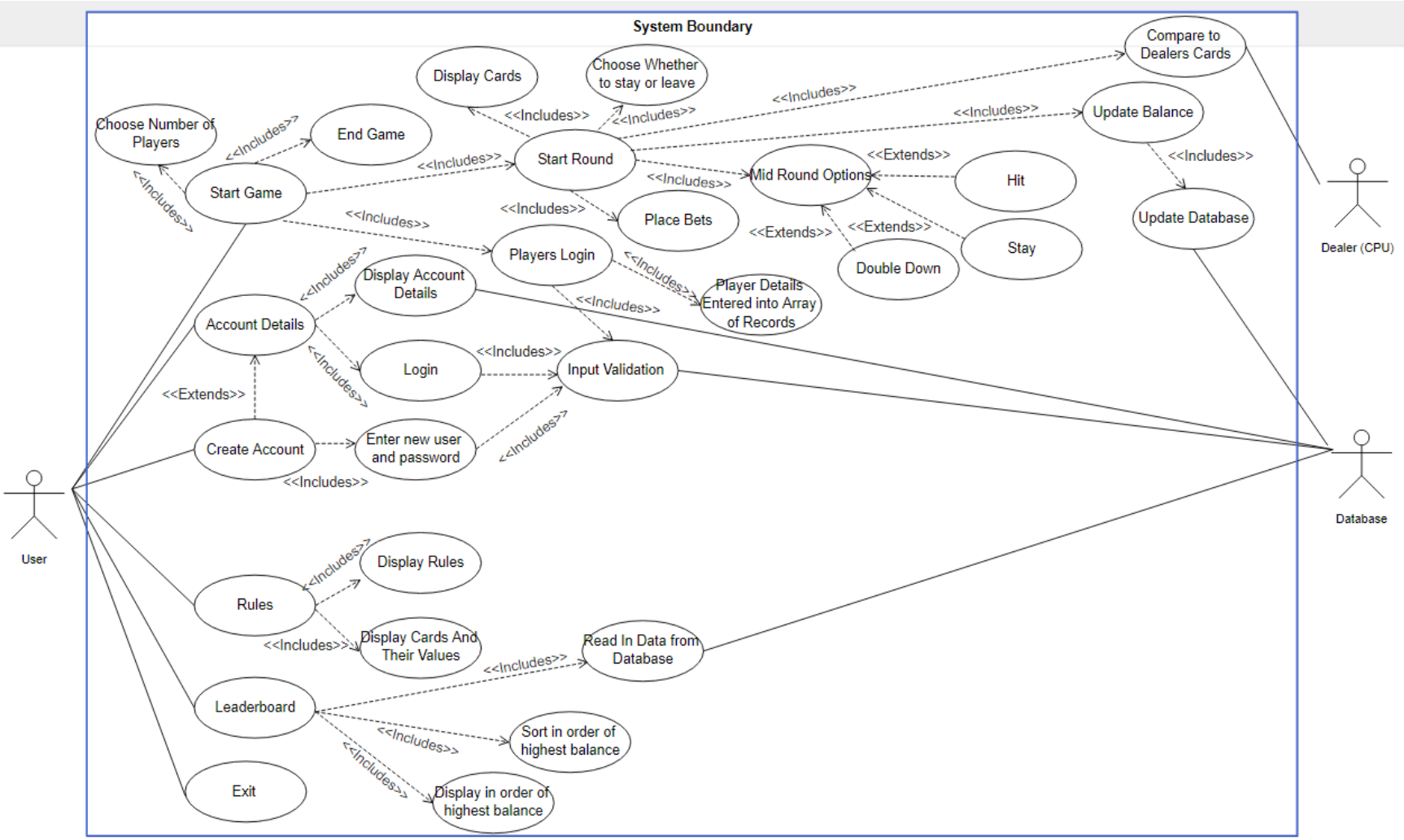
Constraints
The constraints for these are that the school computers aren't powerful enough to cope with over 5 players sending data back and forth from a database and multiple games sending data to and from a database, the constraint on currency is that I do not own a betting licence and also cannot implement code on vb that connects to a server to exchange real money, this constraint also applies to why the game is played on one screen and not across multiple online, as it is not possible for me to implement server side code that would allow online play.

UML use case diagram (2 marks)

Draw a UML use case diagram for your problem. Your diagram should define the following:

- **Actors**
- **use cases**
- **relationships**

Your diagram should include the **integration** you intend to implement, for example a web project connecting to a database.


System Boundary
Compare to Dealers Cards
Display Cards
Choose Whether to stay or leave
Choose Number of Players
End Game
Update Balance
Start Round
Mid Round Options
Hit
Start Game
Place Bets
Update Database
Dealer (CPU)
Players Login
Stay
Double Down
Display Account Details
Player Details Entered into Array of Records
Account Details
Login
Input Validation
Create Account
Enter new user and password
Database
User
Display Rules
Rules
Display Cards And Their Values
Read In Data from Database
Leaderboard
Sort in order of highest balance
Exit
Display in order of highest balance
<<Includes>>
<<Extends>>

Requirements specification (4 marks)

Produce a requirements specification. Your requirements specification should list:

- End-User Requirements

1. For players to be able to sign into their account and check the details of their account such as their account balance and games played.

2. For the program to give each player the option of whether to; Hit, Double Down or Stay during their turn of each round.

3. player should be able to bet up to £500 on each round and either lose what they bet, or win it back doubled.

4. Each user should have the option to check the rules of BlackJack from the main menu, showing a list of the rules and a list of each card, its graphic and what its value is.

5. Each user should be able to check a leaderboard which displays all the users with a rank depending on their balance the higher the balance the higher up on the leaderboard they are

- Functional Requirements

1. For the database to interact with the program after every round of the game, updating each player's information.

2. For the program to display a graphic of the cards that are dealt to the players and the dealer every turn.

3. Once a player has 'stayed' or has gone bust during that round the program should no longer show them a pop up window with each option that they could choose.

4. Once every player has 'stayed' or gone bust during that round, the program will cause the dealer's card that was face down to flip over and reveal itself, if the dealer's total is <=16 then the program will keep giving them cards until their total is >16

5. The program should read in values from a database and be able to validate whether a players username and password are correct

6. The program should allow an unregistered user to create an account, saving the details of the account into a database and automatically setting their balance to £5000

Your requirements specification should consider any input validation that may be required for your problem.

Project plan (2 marks)

Create a project plan for the four remaining stages of your project.

Your project plan should include:

- The tasks you complete in each stage
- any resources you need to implement your solution
- an estimate of how long each stage will take

Guidance for producing a project plan Tasks could be:

- User-interface design
- Refinement of coding
- ongoing testing

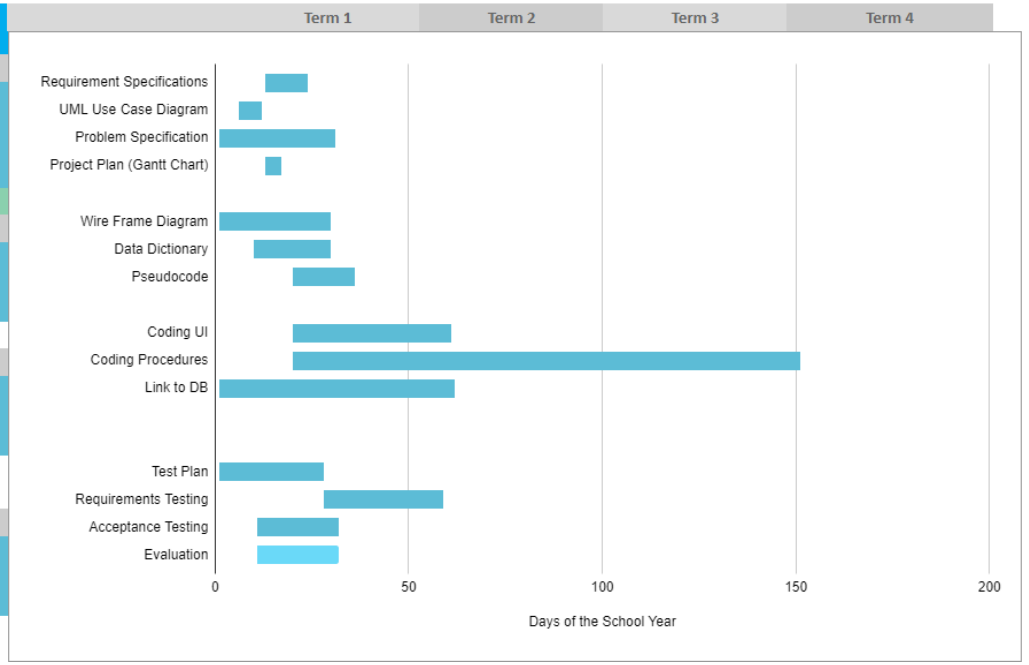
Resources can include access to development tools and end users. Some of these could be available at any time, while others may only be available at certain times. You need to plan to ensure that your project is not held up waiting for resources.

Your timings should allow for holidays, or other events that affect how much time you can spend on your project.

You should review and update your project plan as you work through each stage. **You must submit your final version of the plan as evidence.**

Project Plan - Gantt Chart/Resources Required

TASK NAME	START DATE	DAY OF MONTH*	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	PERCENT COMPLETE
Analysis							
Requirement Specifications	13-Sep-2021	13	24-Sep-2021	11	11	0	100%
UML Use Case Diagram	6-Sep-2021	6	12-Sep-2021	6	6	0	100%
Problem Specification	1-Aug-2021	1	31-Aug-2021	30	30	0	100%
Project Plan (Gantt Chart)	13-Sep-2021	13	17-Sep-2021	4	4	0	100%
		30		0			
Design							
Wire Frame Diagram	1-Aug-2021	1	30-Aug-2021	29	29	0	100%
Data Dictionary	10-Aug-2021	10	30-Aug-2021	20	20	0	100%
Pseudocode	20-Sep-2021	20	6-Oct-2021	16	16	0	100%
		30		0			
Implementation							
Coding UI	20-Oct-2021	20	30-Nov-2021	40	41	0	100%
Coding Procedures	20-Oct-2021	20	28-Feb-2022	128	131	0	100%
Link to DB	1-Dec-2021	1	31-Jan-2022	60	61	0	100%
		30		0			
		30		0			
Testing + Evaluation							
Test Plan	1-Feb-2022	1	28-Feb-2022	27	27	0	100%
Requirements Testing	28-Feb-2022	28	31-Mar-2022	30	31	0	100%
Acceptance Testing	11-Mar-2022	11	1-Apr-2022	20	21	0	100%
Evaluation	11-Mar-2022	11	1-Apr-2022	20	0	21	0%



Design of the solution (20 marks)

You must now design your solution based on your requirements specification.

The problem you are solving will determine the design methodologies you use. The following may be suitable:

Software design and development

- code design — top level design with data flow and refinements of Advanced Higher concepts (structure diagrams or pseudocode)
- UML class diagram — including class names, properties and data types, methods (including constructor) and arguments, and (where appropriate), public and/or private, inheritance

Database design and development

- entity relationship diagram — including (where appropriate) entity name, entity type (strong, weak), attributes, relationship participation (mandatory, optional), relationship name and cardinality
- data dictionary — using SQL attribute types
- query designs

Web design and development

- wireframes relating to media query
- low-fidelity prototypes of pages
- code design for PHP (structure diagrams or pseudocode)

Project design (15 marks)

Design your solution, using appropriate design methodologies or techniques.

Your design should meet the end-user and functional requirements identified at the analysis stage, and include the integrated part of the project.

DDD Design
Data Dictionary -

Fields	Data type	Required	Calculation	Limits
username	Text	Yes	No	Max 15Char
Password	text	Yes	No	Min 9Char
Number of Games played	Number	Yes	Yes	>=0
Account Balance	Currency	Yes	Yes	<100,000

SDD Design
Pseudocode -

Initial Setup of variables
0.1 Connect To Database
0.2 Initialise Array of records using database - userData(username - string, password - string , games played - integer , games won - integer, account balance - single)
0.3 Initialise Class 'Deck'
 0.3.1 Initialise Subclass 'Hearts' (Deck)
 0.3.2 Initialise Subclass 'Diamonds' (Deck)
 0.3.3 Initialise Subclass 'Clubs' (Deck)
 0.3.4 Initialise Subclass 'Spades' (Deck)
0.4 initialise Array of Records - players(name - string, balance - single, bust - boolean, in - boolean, playing - boolean, cardTotal - integer, cardValue as integer, bet - single)

1.1 StartGame_Click
 1.2 Display "Choose an amount of players from 1 - 5" . Drop down menu.1-5

```
 1.3.0 For counter = 1 to NumPlayers
  1.3.1 go_to_signIn.Form
  1.3.2 validate_inputs (userData compared to values in Database)
  1.3.3 Repeat until input is valid
  1.3.4 players(counter).name = userName Input
  1.3.5 players(counter).balance = userData(num).balance
  1.3.6 players(counter).bust = false
  1.3.7 players(counter).playing = true
  1.3.8 for counter = 1 to numPlayers
   1.3.8.1 players(counter).bet = 0
 1.3.8.2 Next
 1.3. Next
 1.4 playersPlaying = numPlayers
 1.5 go_to_game.Form
1.5 End Sub

2.1 CreateAccount_Click
 2.2 go_to_createAccount.Form
 2.2.1 Textbox.Input.("Username")
 2.2.2 Textbox.Input("Password")
 2.2.3 Textbox.Input("Confirm Password")
 2.2.4 Repeat Until Password = Confirm Password AND  len(Username)  < 15 AND
username <> userData().username
2.3 Update Database
  2.3.1 Open Database
  2.3.2 Set up Update Query
  2.3.3 Run Update Query
  2.3.4 Close Database
2.4 Update (userData)
2.5 Display "Thank you for registering an account with us" + userData(num).username
2.6 backToMenu_Click
2.7 End Sub


3.1 CheckRules_Click
 3.2 go_to_DisplayRules.Form
 3.2.1 Display blackJRules
 3.2.2 Display Deck
```

3.2.3 Display deckValues
3.3 End Sub

4.1 CheckAccountDetails_Click
4.2 Go_to_signIn.Form
4.2.1 Validate_Inputs (userData)
4.2.2 Repeat_Until_Valid
4.2.3 backToMenu_Click
4.3 Go_to_accDetails.Form
4.3.1 Display userData(pos).userName
4.3.2 Display userData(pos).Balance
4.3.3 Display userData(pos).played
4.3.4 Display userData(pos).won
4.3.5 backToMenu_Click
4.4 End Sub

4.5 leaderboard_Click
4.6 Go_to_leaderboard.Form
4.6.1 Initialise structure userData
4.6.1.1 Initialise array userInfo as userData
4.6.2 Read In Values from Database
4.6.2.1 Open Database
4.6.2.2 Initialise Select Query
4.6.2.3 Run Select Query
4.6.2.4 Read values into array of records
4.6.2.4 Close Database
4.6.3 Bubble_Sort (In Order of highest balance to lowest)
4.6.3.1 Initialise swapped as Boolean = True
4.6.3.1.1 Initialise temp as userData
4.6.3.2 Do While swapped = True
4.6.3.3 swapped = false
4.6.3.4 For counter = 1 to Len(userInfo)
4.6.3.5 If userInfo(counter -1).balance < userInfo(counter).balance Then
4.6.3.6 swapped = true
4.6.3.7 temp = user(counter - 1)
4.6.3.8 user(counter - 1) = user(counter)
4.6.3.9 user(counter) = temp

4.6.3.10 End If
4.6.3.11 Next
4.6.3.12 End While
4.6.4 Fixed Loop displays all usernames, and balances of every player in descending order
4.6.5 backToMenu_Click
4.7 End Sub

5.1 toDesktop_Click
5.2 terminate_program
5.3 End Sub

Actual game code
6. game.Form
6.1 Display gameTable
6.2 For counter = 1 to numPlayers
6.2.1 In position(counter) display players(counter).name, players(counter).balance, players(counter). bet
6.2.2 Next
6.3 In position(dealer) display “Dealer”
6.4.0 initialise playersPlaying as integer = numPlayers

6.4 While playersPlaying > 1
6.4.1 For counter = 1 to numPlayers
6.4.2 players(counter).in = true
6.4.3 players(counter).bust = false
6.4.4 players(counter).bet = 0
6.4.5 players(counter).cardTotal = 0
6.4.6 players(counter).cardValue = 0
6.4.7 End If
6.4.8 Next

6.5 For counter = 1 to numPlayers
6.5.1 If players(counter).playing = true Then
6.5.2 players(counter).bet = 0

```
  6.5.3 players(counter).bet = Input - (“How much will you bet on this round? ” +
players(counter).name
 6.5.3.1 Validate input with Try Catch (Catches if execution error happens and
responds with the action entered into code, in this case displays an error message)
  6.5.4 players(counter).balance = players(counter).balance - players(counter).bet
  6.5.5 else
  6.5.6 playersIn = playersIn -1
  6.5.7 players(counter).bet = 0
  6.5.8 players(counter).cardTotal = 0
  6.5.9 players(counter).cardValue = 0
 6.5.10 End If
6.5.11 Next

6.6 For counter = 1 to numPlayers
  6.6.1 If players(counter).playing = true
  6.6.2 in position(counter) display 2.cards
  6.6.3 update players(counter).cardValue
  6.6.4 players(counter).cardTotal += 2
  6.6.5 End if
  6.6.6 Next
 6.6.7 in position(dealer) display 1.card(faceDown)
 6.6.8 in position (dealer) display 1.card
 6.6.9 update cardValue(dealer)
6.6.10 cardTotal(dealer) += 2
 6.7 For counter = 1 to numPlayers
  6.7.1 If players(counter).cardTotal = 21 Then
  6.7.2 display (players(counter).name + “ has got Blackjack!”)
  6.7.3 players(counter).in = false
  6.7.4 playersIn = playersIn -1
 6.7.4 endIf
 6.7.5 Next



 6.8 While playersIn > 0
  6.9 For counter = 1 to numPlayers
  6.9.1 If players(counter).bust = false AND players(counter).playing = true AND
players(counter).in = true Then
```

```
   6.9.2 Display (players(counter).name + " Your card value currently is " +
players(counter).cardValue + " What option would you        like to choose". options -
Hit, Stay, Double Down.)
6.9.3 End If
 6.9.4 If option = Hit Then
  6.9.5 in position(counter) display 1.cards
  6.9.6 Update players(counter).cardValue
  6.9.6.1 players(counter).cardTotal +=1
  6.9.7 elseIF option = Stay Then
  6.9.8 players(counter).in = false
  6.9.8.1 playersIn = playersIn - 1
  6.9.9 elseIf option =  Double Down Then
  6.9.10 in position(counter) display 2.cards
  6.9.11 players(counter).balance = player(counter).balance - player(counter).bet
  6.9.12 player(counter).bet = player(counter).bet*2
  6.9.13 update players(counter).cardValue
  6.9.13.1 players(counter).cardTotal += 2
 6.9.14 End If
 6.9.15 If cardValue >21 Then
  6.9.16 Display (players(counter).name + " You've gone bust.")
  6.9.17 players(counter).bust = true
  6.9.18 playersIn = playersIn -1
 6.9.18.1 elseIF players(counter).cardTotal = 5  AND players(counter).cardValue <=
21 Then
  6.9.18.2 players(counter).in = false
  6.9.18.3 playersIn = playersIn -1
  6.9.18.4 display (players(counter).name + " has got 5 cards!")
  6.9.19 End If
6.9.20 Next



6.10 Loop (terminates to while loop on line 6.8)
6.11 Flip card(Dealer).face_down
6.11.1 Update cardValue.Dealer
6.11.2 If cardValue(dealer) = 21 Then
 6.11.2.1 display ("The dealer has got Blackjack!")
6.12 While cardValue.(dealer) <=16
 6.12.1 in position(dealer) display 1.cards
```

6.12.2 update cardValue(dealer)
6.12.3 **Loop**

6.13 If cardValue(dealer) >21 Then
6.13.1 display (“The dealer has gone bust!”)
6.13.2 For counter = 1 to numplayers
6.13.3 If players(counter).playing = true AND players(counter).bust = false Then
6.13.4 display (players(counter).name “ has won “ + players(counter).bet)
6.13.5 players(counter).balance = players(counter).balance + 2*players(counter).bet
Else
6.13.6 display (players(counter).name + “ have lost “ + player(counter).bet)
6.13.7 End If
6.13.8 Next
6.14.9 End If

6.14 If cardValue(dealer)= 21 AND cardTotal(dealer) = 2 Then
6.14.1 For counter = 1 to numPlayers
6.14.2 If players(counter).cardValue = 21 AND players(counter).cardTotal = 2 Then
6.14.2 display (players(counter).name + “ has drawn with the dealer and will not lose
any money.”)
6.14.3 players(counter).balance += players(counter).bet
6.14.4 Else
6.14.5 display (players(counter).name “ has lost ” + players(counter).bet)
6.14.6 End If
6.14.7 Next
6.14.8 End If

6.15 If cardValue(dealer) = 21 AND cardTotal(dealer) > 2 Then
6.15.1 For counter = 1 to numPlayers
6.15.2 If (players(counter).cardValue = 21 AND players(counter).cardTotal = 2) OR
(players(counter).cardValue = 5 AND players(counter).cardValue <=21) Then
6.15.3 display(players(counter).name + “ Has won “ + players(counter).bet)
6.15.4 players(counter).balance += players(counter).bet*2
6.15.5 elseIf players(counter).cardValue = 21 AND players(counter).cardTotal > 2
Then
6.15.6 Display (players(counter).name + “ has drawn with the dealer and won’t lost
their bet.”)
6.15.6 players(counter).balance += players(counter).bet

6.15.7 else
6.15.8 display (players(counter).name + " has lost their bet of " + players(counter).bet)
6.15.9 End if
6.15.10 Next
6.15.11 End if

6.16 If cardValue(dealer) < 21 Then
6.16.1 For counter 1 to num players
6.16.2 If (players(counter).cardValue > cardValue(dealer) AND players(counter).bust = false) OR (players(counter).cardTotal = 5 AND players(counter).bust = false) Then
6.16.3 display (players(counter).name _+ " has won " + players(counter).name)
6.16.4 players(counter).balance += players(counter).bet*2
6.16.5 elseIf player(counter).cardValue = cardValue(dealer) Then
6.16.6 display (players(counter).name + " has drawn with the dealer and won't lose any money this round.")
6.16.7 players(counter).balance += players(counter).bet
6.16.8 elseIf players(counter).bust = true OR players(counter).cardValue < cardValue(dealer)
6.16.9 display(players(counter).name + " has lost " + players(counter).bet
6.16.10 Next
6.16.11 End If

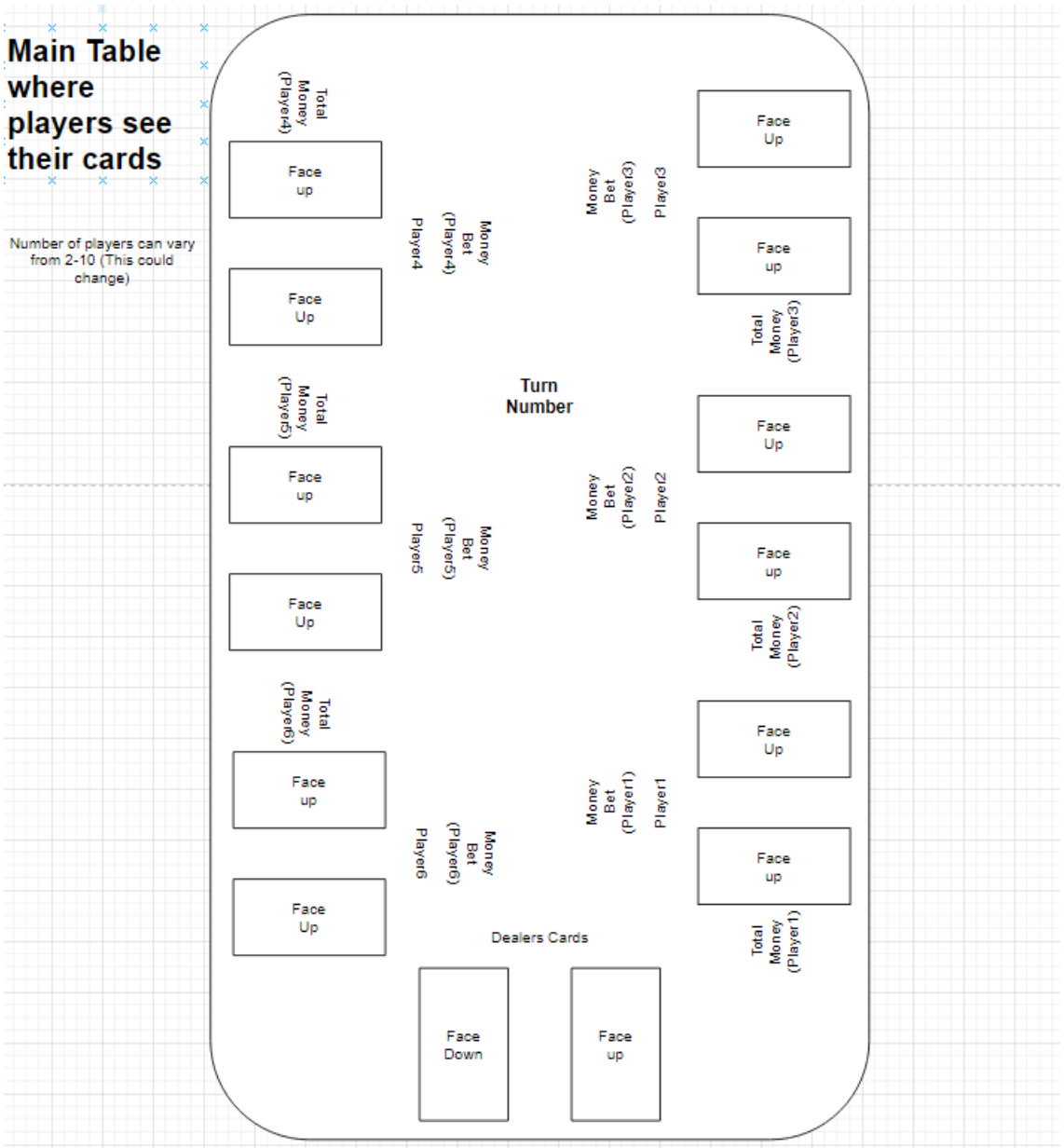
6.17 using players() update Database

6.18 For counter = 1 to numPlayers
6.18.1 If players(counter).playing = true Then
6.18.2 display (players(counter).name + " would you like to continue playing, remember to bet in moderation."). Option - yes, no
6.18.3 If option = no then
6.18.4 players(counter).playing = false
6.18.5 playersPlaying = playersPlaying - 1
6.18.6 End if
6.18.7 End If
6.18.7 Next

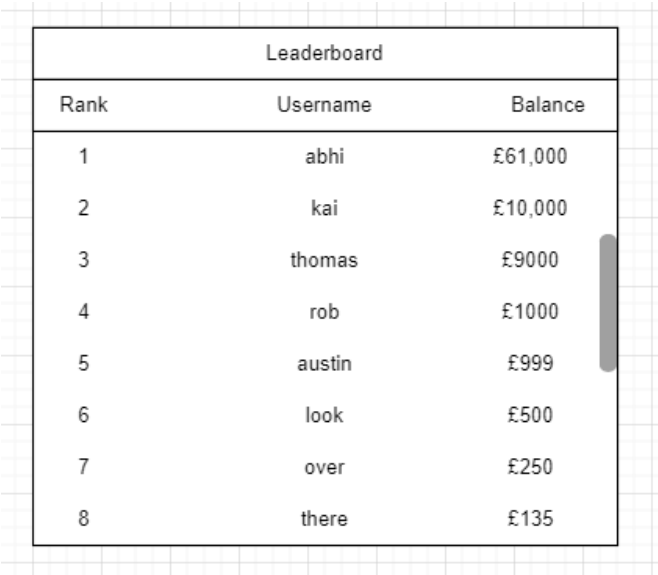
6.19 **Loop**
6.20 display("The game has ended, thank you for playing.")

6.21 back_to_mainMenu.For

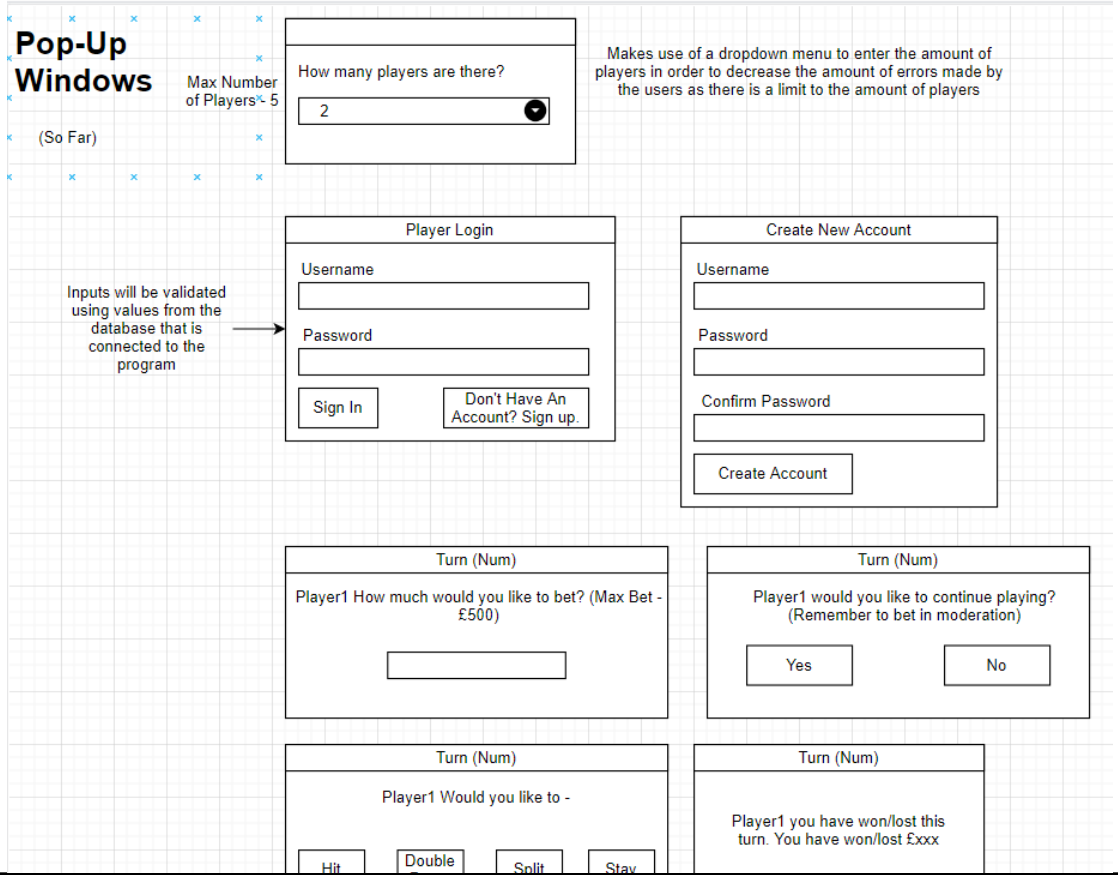
User Interface Design (5 marks)

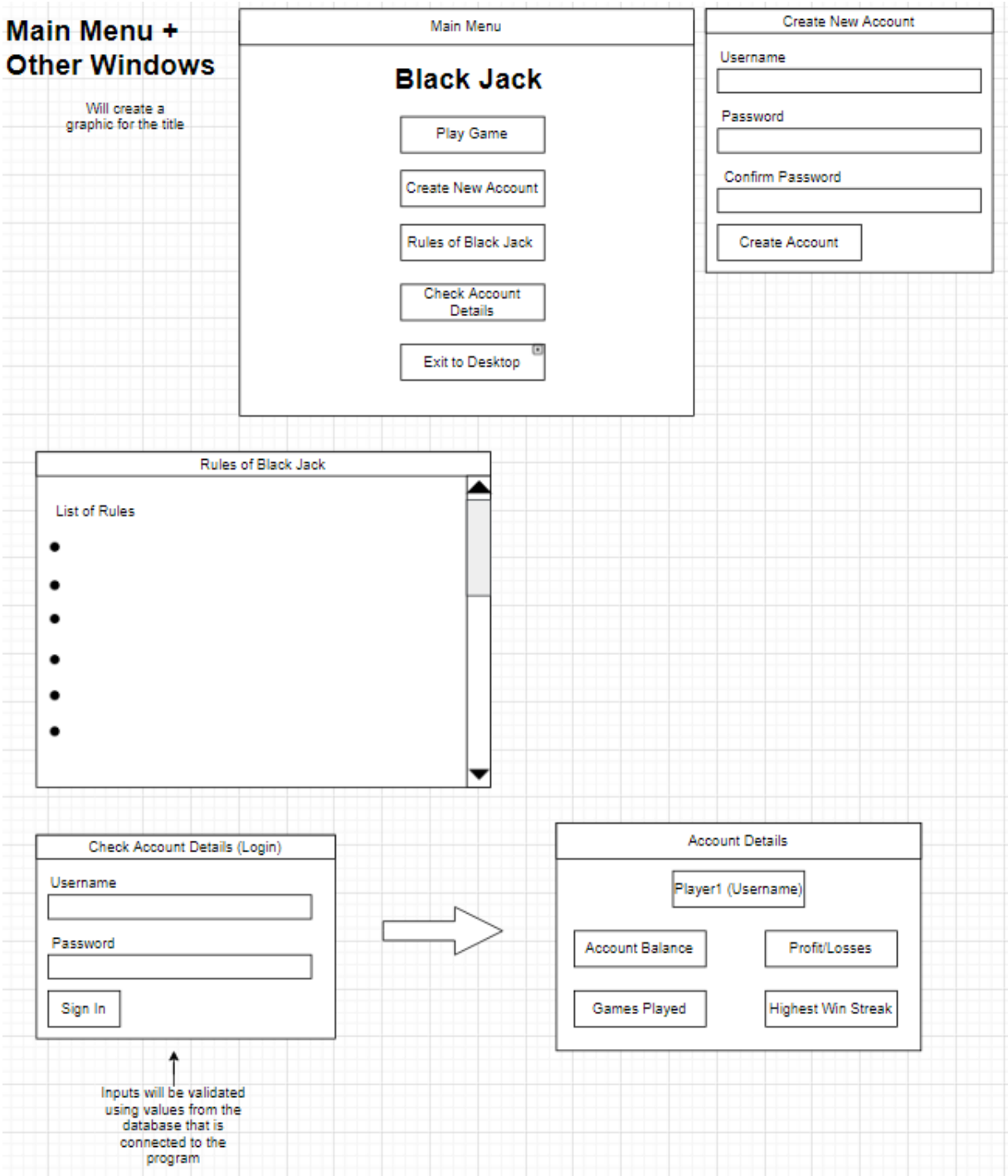

Main Table where players see their cards
Number of players can vary from 2-10 (This could change)
Total Money (Player4)
Face up
Face Up
Money Bet (Player4)
Player4
Money Bet (Player3)
Player3
Face Up
Face up
Total Money (Player3)
Turn Number
Total Money (Player5)
Face up
Face Up
Money Bet (Player5)
Player5
Money Bet (Player2)
Player2
Face Up
Face up
Total Money (Player2)
Total Money (Player6)
Face up
Face Up
Money Bet (Player6)
Player6
Money Bet (Player1)
Player1
Face Up
Face up
Total Money (Player1)
Dealers Cards
Face Down
Face up

The Leaderboard form was added in later on in the implementation process



Leaderboard		
Rank	Username	Balance
1	abhi	£61,000
2	kai	£10,000
3	thomas	£9000
4	rob	£1000
5	austin	£999
6	look	£500
7	over	£250
8	there	£135




Main Menu +
Other Windows
Will create a graphic for the title
Main Menu
Black Jack
Play Game
Create New Account
Rules of Black Jack
Check Account Details
Exit to Desktop
Create New Account
Username
Password
Confirm Password
Create Account
Rules of Black Jack
List of Rules
Check Account Details (Login)
Username
Password
Sign In
Inputs will be validated using values from the database that is connected to the program
Account Details
Player1 (Username)
Account Balance
Profit/Losses
Games Played
Highest Win Streak

Implementation (30 marks)

You must now implement your solution and user interface, based on your design.

The problem you are solving will determine the evidence you provide of your implemented solution, including the user interface. The following may be suitable:

Software design and development

- program code
- screenshots of program user interface

Database design and development

- SQL code
- screenshots of implemented tables

Web design and development

- PHP code
- HTML code and page content
- CSS declarations
- screenshots of pages, as viewed in a browser

Implementation (21 marks)

Implement your solution, including the user interface, ensuring it matches your completed design.

As you implement your solution, you will encounter errors or problems that you need to solve before you can continue. Take notes of errors, solutions and any reference materials you use, for example websites, forums, textbooks or learning resources. You will need to refer to these notes to produce evidence of ongoing testing (see below).

Your implementation should include some coding that goes beyond what you are taught during the Advanced Higher course. This will require you to carry out some research. You should keep a note of any references you use and where in your solution you use the skills and knowledge you learn through your research, as you will need to provide a description of this.

Errors encountered

When reading values from a database into an array it was only reading the first value rather than all of them - this was due to the number being used as the element of the array wasn't being updated each loop, once the number was updated using a running total the code worked fine.

When trying to show the cards in certain positions, the show function was unable to hold parameters that chose the coordinates for where the card would be shown, a fix to this is to add another pre-defined function called 'setbounds' which can be used to set the coordinates of a picturebox.

When dealing the cards every round they were in the same order, this was because the array wasn't shuffled properly, this was fixed by researching a procedure that shuffles an array randomly every time it shuffles it (Fisher Yates Shuffle)

Navigating the program properly, there was no way to go back to a previous form, this has been changed by adding buttons that go back to previous forms, but since some of the same forms can be accessed from differing forms there is a validation system which checks which form the new one has been accessed from.

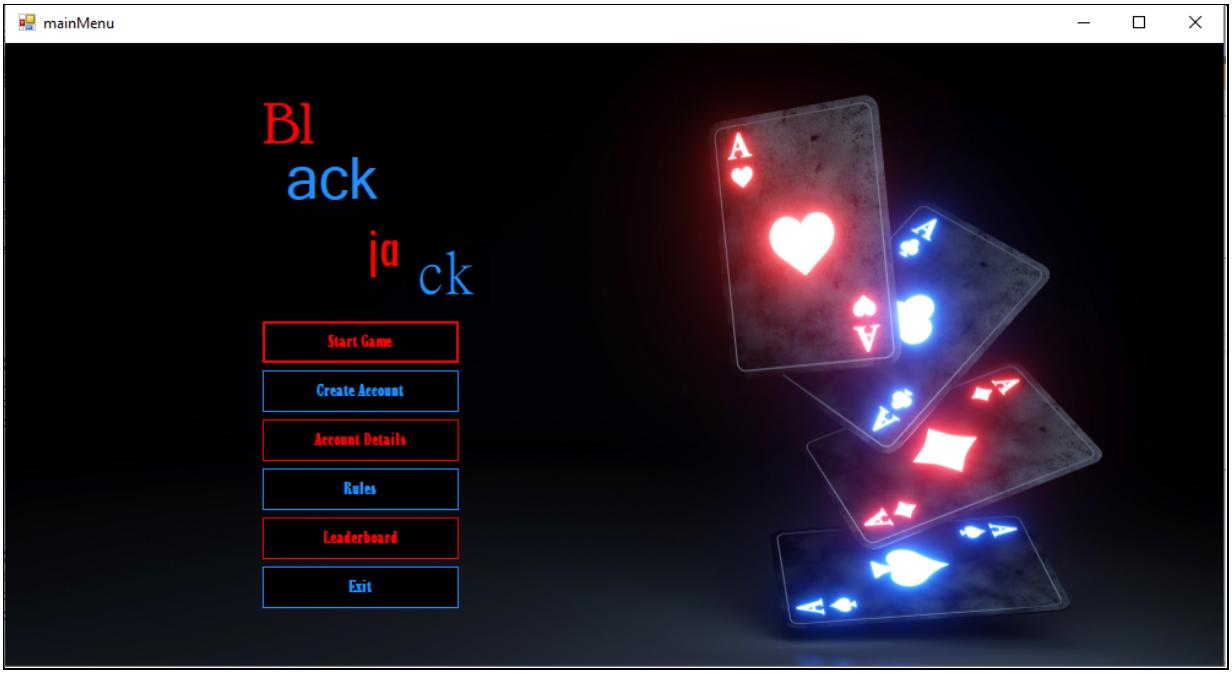
When creating an account during the sign in section of the game form, the form kept resetting the variables that had already had data input into them when hidden, making it so everyone had to re-enter their information if someone created an account.

References

Stack Overflow - passing variables between forms, sql statements in visual basic, shuffling an array
Microsoft VB Website - moving and displaying pictureboxes
Youtube - VB dropdown menu, global variables
Mr Rigg (Comp Sci Teacher) - Link to database. SQL from Visual Basic

Final Coding/UI

Main Menu UI



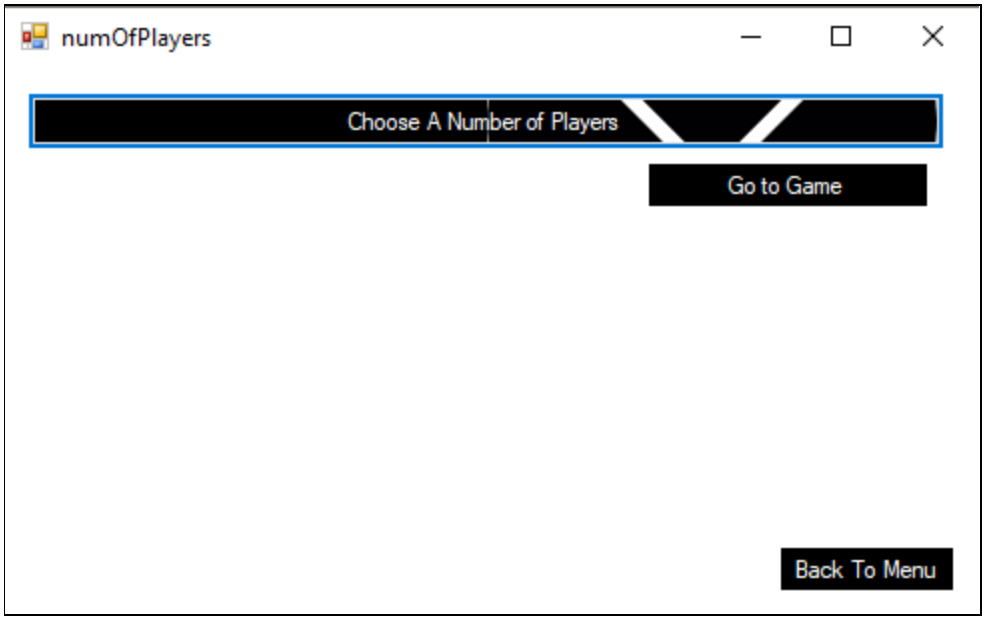
Main Menu Code

```
Public Class mainMenu

    'global variables that let the program know which route the program has
gone through to get to the point it is at
    Public Property clickTypeCreate As String
    Public Property clickTypeSign As String
    Public Property playerNumber1 As New numOfPlayers


    Private Sub StartGame_Click(sender As Object, e As EventArgs) Handles
startGame.Click
'opens the numOfPlayers form, then to the gameUI form

        Me.Hide()
```

```
        playerNumber1.Show()
        clickTypeSign = "game"
        clickTypeCreate = "game"

    End Sub

    Private Sub CreateAccount_Click(sender As Object, e As EventArgs) Handles
createAccount.Click
'opens the createAccount form

        Me.Hide()
        Dim createAccount1 As New createAccount
        createAccount1.Show()
        clickTypeCreate = "menu"

    End Sub

    Private Sub Details_Click(sender As Object, e As EventArgs) Handles
details.Click
'opens signIn form, then goes to accountDetails form


        Dim signInDet As New signIn
        Me.Hide()
        signInDet.Show()
        clickTypeSign = "details"

    End Sub

    Private Sub Rules_Click(sender As Object, e As EventArgs) Handles
rules.Click
' goes the the rules form

        Me.Hide()
        Dim rules1 As New rules
        rules1.Show()

    End Sub

    Private Sub Leave_Click(sender As Object, e As EventArgs) Handles
leave.Click
'terminates the program
```

```
        End
    End Sub
    Private Sub Leaderboard_Click(sender As Object, e As EventArgs) Handles
leaderboard.Click
'goes to the leaderboard form

        Me.Hide()
        Dim leaderboard1 As New leaderboard
        leaderboard1.Show()

    End Sub
End Class

‘Going through the “Start Game” option
```

Choosing Number of players Form UI



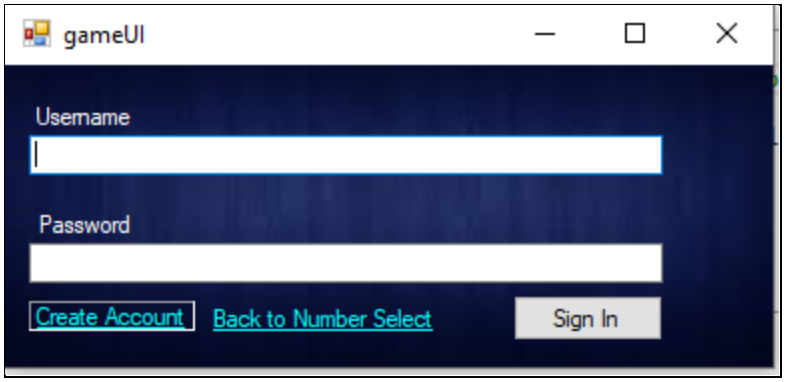
Choosing number of players Form Code

```
Public Class numOfPlayers

    'global variables
    'a variable that creates the gameUI form
    Public Property playingGame As New gameUI

    'variable that holds the chosen number of players to play in the game
    Public Property numOfPlayersInGame As Integer

    'shows a dropdown menu that only appears when this button is clicked
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ContextMenuStrip1.Show(Button1, 0, Button1.Height)
    End Sub

    'goes back to main menu
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click

        Me.Hide()
        mainMenu.Show()

    End Sub

    'carries out when form is loaded, hides the button that takes you to
the game menu until a number of players has been selected
    Private Sub NumOfPlayers_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        loadGame.Hide()

    End Sub

    'each of these button clicks sets the value of a global variables to a
certain number corresponding to the button clicked
    Private Sub ToolStripMenuItem2_Click(sender As Object, e As EventArgs)
Handles ToolStripMenuItem2.Click
```

```
        numOfPlayersInGame = 1
        loadGame.Show()

    End Sub

    Private Sub ToolStripMenuItem3_Click(sender As Object, e As EventArgs)
Handles ToolStripMenuItem3.Click

        numOfPlayersInGame = 2
        loadGame.Show()


    End Sub

    Private Sub ToolStripMenuItem4_Click(sender As Object, e As EventArgs)
Handles ToolStripMenuItem4.Click

        numOfPlayersInGame = 3
        loadGame.Show()

    End Sub

    Private Sub ToolStripMenuItem5_Click(sender As Object, e As EventArgs)
Handles ToolStripMenuItem5.Click

        numOfPlayersInGame = 4
        loadGame.Show()

    End Sub

    Private Sub ToolStripMenuItem6_Click(sender As Object, e As EventArgs)
Handles ToolStripMenuItem6.Click

        numOfPlayersInGame = 5
        loadGame.Show()

    End Sub

    'takes you to the gameUI form when clicked
    Private Sub LoadGame_Click(sender As Object, e As EventArgs) Handles
loadGame.Click
```

```
        loadGame.Hide()
        Me.Hide()
        playingGame.Show()


    End Sub
End Class
```

BlackJack Form (Sign In Stage) UI



BlackJack Form (Sign in Stage) Code

```
Imports System.Data.OleDb
'game form
Public Class gameUI

    Structure validateSignIn
' structure that holds the values that are read in from a database, used
for input validation

        Dim user As String
        Dim pass As String

    End Structure

    Structure playerData
' when a user signs in and uses a correct user and password their
information is stored in this record (array of records)
```

```
        Dim username As String
        Dim balance As Integer
        Dim playing As Boolean
        Dim bust As Boolean
        Dim stayed As Boolean
        Dim cardTotal As Integer
        Dim cardValue As Integer
        Dim bet As Integer

    End Structure

    Structure dealerData
'record structure used to store the dealers information (not an array)

        Dim stayed As Boolean
        Dim bust As Boolean
        Dim cardTotal As Integer
        Dim cardValue As Integer

    End Structure

    'value the counts how many people have signed into game so far and
compares it to the amount that was selected in the previous form
    Private signInCounter As Integer = 0

    'an array of records that holds the data from players who have signed
into the game
    Private playersPlaying(0) As playerData


    Private Sub GameUI_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
' when the form is loaded these procedures are carried out

        hidingEverything()
        ReDim playersPlaying(0)
        Me.Size = New Size(400, 190)

    End Sub

    Private Sub hidingEverything()
```

```
' when players need to sign in this hides everything on the form (cards,
labels) apart from the textboxes

        blank.Hide()
        backofCard.Hide()
        aceSpades.Hide()
        aceClubs.Hide()
        aceDiamonds.Hide()
        aceHearts.Hide()
        twoSpades.Hide()
        twoClubs.Hide()
        twoDiamonds.Hide()
        twoHearts.Hide()
        threeSpades.Hide()
        threeClubs.Hide()
        threeDiamonds.Hide()
        threeHearts.Hide()
        fourSpades.Hide()
        fourClubs.Hide()
        fourDiamonds.Hide()
        fourHearts.Hide()
        fiveSpades.Hide()
        fiveClubs.Hide()
        fiveDiamonds.Hide()
        fiveHearts.Hide()
        sixSpades.Hide()
        sixClubs.Hide()
        sixDiamonds.Hide()
        sixHearts.Hide()
        sevenSpades.Hide()
        sevenClubs.Hide()
        sevenDiamonds.Hide()
        sevenHearts.Hide()
        eightSpades.Hide()
        eightClubs.Hide()
        eightDiamonds.Hide()
        eightHearts.Hide()
        nineSpades.Hide()
        nineClubs.Hide()
        nineDiamonds.Hide()
        nineHearts.Hide()
        tenSpades.Hide()
```

```
        tenClubs.Hide()
        tenDiamonds.Hide()
        tenHearts.Hide()
        jackSpades.Hide()
        jackClubs.Hide()
        jackDiamonds.Hide()
        jackHearts.Hide()
        queenSpades.Hide()
        queenClubs.Hide()
        queenDiamonds.Hide()
        queenHearts.Hide()
        kingSpades.Hide()
        kingClubs.Hide()
        kingDiamonds.Hide()
        kingHearts.Hide()

        startRound.Show()
        endGame.Show()

        player1balance.Hide()
        player1Name.Hide()
        player1total.Hide()
        player1Bet.Hide()

        player2balance.Hide()
        player2Name.Hide()
        player2total.Hide()
        player2Bet.Hide()

        player3balance.Hide()
        player3Name.Hide()
        player3total.Hide()
        player3bet.Hide()

        player4balance.Hide()
        player4Name.Hide()
        player4total.Hide()
        player4bet.Hide()

        player5balance.Hide()
        player5Name.Hide()
        player5total.Hide()
```

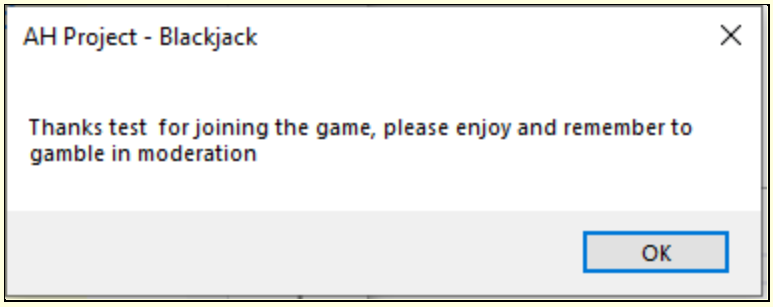
```
        player5bet.Hide()

        dealerName.Hide()
        dealerTotal.Hide()

    End Sub

    Private Sub LinkLabel1_LinkClicked(sender As Object, e As
LinkLabelLinkClickedEventArgs) Handles LinkLabel1.LinkClicked
' opens create account form if players haven't got an account and want to
make one


        mainMenu.clickTypeCreate = "game"
        createAccount.Show()

    End Sub

    Private Sub LinkLabel2_LinkClicked(sender As Object, e As
LinkLabelLinkClickedEventArgs) Handles LinkLabel2.LinkClicked
' hides the game form and reopens the form that lets players redecide the
amount of people playing

        Me.Hide()
        Dim redecideNum As New numOfPlayers
        redecideNum.Show()
        ReDim playersPlaying(0)
        signInCounter = 0

    End Sub

    Private Sub SignInGame_Click(sender As Object, e As EventArgs) Handles
signInGame.Click ' this button is clicked once a player has entered their
username and password and validates them

        'initialising variables
        Dim SignValidation() As validateSignIn

        Dim sqlReader As OleDbDataReader
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
```

```
Higher Project.accdb"

        'initialising connection to database
        Dim conn As OleDbConnection
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()


        ReDim SignValidation(elementsinArray("SELECT Count(username) AS
[Num] FROM [Players];", conn, sqlReader))


        userANDPasswords("SELECT [username], [password] FROM [Players];",
conn, sqlReader, SignValidation)


        Dim signInUser As String = gameSignUser.Text
        Dim signInPass As String = gameSignPass.Text


        validateInputs(signInUser, signInPass, SignValidation, conn,
sqlReader)
        conn.Close()



    End Sub

    Private Function elementsinArray(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader) As Integer
'function finds the number of elements in the database and sets up the
array to hold that many values

        Dim commandNum As New OleDbCommand(query, conn)
        reader = commandNum.ExecuteReader

        Dim elements As Integer
        If reader.HasRows Then
            While reader.Read

                elements = reader("Num")
```

```
            End While
        End If

        Return elements - 1
    End Function

        Private Sub userANDPasswords(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader, ByRef validation() As
validateSignIn)
' enters all usernames and passwords into an array of records in order to
validate user inputs


        Dim commandUsernames As New OleDbCommand(query, conn)
        reader = commandUsernames.ExecuteReader

        If reader.HasRows Then

            Dim num As Integer = 0
            While reader.Read

                validation(num).user = reader("username")
                validation(num).pass = reader("password")
                num += 1
            End While
        End If

    End Sub

       Private Sub validateInputs(ByVal user As String, ByVal pass As
String, ByRef validation() As validateSignIn, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader)
 'compares values from the database to the user entered values in order to
validate inputs if the values are


        Dim Check As Boolean = False


        For counter = 0 To UBound(validation)
            If (user = validation(counter).user And pass =
validation(counter).pass) Then
```

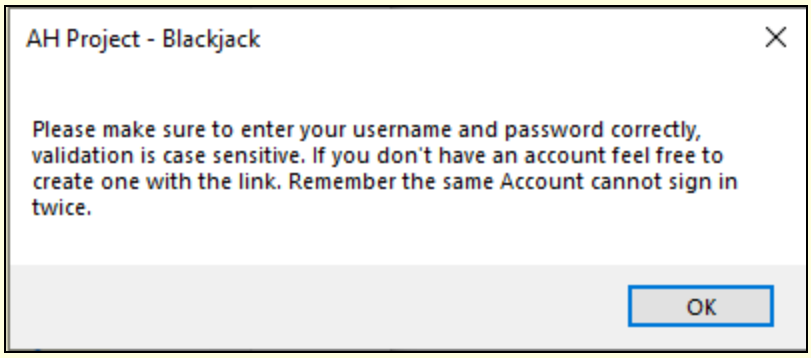
```
                Check = True
                gameSignUser.Text = ""
                gameSignPass.Text = ""
            End If
        Next

        For counter = 0 To UBound(playersPlaying)
            If user = playersPlaying(counter).username Then

                Check = False
            End If

        Next


        If Check = True Then
            Dim query As String = "SELECT [username], [accountBalance] FROM
[players] WHERE [username] = '" + user + "'"

            Dim CommandInfo As New OleDbCommand(query, conn)
            reader = CommandInfo.ExecuteReader


            While reader.Read

           playersPlaying(signInCounter).username = reader("username")
           playersPlaying(signInCounter).balance = reader("accountBalance")

            End While
            playersPlaying(signInCounter).bust = False
            playersPlaying(signInCounter).playing = True
            playersPlaying(signInCounter).cardValue = 0
            playersPlaying(signInCounter).cardValue = 0

            signInCounter += 1
```

```
MsgBox("Thanks " + user + "  for joining the game, please enjoy and remember to gamble in moderation")
```



Msgbox for previous line of code

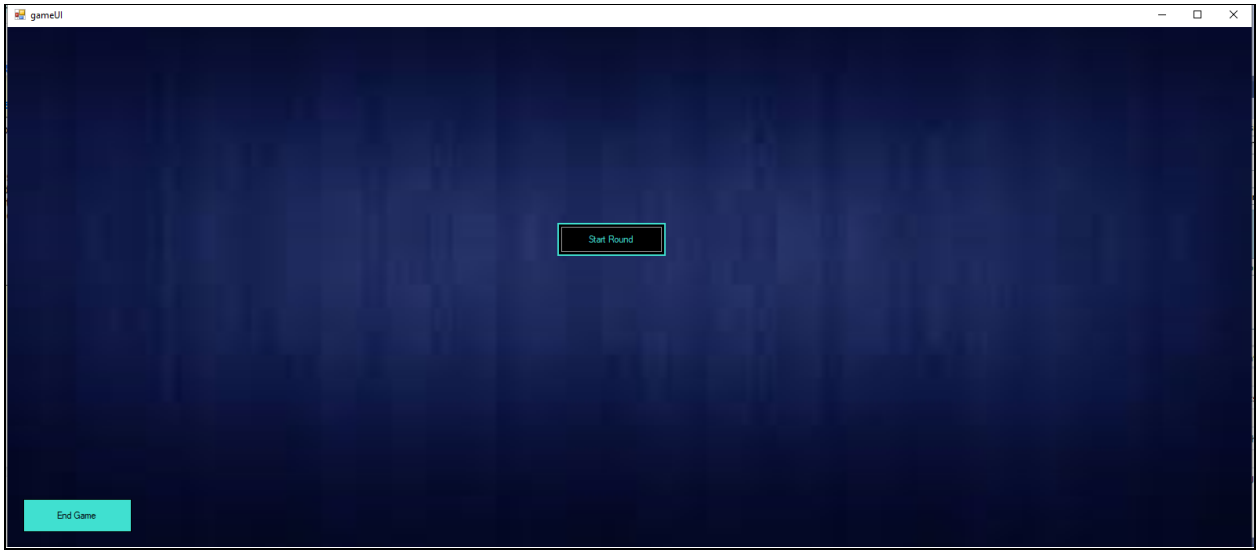
```
Else
            MsgBox("Please make sure to enter your username and password correctly, validation is case sensitive. If you don't have an account feel free to create one with the link. Remember the same Account cannot sign in twice.")
```

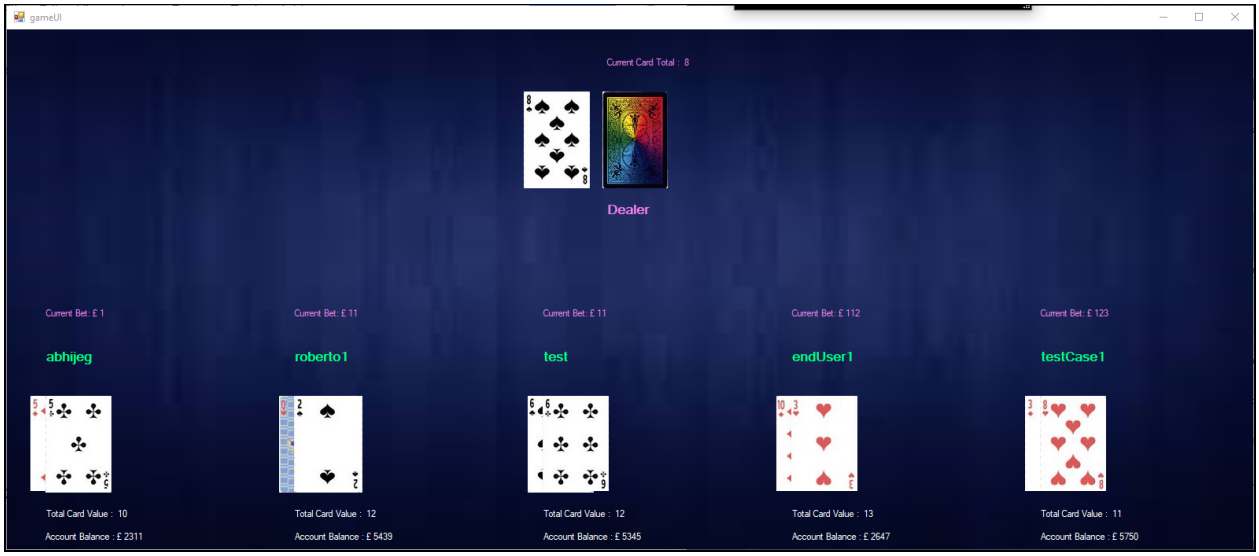



```
End If

        If signInCounter = mainMenu.playerNumber1.numOfPlayersInGame Then
            changeSignIn()
            MsgBox("Thank you everyone for signing into this game, please enjoy your time and play responsibly.")
```

```
        Else
            ReDim Preserve playersPlaying(signInCounter)
        End If


    End Sub

    Private Sub changeSignIn()
        ' once all users are signed in this procedure changes the layout of
the form to be able to fit the user interface for every player

        Me.Size = New Size(1600, 700)
        usernameInput.Hide()
        passInput.Hide()
        gameSignPass.Hide()
        gameSignUser.Hide()
        LinkLabel1.Hide()
        LinkLabel2.Hide()
        signInGame.Hide()

        endGame.Show()
        startRound.Show()

    End Sub
```

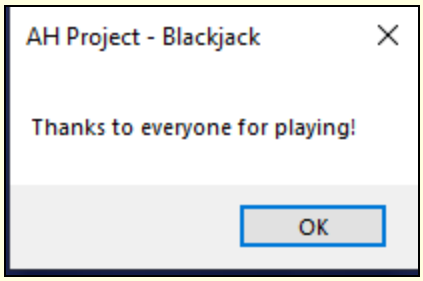
BlackJack Form (Game Stage) UI





BlackJack Form (Game Stage) Code, with pop ups

```
Private Sub EndGame_Click(sender As Object, e As EventArgs) Handles
endGame.Click
'if all players finish a round and decide they want to stop playing, this
closes the game form and reopens the main menu

        MsgBox("Thanks to everyone for playing!")
```




```
Private Sub hidingEverything()
        ReDim playersPlaying(0)
        Me.Size = New Size(400, 190)
        usernameInput.Show()
        passInput.Show()
        gameSignUser.Show()
        gameSignPass.Show()
        signInGame.Show()
        LinkLabel1.Show()
        LinkLabel2.Show()
        Me.Hide()
        mainMenu.Show()
        signInCounter = 0
    End Sub

    Private Sub StartRound_Click(sender As Object, e As EventArgs) Handles
startRound.Click
'this button is clicked to start the game and contains the majority of code
related to the game

        'initialising variables and setting up form for round of blackjack,
each round will start only when the start button is pressed
        Dim sqlReader As OleDbDataReader
```

```
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
Higher Project.accdb"

        'initialising connection to database
        Dim conn As OleDbConnection
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()

        'hides these 2 buttons so a players can't click them during a game
        endGame.Hide()
        startRound.Hide()

        'sets up 2 array that hold values for the coordinates of where the
cards will show depending on the player who has got them
        Dim playerPositions(24, 1) As Integer
        Dim dealerPositions(4, 1) As Integer
        inputCoordinates(playerPositions, dealerPositions)

        'clearing card values and totals from previous round
        resetRound()

        ' a record detailing information about the dealer
        Dim dealerInfo As dealerData

        ' initiating labels into arrays so it is possible to use fixed loops
to edit and display them
        Dim userLabels() As Label = {player1Name, player2Name, player3Name,
player4Name, player5Name}

        Dim balanceLabels() As Label = {player1balance, player2balance,
player3balance, player4balance, player5balance}

        Dim betLabels() As Label = {player1Bet, player2Bet, player3bet,
player4bet, player5bet}

        Dim totalLabels() As Label = {player1total, player2total,
player3total, player4total, player5total}

        'variable that keeps track of how many people are still playing in
this round
```

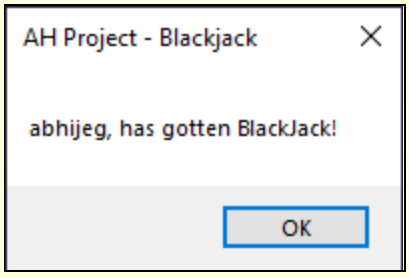
```
        Dim inRound = UBound(playersPlaying) + 1

        'updates the inround counter each time a game is started
        For counter = 0 To UBound(playersPlaying)

            If playersPlaying(counter).playing = False Then
                inRound -= 1
                userLabels(counter).Text += "(left)"
            End If

        Next

        'fixed loop that displays the username and other information about
each player so they get a grasp on the game easier
        For counter = 0 To UBound(playersPlaying)
                displayLabels(userLabels(counter), balanceLabels(counter),
betLabels(counter), totalLabels(counter), counter)

            Next



        dealerName.Show()
        dealerTotal.Show()

        'initialising all cards without using loops due to the fact each card
has a corresponding picturebox
        Dim sAce As New aces("Spades", 11, aceSpades, 1, False, -1)
        Dim s2 As New cards("Spades", 2, twoSpades)
        Dim s3 As New cards("Spades", 3, threeSpades)
        Dim s4 As New cards("Spades", 4, fourSpades)
        Dim s5 As New cards("Spades", 5, fiveSpades)
        Dim s6 As New cards("Spades", 6, sixSpades)
        Dim s7 As New cards("Spades", 7, sevenSpades)
        Dim s8 As New cards("Spades", 8, eightSpades)
        Dim s9 As New cards("Spades", 9, nineSpades)
        Dim s10 As New cards("Spades", 10, tenSpades)
        Dim sJack As New cards("Spades", 10, jackSpades)
        Dim sQueen As New cards("Spades", 10, queenSpades)
        Dim sKing As New cards("Spades", 10, kingSpades)

        Dim cAce As New aces("Clubs", 11, aceClubs, 1, False, -1)
```

```
        Dim c2 As New cards("Clubs", 2, twoClubs)
        Dim c3 As New cards("Clubs", 3, threeClubs)
        Dim c4 As New cards("Clubs", 4, fourClubs)
        Dim c5 As New cards("Clubs", 5, fiveClubs)
        Dim c6 As New cards("Clubs", 6, sixClubs)
        Dim c7 As New cards("Clubs", 7, sevenClubs)
        Dim c8 As New cards("Clubs", 8, eightClubs)
        Dim c9 As New cards("Clubs", 9, nineClubs)
        Dim c10 As New cards("Clubs", 10, tenClubs)
        Dim cJack As New cards("Clubs", 10, jackClubs)
        Dim cQueen As New cards("Clubs", 10, queenClubs)
        Dim cKing As New cards("Clubs", 10, kingClubs)

        Dim hAce As New aces("Hearts", 11, aceHearts, 1, False, -1)
        Dim h2 As New cards("Hearts", 2, twoHearts)
        Dim h3 As New cards("Hearts", 3, threeHearts)
        Dim h4 As New cards("Hearts", 4, fourHearts)
        Dim h5 As New cards("Hearts", 5, fiveHearts)
        Dim h6 As New cards("Hearts", 6, sixHearts)
        Dim h7 As New cards("Hearts", 7, sevenHearts)
        Dim h8 As New cards("Hearts", 8, eightHearts)
        Dim h9 As New cards("Hearts", 9, nineHearts)
        Dim h10 As New cards("Hearts", 10, tenHearts)
        Dim hJack As New cards("Hearts", 10, jackHearts)
        Dim hQueen As New cards("Hearts", 10, queenHearts)
        Dim hKing As New cards("Hearts", 10, kingHearts)

        Dim dAce As New aces("Diamonds", 11, aceDiamonds, 1, False, -1)
        Dim d2 As New cards("Diamonds", 2, twoDiamonds)
        Dim d3 As New cards("Diamonds", 3, threeDiamonds)
        Dim d4 As New cards("Diamonds", 4, fourDiamonds)
        Dim d5 As New cards("Diamonds", 5, fiveDiamonds)
        Dim d6 As New cards("Diamonds", 6, sixDiamonds)
        Dim d7 As New cards("Diamonds", 7, sevenDiamonds)
        Dim d8 As New cards("Diamonds", 8, eightDiamonds)
        Dim d9 As New cards("Diamonds", 9, nineDiamonds)
        Dim d10 As New cards("Diamonds", 10, tenDiamonds)
        Dim dJack As New cards("Diamonds", 10, jackDiamonds)
        Dim dQueen As New cards("Diamonds", 10, queenDiamonds)
        Dim dKing As New cards("Diamonds", 10, kingDiamonds)

        'using an array as a deck of cards
```

```
        Dim deckOfCards() As cards = {sAce, s2, s3, s4, s5, s6, s7, s8, s9,
s10, sJack, sQueen, sKing, cAce, c2, c3, c4, c5, c6, c7, c8, c9, c10, cJack,
cQueen, cKing, hAce, h2, h3, h4, h5, h6, h7, h8, h9, h10, hJack, hQueen,
hKing, dAce, d2, d3, d4, d5, d6, d7, d8, d9, d10, dJack, dQueen, dKing}

shuffleCards(deckOfCards)


 'using a running total to keep track of the cards that have been dealt
        Dim cardsOut As Integer = 0

        ' fixed loop that allows each player to place their bets
        For counter = 0 To UBound(playersPlaying)
            If playersPlaying(counter).playing = True Then
                placeBets(counter, betLabels(counter),
balanceLabels(counter))
            End If


        Next

        'deals the initial 2 cards that everyone gets to the players that are
in the game and updates their totals
        For i = 0 To 1

            For counter = 0 To UBound(playersPlaying)

                If playersPlaying(counter).playing = True Then

                    dealCards(cardsOut, counter, playerPositions,
deckOfCards, totalLabels(counter))

                    'this statement checks if an ace has been dealt to a
player and if it has then it sets the players counter the the aces position
                    If cAce.getdealt() = True Then
                        cAce.setdealtPosition(counter)
                        cAce.setdealt(False)

                    ElseIf sAce.getdealt() = True Then
                        sAce.setdealtPosition(counter)
                        sAce.setdealt(False)
```

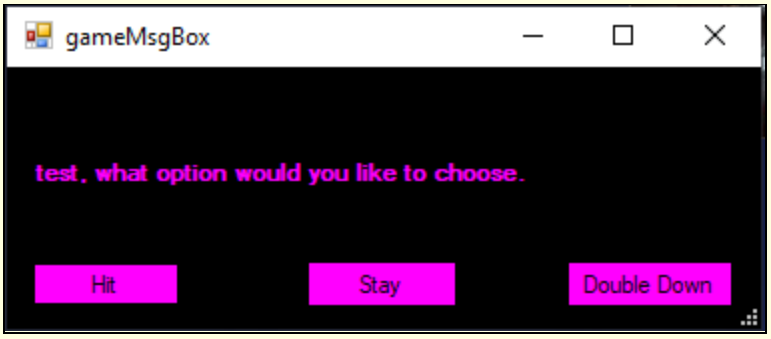
```
                    ElseIf hAce.getdealt = True Then
                        hAce.setdealtPosition(counter)
                        hAce.setdealt(False)
                    ElseIf dAce.getdealt = True Then
                        dAce.setdealtPosition(counter)
                        dAce.setdealt(False)

                    End If

                End If

            Next

        Next

        'deals one card to the dealer and displays an upside down card, also
updates their totals
        dealerInfo.cardValue += deckOfCards(cardsOut).getValue
        deckOfCards(cardsOut).showCard(dealerPositions(0, 0),
dealerPositions(0, 1))

        'checks if any player has an ace and if their card value is over 21,
if that is the case it changes the value of an ace from 11 to 1
        If cAce.getdealt() = True And cAce.getdealtPosition() = -1 Then
            cAce.setdealtPosition(100)
            cAce.setdealt(False)

        ElseIf sAce.getdealt() = True And sAce.getdealtPosition() = -1 Then
            sAce.setdealtPosition(100)
            sAce.setdealt(False)

        ElseIf hAce.getdealt = True And hAce.getdealtPosition() = -1 Then
            hAce.setdealtPosition(100)
            hAce.setdealt(False)

        ElseIf dAce.getdealt = True And dAce.getdealtPosition() = -1 Then
            dAce.setdealtPosition(100)
            dAce.setdealt(False)

        End If

        cardsOut += 1
```

```
        dealerInfo.cardTotal += 1

        backofCard.SetBounds(dealerPositions(1, 0), dealerPositions(1, 1),
aceSpades.Width, aceSpades.Height)
        backofCard.Show()

        dealerTotal.Text = "Current Card Total : " +
Str(dealerInfo.cardValue)

        'checks if any of the players got blackjack, causes them to stay
straight away
        For counter = 0 To UBound(playersPlaying)

            If playersPlaying(counter).cardValue = 21 Then

                MsgBox(playersPlaying(counter).username + ", has gotten
BlackJack!")
```




```
                playersPlaying(counter).stayed = True
                inRound -= 1
                userLabels(counter).Text += "(Stayed)"
            End If

        Next



        ' keeps going until all players playing in the round are finished,
checks if the players want more cards or want to stop
        While inRound > 0

            'fixed loop displays a form for each player in turn giving them
the options that they can choose from
```

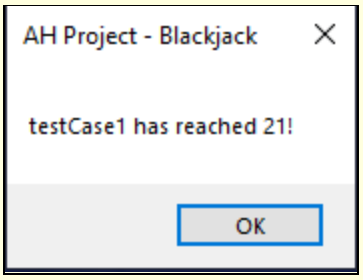
```
            For counter = 0 To UBound(playersPlaying)

                If playersPlaying(counter).bust = False And
playersPlaying(counter).playing = True And playersPlaying(counter).stayed =
False Then

                    Dim messageBox As New gameMsgBox
                    messageBox.roundOptionUsername =
playersPlaying(counter).username
                    messageBox.ShowDialog()
```




```
                    If messageBox.roundOptionDecision = "hit" Then
                        dealCards(cardsOut, counter, playerPositions,
deckOfCards, totalLabels(counter))


                    ElseIf messageBox.roundOptionDecision = "stay" Then
                        playersPlaying(counter).stayed = True
                        userLabels(counter).Text += "(Stayed)"
                        inRound -= 1

                    ElseIf messageBox.roundOptionDecision = "dd" Then
                        dealCards(cardsOut, counter, playerPositions,
deckOfCards, totalLabels(counter))
                        dealCards(cardsOut, counter, playerPositions,
deckOfCards, totalLabels(counter))
          playersPlaying(counter).balance -= playersPlaying(counter).bet
          playersPlaying(counter).bet = playersPlaying(counter).bet * 2

                        betLabels(counter).Text = "Current Bet : £" +
Str(playersPlaying(counter).bet)
```

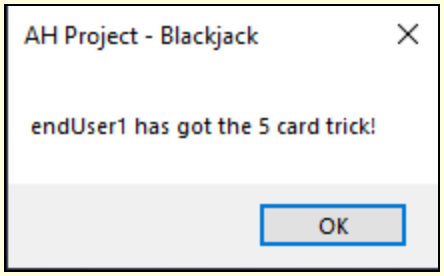
```
                        balanceLabels(counter).Text = "Account Balance : £" +
Str(playersPlaying(counter).balance)

                    End If


                    If cAce.getdealt() = True Then
                        cAce.setdealtPosition(counter)
                        cAce.setdealt(False)

                    ElseIf sAce.getdealt() = True Then
                        sAce.setdealtPosition(counter)
                        sAce.setdealt(False)

                    ElseIf hAce.getdealt = True Then
                        hAce.setdealtPosition(counter)
                        hAce.setdealt(False)

                    ElseIf dAce.getdealt = True Then
                        dAce.setdealtPosition(counter)
                        dAce.setdealt(False)

                    End If


                    If playersPlaying(counter).cardValue > 21 And
cAce.getdealtPosition = counter Then

                        cAce.setValue(1)
                        playersPlaying(counter).cardValue -= 10
                        totalLabels(counter).Text = "Total Card Value: " +
Str(playersPlaying(counter).cardValue)
                        cAce.setdealtPosition(-1)

                    ElseIf playersPlaying(counter).cardValue > 21 And
sAce.getdealtPosition = counter Then

                        sAce.setValue(1)
                        playersPlaying(counter).cardValue -= 10
                        totalLabels(counter).Text = "Total Card Value: " +
Str(playersPlaying(counter).cardValue)
                        sAce.setdealtPosition(-1)
```

```
                    ElseIf playersPlaying(counter).cardValue > 21 And
dAce.getdealtPosition = counter Then

                        dAce.setValue(1)
                        playersPlaying(counter).cardValue -= 10
                        totalLabels(counter).Text = "Total Card Value: " +
Str(playersPlaying(counter).cardValue)
                        dAce.setdealtPosition(-1)

                    ElseIf playersPlaying(counter).cardValue > 21 And
hAce.getdealtPosition = counter Then

                        hAce.setValue(1)
                        playersPlaying(counter).cardValue -= 10
                        totalLabels(counter).Text = "Total Card Value: " +
Str(playersPlaying(counter).cardValue)
                        hAce.setdealtPosition(-1)

                    End If


                    'checks the values of each players cards and changes
their data accordingly
                    If playersPlaying(counter).cardValue = 21 Then
                        playersPlaying(counter).stayed = True
                        inRound -= 1
                        userLabels(counter).Text += "(Stayed)"
                        MsgBox(playersPlaying(counter).username + " has
reached 21!")
```

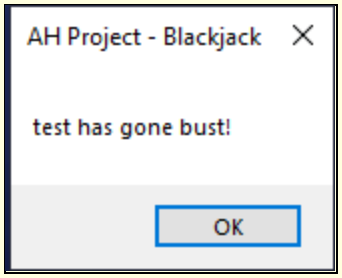



```
                    ElseIf playersPlaying(counter).cardValue <= 21 And
playersPlaying(counter).cardTotal = 5 Then
                        playersPlaying(counter).stayed = True
```

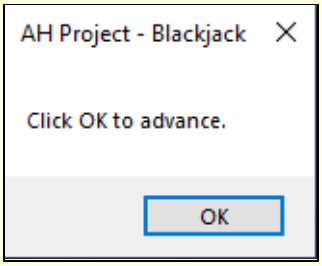
```
                        inRound -= 1
                        userLabels(counter).Text += "(Stayed)"
                        MsgBox(playersPlaying(counter).username + " has got
the 5 card trick!")
```




```
                    ElseIf playersPlaying(counter).cardValue > 21 Then
                        playersPlaying(counter).bust = True
                        inRound -= 1
                        userLabels(counter).Text += "(Bust)"
                        MsgBox(playersPlaying(counter).username + " has gone
bust!")
```



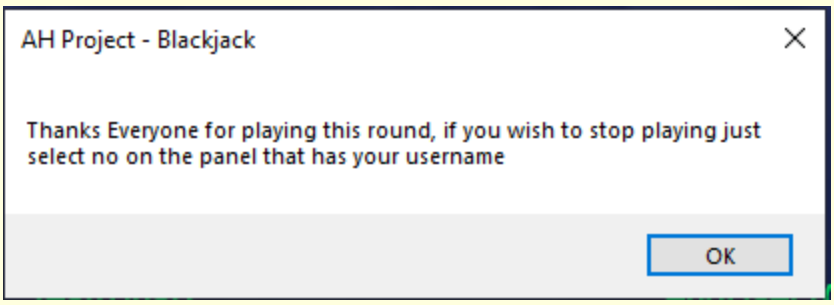

```
                    End If

                End If

            Next

        End While

        MsgBox("Click OK to advance.")
```



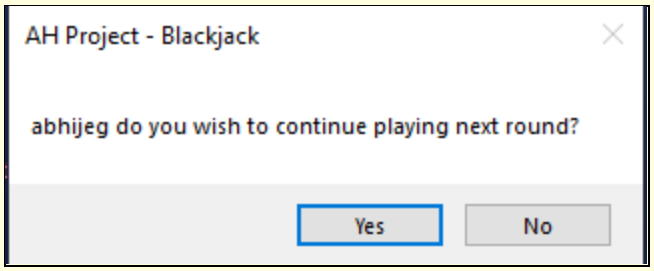
```
'gives the dealer a second card while flipping the face down card over
        backofCard.Hide()
        dealerInfo.cardValue += deckOfCards(cardsOut).getValue
        deckOfCards(cardsOut).showCard(dealerPositions(1, 0),
dealerPositions(1, 1))
        cardsOut += 1
        dealerInfo.cardTotal += 1
        dealerTotal.Text = "Current Card Total : " +
Str(dealerInfo.cardValue)

        'checks if the dealers cards are over 21 and if there is an ace among
them the value is changed from 11 to 1
        If dealerInfo.cardValue > 21 And cAce.getdealtPosition = 100 Then

            cAce.setValue(1)
            dealerInfo.cardValue -= 10
            dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
            cAce.setdealtPosition(-1)

        ElseIf dealerInfo.cardValue > 21 And sAce.getdealtPosition = 100 Then

            sAce.setValue(1)
            dealerInfo.cardValue -= 10
            dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
            sAce.setdealtPosition(-1)

        ElseIf dealerInfo.cardValue > 21 And dAce.getdealtPosition = 100 Then

            dAce.setValue(1)
            dealerInfo.cardValue -= 10
            dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
            dAce.setdealtPosition(-1)

        ElseIf dealerInfo.cardValue > 21 And hAce.getdealtPosition = 100 Then

            hAce.setValue(1)
            dealerInfo.cardValue -= 10
            dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
```

```
            hAce.setdealtPosition(-1)

        End If

        'conditional loop that checks the value of the dealers cards and if
it is 16 or below it is
        While dealerInfo.cardValue <= 16 And dealerInfo.cardTotal < 5
            dealerInfo.cardValue += deckOfCards(cardsOut).getValue

deckOfCards(cardsOut).showCard(dealerPositions(dealerInfo.cardTotal, 0),
dealerPositions(dealerInfo.cardTotal, 1))

            If cAce.getdealt() = True And cAce.getdealtPosition() = -1 Then
                cAce.setdealtPosition(100)
                cAce.setdealt(False)

            ElseIf sAce.getdealt() = True And sAce.getdealtPosition() = -1
Then
                sAce.setdealtPosition(100)
                sAce.setdealt(False)

            ElseIf hAce.getdealt = True And hAce.getdealtPosition() = -1 Then
                hAce.setdealtPosition(100)
                hAce.setdealt(False)

            ElseIf dAce.getdealt = True And dAce.getdealtPosition() = -1 Then
                dAce.setdealtPosition(100)
                dAce.setdealt(False)

            End If

            cardsOut += 1
            dealerInfo.cardTotal += 1

            If dealerInfo.cardValue > 21 And cAce.getdealtPosition = 100 Then

                cAce.setValue(1)
                dealerInfo.cardValue -= 10
                dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
                cAce.setdealtPosition(-1)
```

```
            ElseIf dealerInfo.cardValue > 21 And sAce.getdealtPosition = 100
Then

                sAce.setValue(1)
                dealerInfo.cardValue -= 10
                dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
                sAce.setdealtPosition(-1)

            ElseIf dealerInfo.cardValue > 21 And dAce.getdealtPosition = 100
Then

                dAce.setValue(1)
                dealerInfo.cardValue -= 10
                dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
                dAce.setdealtPosition(-1)

            ElseIf dealerInfo.cardValue > 21 And hAce.getdealtPosition = 100
Then

                hAce.setValue(1)
                dealerInfo.cardValue -= 10
                dealerTotal.Text = "Total Card Value: " +
Str(dealerInfo.cardValue)
                hAce.setdealtPosition(-1)

            End If
            dealerTotal.Text = "Total Card Value : " +
Str(dealerInfo.cardValue)

        End While

        'procedure that runs through every possible outcome concerning the
dealer's card value and compares it to the players, causing them to win or
lose their bets accordingly
        compareDealerCards(dealerInfo)

        MsgBox("Thanks Everyone for playing this round, if you wish to stop
playing just select no on the panel that has your username")
```

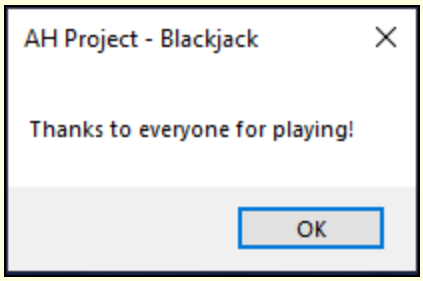



```
        ' fixed loop that updates the database with the new balance of each player and the number of games played
        For counter = 0 To UBound(playersPlaying)
            If playersPlaying(counter).playing = True Then
                updateDatabase(conn, sqlReader, counter)
            End If
        Next

        'fixed loop and shows a pop up window that gives the player the option to stop playing or continue playing
        For counter = 0 To UBound(playersPlaying)

            If playersPlaying(counter).playing = True Then
                Dim answer = MsgBox(playersPlaying(counter).username + " do you wish to continue playing next round?", vbYesNo)
```




```
                If answer = vbNo Then
                    playersPlaying(counter).playing = False

                End If
            End If
        Next

        'checks if there is at least one player still playing
        Dim continuePlaying As Boolean = False
        For counter = 0 To UBound(playersPlaying)
```

```
            If playersPlaying(counter).playing = True Then
                continuePlaying = True
            End If

        Next

        'if there is one player still playing the game can continue but if no
one is playing this statement causes the form the close
        If continuePlaying = False Then
            MsgBox("Thanks to everyone for playing!")
```




```
            hidingEverything()
            ReDim playersPlaying(0)
            Me.Size = New Size(400, 190)
            usernameInput.Show()
            passInput.Show()
            gameSignUser.Show()
            gameSignPass.Show()
            signInGame.Show()
            LinkLabel1.Show()
            LinkLabel2.Show()
            Me.Hide()
            mainMenu.Show()
            signInCounter = 0
        End If

        'restarts the form, setting it to how it was with the startround and
endround buttons
        conn.Close()
        hidingEverything()
        startRound.Show()
        endGame.Show()

    End Sub
```

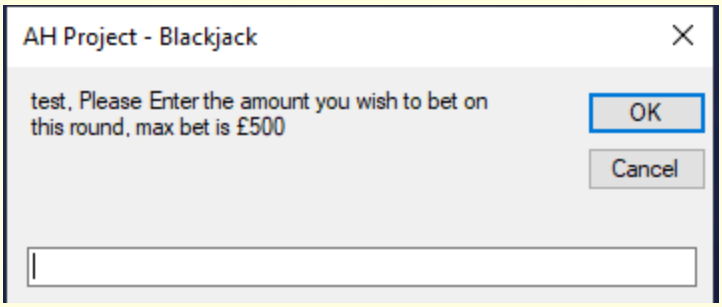
```
    'procedure that sets a 2d array of values for coordinates
    Private Sub inputCoordinates(ByRef player(,) As Integer, ByRef dealer(,)
As Integer)

        player(0, 0) = 30
        player(0, 1) = 466

        player(1, 0) = 49
        player(1, 1) = 466

        player(2, 0) = 68
        player(2, 1) = 466

        player(3, 0) = 87
        player(3, 1) = 466

        player(4, 0) = 106
        player(4, 1) = 466

        player(5, 0) = 346
        player(5, 1) = 466

        player(6, 0) = 365
        player(6, 1) = 466

        player(7, 0) = 384
        player(7, 1) = 466

        player(8, 0) = 403
        player(8, 1) = 466

        player(9, 0) = 422
        player(9, 1) = 466

        player(10, 0) = 662
        player(10, 1) = 466

        player(11, 0) = 681
        player(11, 1) = 466

        player(12, 0) = 700
```

```
player(12, 1) = 466

player(13, 0) = 719
player(13, 1) = 466

player(14, 0) = 738
player(14, 1) = 466

player(15, 0) = 978
player(15, 1) = 466

player(16, 0) = 997
player(16, 1) = 466

player(17, 0) = 1016
player(17, 1) = 466

player(18, 0) = 1035
player(18, 1) = 466

player(19, 0) = 1054
player(19, 1) = 466

player(20, 0) = 1294
player(20, 1) = 466

player(21, 0) = 1313
player(21, 1) = 466

player(22, 0) = 1332
player(22, 1) = 466

player(23, 0) = 1351
player(23, 1) = 466

player(24, 0) = 1370
player(24, 1) = 466


dealer(0, 0) = 657
dealer(0, 1) = 79
```

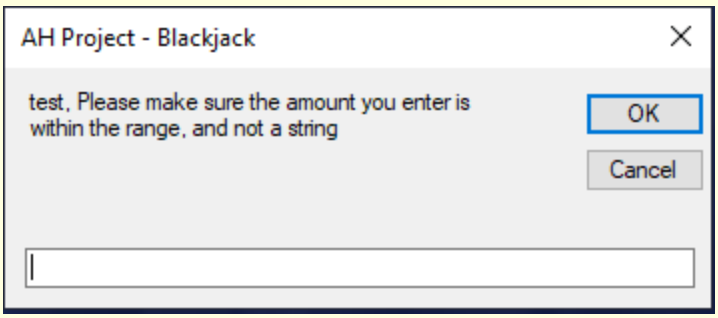
```
        dealer(1, 0) = 757
        dealer(1, 1) = 79

        dealer(2, 0) = 857
        dealer(2, 1) = 79

        dealer(3, 0) = 957
        dealer(3, 1) = 79

        dealer(4, 0) = 1057
        dealer(4, 1) = 79

    End Sub

    ' at the start of each round sets the values in the playersplaying array
of records back to the defaults
    Private Sub resetRound()

        For counter = 0 To UBound(playersPlaying)

            playersPlaying(counter).bet = 0
            playersPlaying(counter).bust = False
            playersPlaying(counter).stayed = False
            playersPlaying(counter).cardTotal = 0
            playersPlaying(counter).cardValue = 0
            dealerTotal.Text = "Current Card Total:"

        Next
    End Sub

    'displays the names and other info of players in the game
    Private Sub displayLabels(ByRef name As Label, ByRef balance As Label,
ByRef bet As Label, ByRef total As Label, ByVal counter As Integer)


        name.Show()
        name.Text = playersPlaying(counter).username

        balance.Show()
        balance.Text = "Account Balance: £" +
Str(playersPlaying(counter).balance)
```

```
        bet.Show()
        bet.Text = "Current Bet: "

        total.Show()
        total.Text = "Total Card Value: 0"




    End Sub


'randomises the order of the playing cards in the array they are stored in
    Private Sub shuffleCards(ByRef deck() As cards)

        Dim rnd As New Random()
        Dim j As Integer
        Dim temp As cards

        For counter = 0 To UBound(deck)
            j = rnd.Next(0, UBound(deck))

            temp = deck(counter)
            deck(counter) = deck(j)
            deck(j) = temp

        Next

        For counter = 0 To UBound(deck)
            j = rnd.Next(0, UBound(deck))

            temp = deck(counter)
            deck(counter) = deck(j)
            deck(j) = temp

        Next

        For counter = 0 To UBound(deck)
            j = rnd.Next(0, UBound(deck))

            temp = deck(counter)
            deck(counter) = deck(j)
```

```
            deck(j) = temp

        Next
    End Sub

    'allows the players to place heir bets
    Private Sub placeBets(ByRef i As Integer, ByRef bet As Label, ByRef
balance As Label)

        Dim valid As Boolean
        Dim num As Integer = 0
        Do
            valid = True

            Try
                If num = 0 Then
                    playersPlaying(i).bet =
InputBox(playersPlaying(i).username + ", Please Enter the amount you wish to
bet on this round, max bet is £500")
```



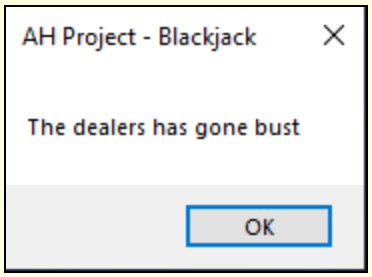

```
                Else
                    playersPlaying(i).bet =
InputBox(playersPlaying(i).username + ", Please make sure the amount you
enter is within the range, and not a string")
```



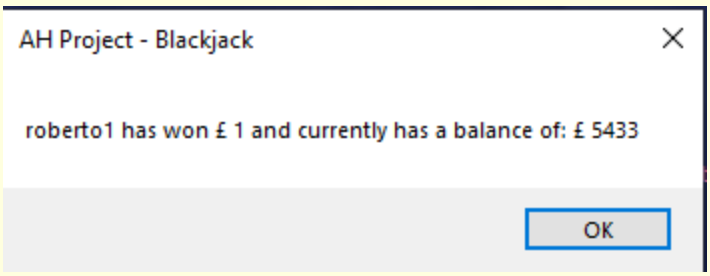

```
                End If

            Catch ex As Exception
```

```
                valid = False

            End Try


            If playersPlaying(i).bet > 500 Or playersPlaying(i).bet >
playersPlaying(i).balance Then
                valid = False

            End If

            If valid = False Then
                num = 1
            End If
        Loop Until valid = True

        playersPlaying(i).balance -= playersPlaying(i).bet

        bet.Text += "£" + Str(playersPlaying(i).bet)
        balance.Text = "Account Balance : £" + Str(playersPlaying(i).balance)

    End Sub

    'displays the cards in the correct position and updates the values in the
playersplaying array of records that need to be changed
    Private Sub dealCards(ByRef cardsOut As Integer, ByVal i As Integer,
ByRef positions(,) As Integer, ByRef deck() As cards, ByRef total As Label)

        Dim positionElement As Integer = (i * 5) +
playersPlaying(i).cardTotal

        playersPlaying(i).cardValue += deck(cardsOut).getValue()
        deck(cardsOut).showCard(positions(positionElement, 0),
positions(positionElement, 1))

        cardsOut += 1
        playersPlaying(i).cardTotal += 1

        total.Text = "Total Card Value : " + Str(playersPlaying(i).cardValue)

    End Sub
```

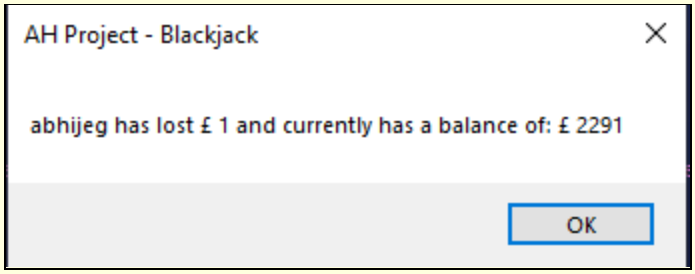
```
    'updates the balance and games played of each player at the end of each
round
    Private Sub updateDatabase(ByVal conn As OleDbConnection, ByVal reader As
OleDbDataReader, ByVal counter As Integer)

        Dim query As String = "UPDATE [players] SET [accountBalance] = " +
Str(playersPlaying(counter).balance) + ", [gamesPlayed] = [gamesPlayed] + 1
WHERE [username] = '" + playersPlaying(counter).username + "';"

        Dim CommandInfo As New OleDbCommand(query, conn)
        reader = CommandInfo.ExecuteReader


    End Sub

    'compares the dealers cards to every players cards
    Private Sub compareDealerCards(ByRef dealerinfo As dealerData)

        ' this list of "if" statements will compare the dealers cards to each
players and decide which player has won and which has lost

        'if the dealer has gone bust
        If dealerinfo.cardValue > 21 Then
            MsgBox("The dealers has gone bust")
```




```
            For counter = 0 To UBound(playersPlaying)

                If playersPlaying(counter).bust = False And
playersPlaying(counter).playing = True Then
                    playersPlaying(counter).balance += (2 *
playersPlaying(counter).bet)
                    MsgBox(playersPlaying(counter).username + " has won £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of: £" +
Str(playersPlaying(counter).balance))
```




```
                ElseIf playersPlaying(counter).playing = True And
playersPlaying(counter).bust = True Then
                    MsgBox(playersPlaying(counter).username + " has lost £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of: £" +
Str(playersPlaying(counter).balance))
```




```
                End If
            Next


            'if the dealer has blackjack
        ElseIf dealerinfo.cardValue = 21 And dealerinfo.cardTotal = 2 Then

            For counter = 0 To UBound(playersPlaying)

                If playersPlaying(counter).cardTotal = 2 And
playersPlaying(counter).cardValue = 21 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance +=
playersPlaying(counter).bet
                    MsgBox(playersPlaying(counter).username + " has drawn
with the dealer and will retain their balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).playing = True Then
                    MsgBox(playersPlaying(counter).username + " has lost £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of: £ " +
```

```
Str(playersPlaying(counter).balance))

                End If
            Next


            'if the dealer has a 5 card trick
        ElseIf dealerinfo.cardValue <= 21 And dealerinfo.cardTotal = 5 Then

            For counter = 0 To UBound(playersPlaying)

                If playersPlaying(counter).cardValue = 21 And
playersPlaying(counter).cardTotal = 2 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance += (2 *
playersPlaying(counter).bet)
                    MsgBox(playersPlaying(counter).username + " has won £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).cardValue <= 21 And
playersPlaying(counter).cardTotal = 5 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance +=
playersPlaying(counter).bet
                    MsgBox(playersPlaying(counter).username + " has drawn
with the dealer and will retain their balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).playing = True Then
                    MsgBox(playersPlaying(counter).username + " has lost £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of : £" +
Str(playersPlaying(counter).balance))

                End If
            Next


            'if the dealer has 21 but not blackjack
        ElseIf dealerinfo.cardValue = 21 And dealerinfo.cardTotal > 2 Then

            For counter = 0 To UBound(playersPlaying)
```

```
                If playersPlaying(counter).cardValue = 21 And
playersPlaying(counter).cardTotal = 2 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance += (2 *
playersPlaying(counter).bet)
                    MsgBox(playersPlaying(counter).username + " has won £" +
Str(playersPlaying(counter).bet) + " with a Blackjack, and currently has a
balance of: £" + Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).cardValue <= 21 And
playersPlaying(counter).cardTotal = 5 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance += (2 *
playersPlaying(counter).bet)
                    MsgBox(playersPlaying(counter).username + " has won £" +
Str(playersPlaying(counter).bet) + " with a 5 card trick, and currently has a
balance of: £" + Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).cardValue = 21 And
playersPlaying(counter).cardTotal > 2 And playersPlaying(counter).playing =
True Then
                    playersPlaying(counter).balance +=
playersPlaying(counter).bet
                    MsgBox(playersPlaying(counter).username + " has drawn
with the dealer and will retain their balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).playing = True Then
                    MsgBox(playersPlaying(counter).username + " has lost £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of : £" +
Str(playersPlaying(counter).balance))

                End If
            Next


            'if the dealer has anything else
        ElseIf dealerinfo.cardValue <= 21 And dealerinfo.cardTotal < 5 Then

            For counter = 0 To UBound(playersPlaying)
```

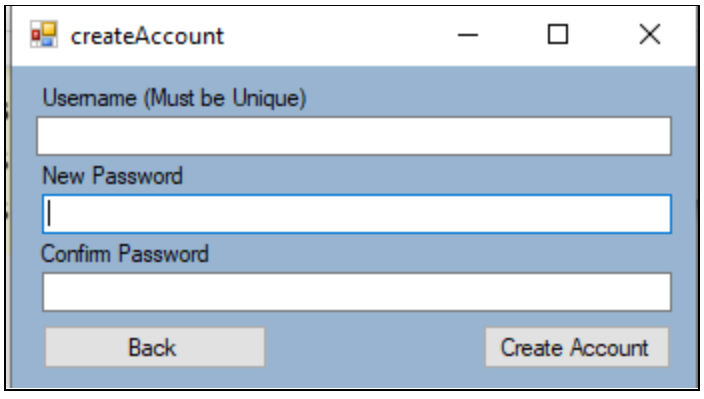
```
                If (playersPlaying(counter).cardValue > dealerinfo.cardValue
And playersPlaying(counter).bust = False And playersPlaying(counter).playing
= True) Or (playersPlaying(counter).cardValue <= 21 And
playersPlaying(counter).cardTotal = 5 And playersPlaying(counter).playing =
True) Then
                    playersPlaying(counter).balance += (2 *
playersPlaying(counter).bet)
                    MsgBox(playersPlaying(counter).username + " has won £" +
Str(playersPlaying(counter).bet) + ", and currently has a balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf playersPlaying(counter).cardValue =
dealerinfo.cardValue Then
                    playersPlaying(counter).balance +=
playersPlaying(counter).bet
                    MsgBox(playersPlaying(counter).username + " has drawn
with the dealer and will retain their balance of: £" +
Str(playersPlaying(counter).balance))

                ElseIf (playersPlaying(counter).cardValue <
dealerinfo.cardValue And playersPlaying(counter).cardTotal < 5 And
playersPlaying(counter).playing = True) Or (playersPlaying(counter).bust =
True And playersPlaying(counter).playing = True) Then
                    MsgBox(playersPlaying(counter).username + " has lost £" +
Str(playersPlaying(counter).bet) + " and currently has a balance of : £" +
Str(playersPlaying(counter).balance))

                End If
            Next
        End If
    End Sub

End Class




' class used to store data of each card
Public Class cards
```

```
    'properties of the card class
    Private suit As String
    Private value As Integer
    Public pictureBox1 As PictureBox

    'constructor method for cards classes
    Public Sub New(ByVal suitN As String, ByVal valueN As Integer, ByVal
picture As PictureBox)

        suit = suitN
        value = valueN
        pictureBox1 = picture

    End Sub

    'setters
    Public Sub setSuit(ByVal value As String)

        suit = value

    End Sub

    Public Sub setValue(ByVal value1 As Integer)

        value = value1

    End Sub
    'getters
    Public Function getSuit()
        Return suit

    End Function

    Public Function getValue()
        Return value

    End Function
    ' this piece of code takes a picture box that is passed in as an actual
parameter and displays it
    Public Overridable Sub showCard(ByVal val1 As Integer, ByVal val2 As
Integer)
```

```
        pictureBox1.BringToFront()
        pictureBox1.Show()
        pictureBox1.SetBounds(val1, val2, pictureBox1.Width,
pictureBox1.Height)


    End Sub
End Class

'seperate classes for the aces as they need additional methods and properties
Public Class aces
    'subclass of cards, inherits all properties and methods
    Inherits cards

    Private secondValue As Integer
    Private dealtPosition As Integer
    Private dealt As Boolean

    'constructor method of aces, inherits properties from cards and are used
here
    Public Sub New(ByVal suitN As String, ByVal valueN As Integer, ByVal
picture As PictureBox, ByVal value2N As Integer, ByVal dealtN As Boolean,
ByVal dealtP As Integer)

        MyBase.New(suitN, valueN, picture)
        secondValue = value2N
        dealt = dealtN
        dealtPosition = dealtP

    End Sub

    'setters
    Public Sub setValue2(ByVal value As Integer)
        secondValue = value

    End Sub
    Public Sub setdealt(ByVal value As Boolean)
        dealt = value

    End Sub
    Public Function setdealtPosition(ByVal value As Integer)
```

```
        dealtPosition = value

    End Function
    'getters
    Public Function getValue2()
        Return secondValue

    End Function

    Public Function getdealt()
        Return dealt

    End Function

    Public Function getdealtPosition()
        Return dealtPosition

    End Function

    'overridden method from cards class, adds extra features to method
    Public Overrides Sub showCard(val1 As Integer, val2 As Integer)
        MyBase.showCard(val1, val2)

        pictureBox1.BringToFront()
        pictureBox1.Show()
        pictureBox1.SetBounds(val1, val2, pictureBox1.Width,
pictureBox1.Height)
        dealt = True

    End Sub
End Class
```

Create Account Form UI

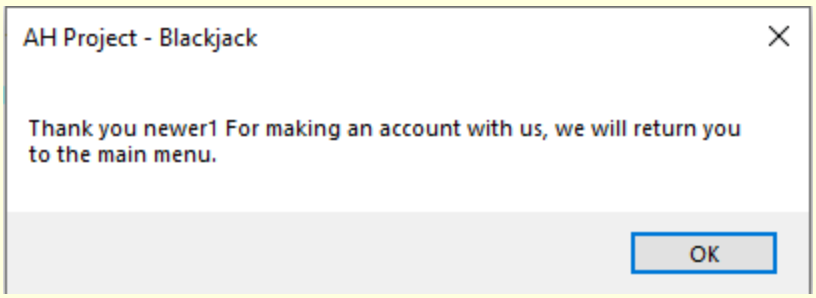


Create Account Form Code

```
Imports System.Data.OleDb

Public Class createAccount
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        'initialising variables
        Dim newUser As String
        Dim newPass As String
        Dim confirmPass As String

        Dim userNames() As String

        Dim sqlReader As OleDbDataReader
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
Higher Project.accdb"

        'initialising connection to database
        Dim conn As OleDbConnection
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()

        ' finds how many elements are supposed to be in array and
re-initialises the array to fit the correct amount
        ReDim userNames(elementsinArray("SELECT Count(username) AS [Num]
FROM [Players];", conn, sqlReader))

        ' enters all usernames already in the database into an array
        takenUsernames("SELECT username FROM [Players];", conn, sqlReader,
userNames)

        newUser = newUsertxt.Text
        newPass = newPasstxt.Text
        confirmPass = confirmPasstxt.Text

        userValidation(newUser, newPass, confirmPass, userNames, conn)
```

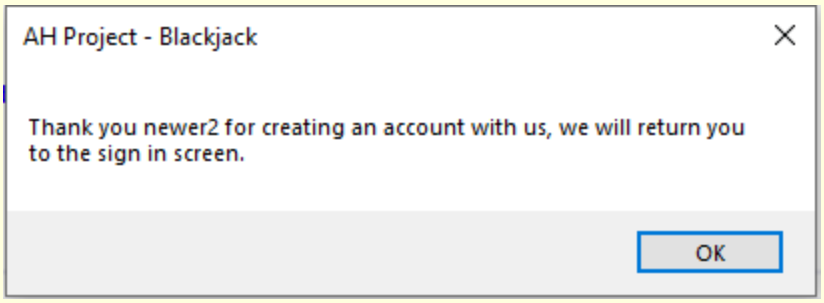
```
        conn.Close()

    End Sub

    'calculates the number of elements in the database (unnecessary but was
a stopgap at the time)
    Private Function elementsinArray(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader) As Integer

        Dim commandNum As New OleDbCommand(query, conn)
        reader = commandNum.ExecuteReader

        Dim elements As Integer
        If reader.HasRows Then
            While reader.Read

                elements = reader("Num")

            End While
        End If

        Return elements - 1
    End Function

    'enters the usernames already in the database into an array to compare
them to the new username that is being entered
    Private Sub takenUsernames(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader, ByRef user() As String)


        Dim commandUsernames As New OleDbCommand(query, conn)
        reader = commandUsernames.ExecuteReader

        If reader.HasRows Then

            Dim num As Integer = 0
            While reader.Read

                user(num) = reader("username")
                num += 1
            End While
        End If
```

```
    End Sub

    'checks if the created username and password are valid (unique
username)
    Private Sub userValidation(ByVal newuser As String, ByVal newpass As
String, ByVal confirmpass As String, ByRef user() As String, ByVal conn As
OleDbConnection)

        Dim approvedUser As Boolean = True
        Dim approvedpass As Boolean = True

        For counter = 0 To UBound(user)

            If newuser = user(counter) Then
                approvedUser = False
            End If
        Next
        If newpass <> confirmpass Then
            approvedpass = False
        End If

        If approvedUser = True And approvedpass = True Then
            Dim queryInsert As String = "INSERT INTO [Players] ([username],
[password], [accountBalance], [gamesPlayed]) VALUES ('" & newuser & "', '"
& newpass & "', 5000, 0);"
            Dim commandInsert As New OleDbCommand(queryInsert, conn)
            commandInsert.ExecuteReader()


            If mainMenu.clickTypeCreate = "menu" Then
                MsgBox("Thank you " + newuser + " For making an account
with us, we will return you to the main menu.")
```

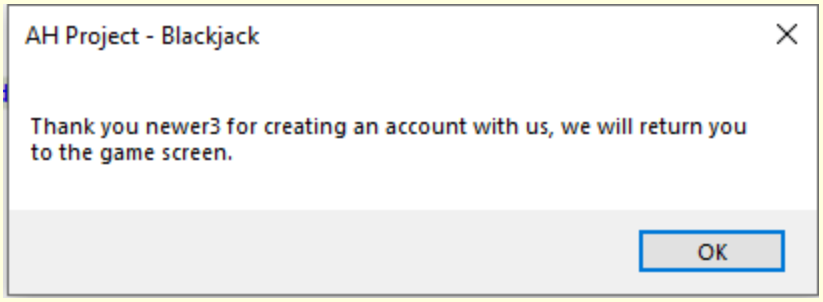



```
                Me.Hide()
```

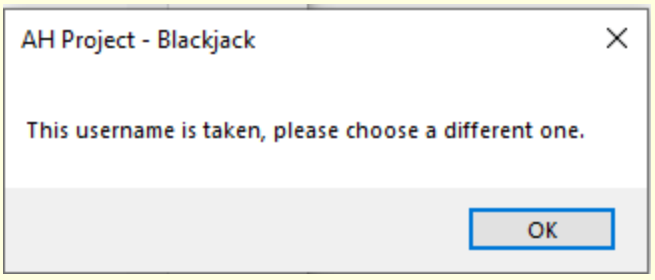
```
                mainMenu.Show()
            ElseIf mainMenu.clickTypeCreate = "sign" Then
                MsgBox("Thank you " + newuser + " for creating an account
with us, we will return you to the sign in screen.")
```



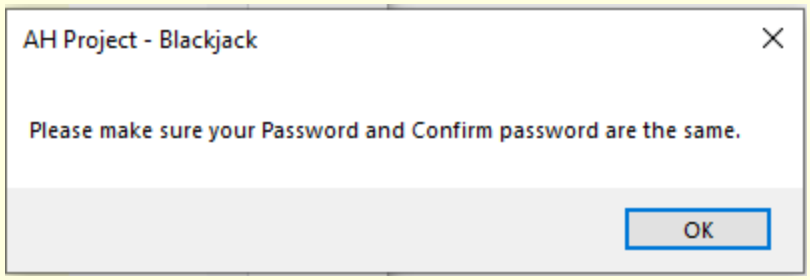

```
                Me.Hide()
                signIn.Show()
            ElseIf mainMenu.clickTypecreate = "game" Then
                MsgBox("Thank you " + newuser + " for creating an account
with us, we will return you to the game screen.")
```



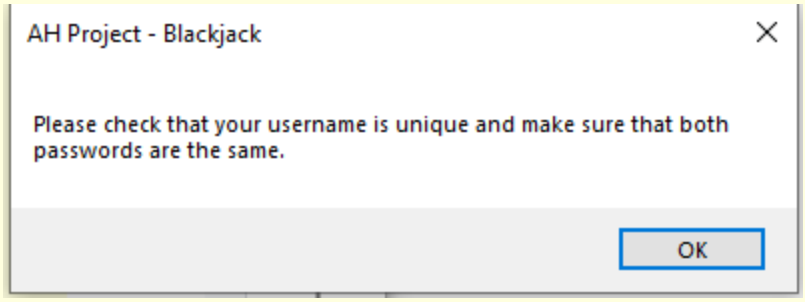

```
                Me.Hide()
                gameUI.Size = New Size(490, 190)
            End If
        ElseIf approvedUser = False And approvedpass = True Then
            MsgBox("This username is taken, please choose a different
one.")
```




```
ElseIf approvedUser = True And approvedpass = False Then
            MsgBox("Please make sure your Password and Confirm password are
the same.")
```

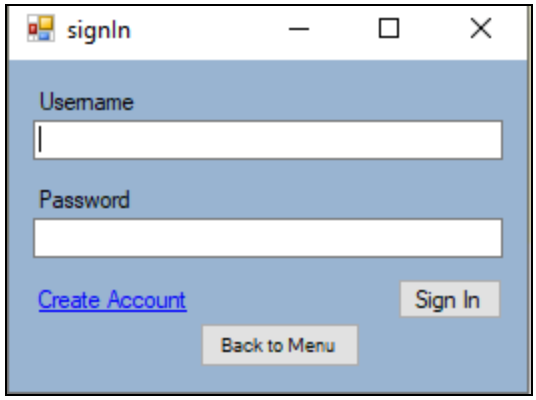


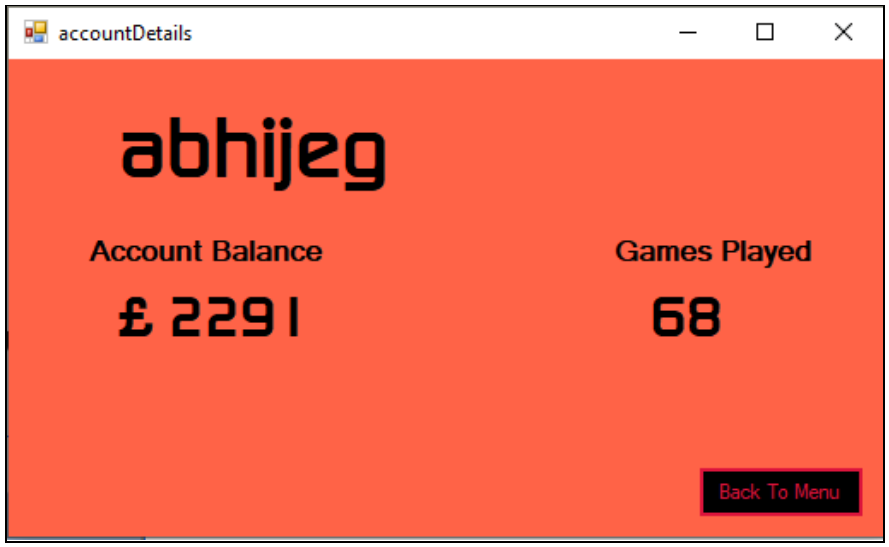

```
        ElseIf approvedUser = False And approvedpass = False Then
            MsgBox("Please check that your username is unique and make sure
that both passwords are the same.")
```




```
        End If

    End Sub

    'if the user doesn't want to create an account this button lets them go
back to the form they were on previously
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click

        If mainMenu.clickTypeCreate = "menu" Then
            Me.Hide()
            mainMenu.Show()
        ElseIf mainMenu.clickTypeCreate = "game" Then
            Me.Hide()

        ElseIf mainMenu.clickTypeCreate = "sign" Then
            Me.Hide()
            Dim signIn As New signIn
            signIn.Show()
        End If


    End Sub
End Class
```

Sign In and Account Details Form UI



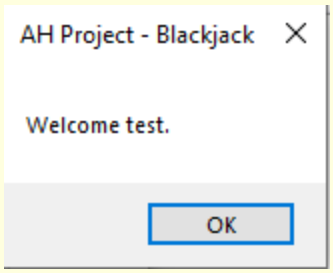


Sign In Form Code

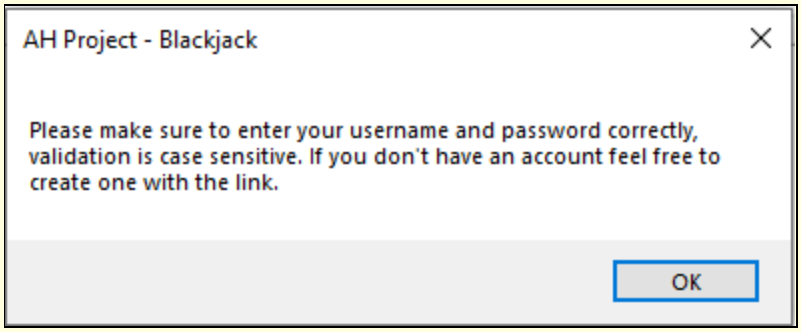
```
Imports System.Data.OleDb
'array of records to hold username and password for input validation
Structure ValidationData

    Dim usernameCheck As String
    Dim passwordCheck As String

End Structure
Public Class signIn
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        'initialising variables
        Dim SignValidation() As ValidationData

        Dim sqlReader As OleDbDataReader
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
Higher Project.accdb"
```

```
        'initialising connection to database
        Dim conn As OleDbConnection
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()

        'function finds the number of elements in the database and sets up
the array to hold that many values
        ReDim SignValidation(elementsinArray("SELECT Count(username) AS
[Num] FROM [Players];", conn, sqlReader))

        ' enters all usernames and passwords into an array of records in
order to validate user inputs
        userANDPasswords("SELECT [username], [password] FROM [Players];",
conn, sqlReader, SignValidation)
        conn.Close()

        Dim signInUser As String = usertxt.Text
        Dim signInPass As String = passtxt.Text

        validateInputs(signInUser, signInPass, SignValidation)


    End Sub
    Private Function elementsinArray(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader) As Integer

        Dim commandNum As New OleDbCommand(query, conn)
        reader = commandNum.ExecuteReader

        Dim elements As Integer
        If reader.HasRows Then
            While reader.Read

                elements = reader("Num")

            End While
        End If

        Return elements - 1
    End Function

    Private Sub userANDPasswords(ByVal query As String, ByVal conn As
```

```
OleDbConnection, ByVal reader As OleDbDataReader, ByRef validation() As
ValidationData)


        Dim commandUsernames As New OleDbCommand(query, conn)
        reader = commandUsernames.ExecuteReader

        If reader.HasRows Then

            Dim num As Integer = 0
            While reader.Read

                validation(num).usernameCheck = reader("username")
                validation(num).passwordCheck = reader("password")
                num += 1
            End While
        End If

    End Sub

    'checks values entered against the values in the database in order to
validate the inputs
    Private Sub validateInputs(ByVal user As String, ByVal pass As String,
ByRef validation() As ValidationData)


        Dim Check As Boolean = False


        For counter = 0 To UBound(validation)
            If user = validation(counter).usernameCheck And pass =
validation(counter).passwordCheck Then
                Check = True
            End If
        Next

        If Check = True Then
            If mainMenu.clickTypeSign = "details" Then
                MsgBox("Welcome " + user + ".")
```




```
                Dim accountDetails1 As New accountDetails
                accountDetails1.accountDetailsUsername = user
                Me.Hide()
                accountDetails1.Show()
            End If
        Else
            MsgBox("Please make sure to enter your username and password
correctly, validation is case sensitive. If you don't have an account feel
free to create one with the link.")
```




```
        End If
    End Sub

    'takes user to createAccount form
    Private Sub CreateAccount_LinkClicked(sender As Object, e As
LinkLabelLinkClickedEventArgs) Handles createAccount.LinkClicked

        Me.Hide()
        Dim createSign As New createAccount
        createSign.Show()
        mainMenu.clickTypeCreate = "sign"

    End Sub
```

```
    'takes user to mainMenu
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click

        Me.Hide()
        mainMenu.Show()

    End Sub

    Private Sub SignIn_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

    End Sub
End Class
```

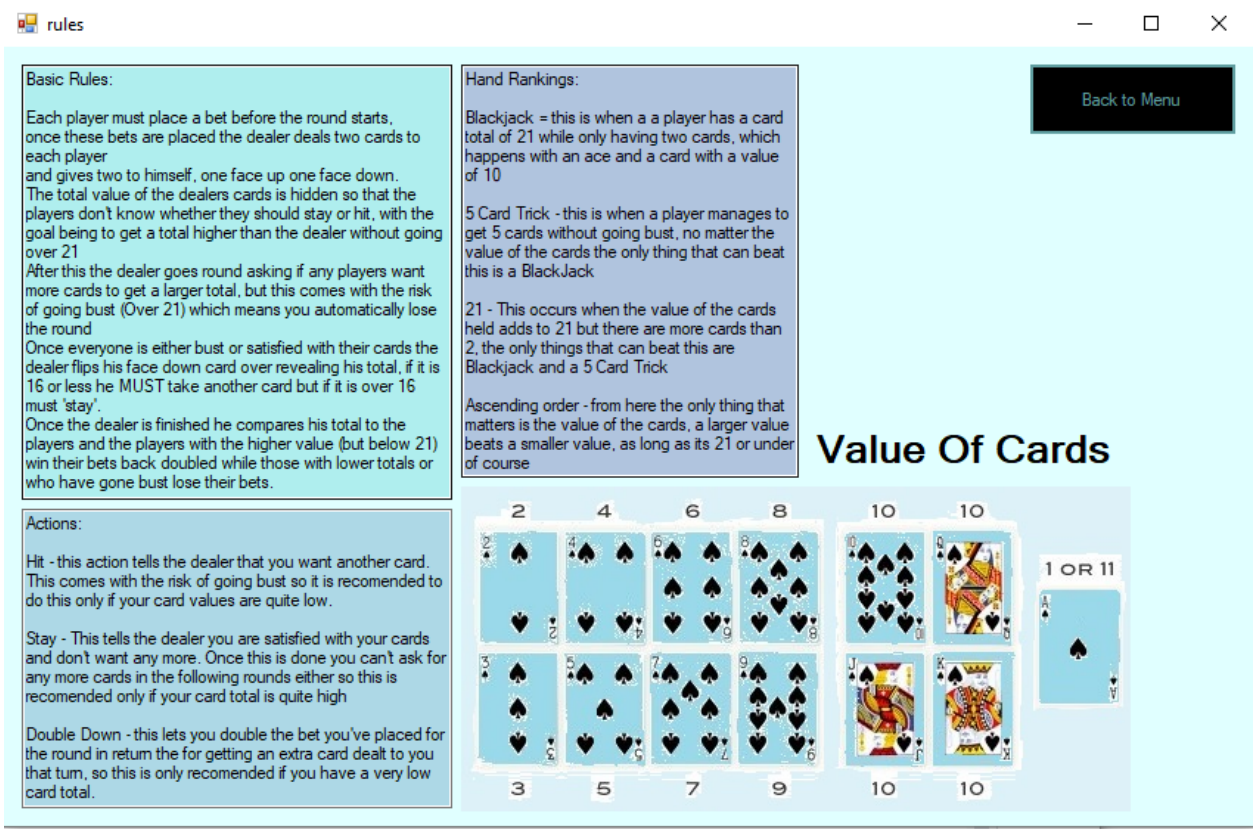
Account Details Form Code

```
Imports System.Data.OleDb
Public Class accountDetails
    Public Property accountDetailsUsername

    Private Sub AccountDetails_Load(sender As Object, e As EventArgs)
Handles MyBase.Load

        Dim userdetail As String = accountDetailsUsername
        Dim accountbalancedetail As Integer
        Dim gamesPlayedDetail As Integer


        Dim sqlReader As OleDbDataReader
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
Higher Project.accdb"

        'initialising connection to database
        Dim conn As OleDbConnection
```

```
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()

        findingDetails("SELECT [accountBalance], [gamesPlayed] FROM
[Players] WHERE username = '" + userdetail + "'", conn, sqlReader,
accountbalancedetail, gamesPlayedDetail)

        displayDetails(userdetail, accountbalancedetail, gamesPlayedDetail)

    End Sub

    Private Sub findingDetails(ByVal query As String, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader, ByRef balance As Integer,
ByRef played As Integer)


        Dim commandUsernames As New OleDbCommand(query, conn)
        reader = commandUsernames.ExecuteReader

        If reader.HasRows Then

            While reader.Read

                balance = reader("accountBalance")
                played = reader("gamesPlayed")

            End While
        End If

    End Sub

    Private Sub displayDetails(ByVal user As String, ByVal balance As
Integer, ByVal played As Integer)

        Label1.Text = user
        Label2.Text = "£" + Str(balance)
        Label3.Text = Str(played)

    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
```

```
        Me.Hide()
        mainMenu.Show()

    End Sub

    Private Sub Label2_Click(sender As Object, e As EventArgs) Handles
Label2.Click

    End Sub
End Class
```

Rules Form UI

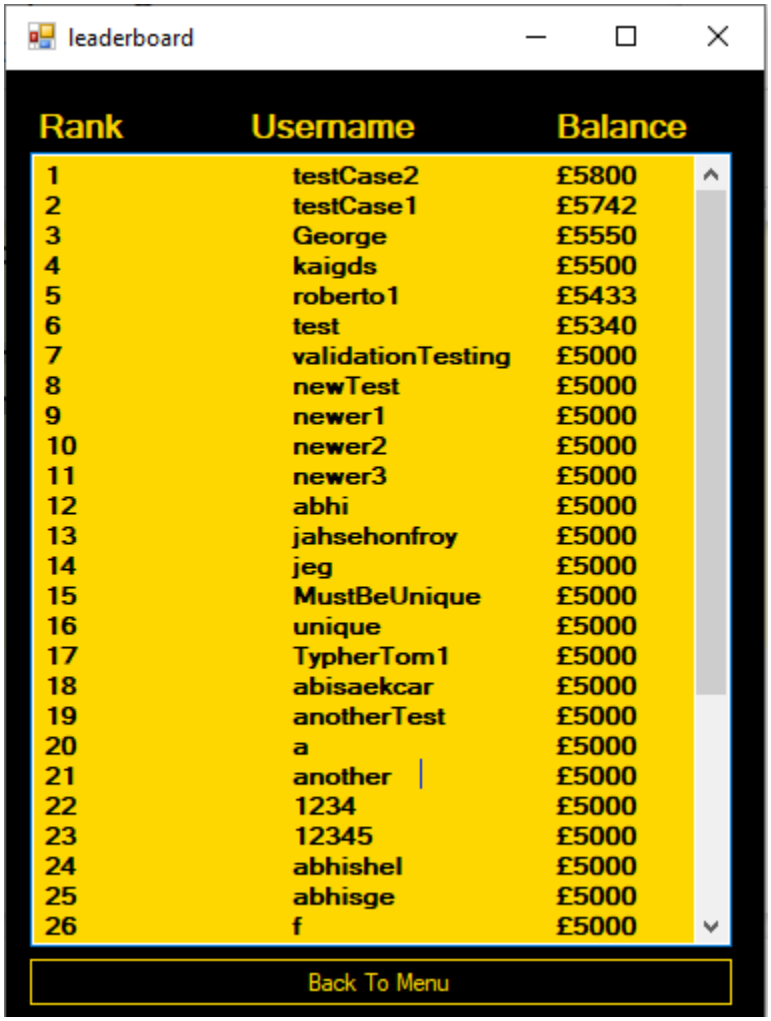


Rules Form Code

```
Public Class rules
    Private Sub Rules_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        'displays information in textboxes about the rules and the game of
blackjack
        TextBox1.Text += "Basic Rules:" + vbNewLine + vbNewLine + "Each
player must place a bet before the round starts, " + vbNewLine + "once
these bets are placed the dealer deals two cards to each player " +
vbNewLine + "and gives two to himself, one face up one face down." +
vbNewLine + "The total value of the dealers cards is hidden so that the
players don't know whether they should stay or hit, with the goal being to
get a total higher than the dealer without going over 21" + vbNewLine +
"After this the dealer goes round asking if any players want more cards to
get a larger total, but this comes with the risk of going bust (Over 21)
which means you automatically lose the round" + vbNewLine + "Once everyone
is either bust or satisfied with their cards the dealer flips his face down
card over revealing his total, if it is 16 or less he MUST take another
card but if it is over 16 must 'stay'." + vbNewLine + "Once the dealer is
finished he compares his total to the players and the players with the
higher value (but below 21) win their bets back doubled while those with
lower totals or who have gone bust lose their bets."
        TextBox3.Text += "Actions:" + vbNewLine + vbNewLine + "Hit - this
action tells the dealer that you want another card. This comes with the
risk of going bust so it is recommended to do this only if your card values
are quite low." + vbNewLine + vbNewLine + "Stay - This tells the dealer you
are satisfied with your cards and don't want any more. Once this is done
you can't ask for any more cards in the following rounds either so this is
recommended only if your card total is quite high" + vbNewLine + vbNewLine
+ "Double Down - this lets you double the bet you've placed for the round
in return the for getting an extra card dealt to you that turn, so this is
only recommended if you have a very low card total."

        TextBox2.Text += "Hand Rankings:" + vbNewLine + vbNewLine +
"Blackjack = this is when a player has a card total of 21 while only having
two cards, which happens with an ace and a card with a value of 10" +
vbNewLine + vbNewLine + "5 Card Trick - this is when a player manages to
get 5 cards without going bust, no matter the value of the cards the only
```

```
thing that can beat this is a BlackJack" + vbNewLine + vbNewLine + "21 -
This occurs when the value of the cards held adds to 21 but there are more
cards than 2, the only things that can beat this are Blackjack and a 5 Card
Trick" + vbNewLine + vbNewLine + "Ascending order - from here the only
thing that matters is the value of the cards, a larger value beats a
smaller value, as long as its 21 or under of course"

    End Sub
    'takes the user to the main menu
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Me.Hide()
        mainMenu.Show()

    End Sub

End Class
```

Leaderboard Form UI



Leaderboard Form Code

```
Imports System.Data.OleDb

'record structure
Structure leaderboardData

    Dim user As String
    Dim balance As String

End Structure

Public Class leaderboard

    Private Sub Leaderboard_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        'initialises an array of records that holds the data that will be
read in from a database
        Dim leaderInfo() As leaderboardData

        'initialising database related variables
        Dim sqlReader As OleDbDataReader
        Dim connectionType As String = "Provider=Microsoft.ACE.OLEDB.12.0;"
        Dim fileLocation As String = "Data
Source=\\ni1\Redirected\abhi1k\Documents\S6\Computing Science\Project\Adv
Higher Project.accdb"

        'initialising connection to database
        Dim conn As OleDbConnection
        conn = New OleDbConnection(connectionType + fileLocation)
        conn.Open()

        readInValues(leaderInfo, conn, sqlReader)

        'closes connection to database
        conn.Close()

        bubbleSort(leaderInfo)
```

```
        'writes out in a textbox (descending order of balance), the
usernames, ranks and balances of each player in the database
        For counter = 0 To UBound(leaderInfo) - 1
            If Len(leaderInfo(counter).user) >= 10 Then
                TextBox1.Text += Str(counter + 1) + vbTab + vbTab +
leaderInfo(counter).user + vbTab + " £" + leaderInfo(counter).balance +
vbNewLine
            Else
                TextBox1.Text += Str(counter + 1) + vbTab + vbTab +
leaderInfo(counter).user + vbTab + vbTab + " £" +
leaderInfo(counter).balance + vbNewLine
            End If



        Next

    End Sub

    'reads in the values from the database into the leaderInfo array of
records
    Private Sub readInValues(ByRef list() As leaderboardData, ByVal conn As
OleDbConnection, ByVal reader As OleDbDataReader)

        Dim query As String = "SELECT [username], [accountBalance] FROM
[Players];"

        Dim command As New OleDbCommand(query, conn)

        reader = command.ExecuteReader

        If reader.HasRows Then

            Dim num As Integer = 0
            ReDim Preserve list(num)
            While reader.Read
                list(num).user = reader("username")
                list(num).balance = reader("accountBalance")

                num += 1
                ReDim Preserve list(num)
```

```
            End While
        End If
    End Sub

    'sorts the array of records by account balance in descending order
    Private Sub bubbleSort(ByRef list() As leaderboardData)

        Dim swapped As Boolean = True

        While swapped = True
            swapped = False
            For counter = 1 To UBound(list)

                If list(counter - 1).balance < list(counter).balance Then

                    swap(list(counter - 1), list(counter))
                    swapped = True
                End If

            Next



        End While

    End Sub

    'procedure swaps the elements in the array of records so they are
sorted in descending order
    Private Sub swap(ByRef val1 As leaderboardData, ByRef val2 As
leaderboardData)

        Dim temp As leaderboardData

        temp = val1
        val1 = val2
        val2 = temp

    End Sub

    Private Sub BackToMenu_Click(sender As Object, e As EventArgs) Handles
backToMenu.Click
```

```
        Me.Hide()
        mainMenu.Show()

    End Sub
End Class
```

Database

	Field Name	Data Type
🔑	username	Short Text
	password	Short Text
	accountBalance	Number
	gamesPlayed	Number

Players

username	password	accountBalance	gamesPlayed
1234	1234	5000	0
12345	12345	5000	0
a	1234	5000	0
abhi	pass	5000	0
abhijeg	password	2291	68
abhisge	hh	5000	0
abhishek	abhishek	5000	0
abhishel	shel	5000	0
abisaekcar	poss1234	5000	0
another	password	5000	0
anotherTest	pass1234	5000	0
calum	calum	4400	5
Croob	bruh	4999	3
endUser1	password	2643	37
f	f	5000	0
George	george	5550	2
jahsehonfroy	xxx	5000	0
jeg	pass2	5000	0
kaigds	w	5500	3
MustBeUnique	will	5000	0
new	new	5000	0
new1	new1	5000	0
new2	new2	5000	0
newer1	pass	5000	0

Research and development of new skills and/or knowledge (4 marks)

Describe:

- the new skills and/or knowledge that you researched
- how you applied these new skills and/or knowledge to your project

You should reference the resources you used to research and develop these new skills and/or knowledge.

Fisher yates shuffle - is an algorithm that is used to randomise the order of a 1D array of any data type, and I have used it in order to shuffle the array of cards which has been extremely effective. Used StackOverflow, Microsoft VB Website

DropDown menu - in Visual Studio there is no dropdown menu tool, instead you have to combine multiple things, a button and a contextStripMenu, when the button is clicked the menu shows and an option can be selected. This helped me in being able to choose a different number of players for each game I wanted to play and test in the program. Used Youtube

Overriding Methods - this is useful as each subclass has multiple extra properties that need to be taken into account when coding methods for this class, in some cases the old inherited methods need to be updated to suit the new properties better. This has been used in this program by overriding the 'showcard' method in the aces subclass to change the value of a new property that has been added into the aces subclass. Used Microsoft VB Website

Global Variables - at the start of the implementation I was having trouble passing information through different forms but researching global variables helped with this problem, being able to have a variables that is initialised inside one form but can be passed through other forms has helped with many aspects of my program, mostly navigation, and is a useful skill to be able to utilise. Used Youtube.

Picturebox Pre-defined functions - this research helped me in being ables to display the cards that each player had been dealt in the relevant positions when needed, with a picturebox it is not possible to change the position of it and show it at the same time, so research was needed in order to find the correct pre defined function that would allow me to change the positions(setbounds, bringtofront). The second one is so that each new card that is shown is in front of the one before it so the cards of each player appear layered.

Log of ongoing testing (5 marks)

Produce a log of the ongoing testing you carry out during implementation. Your log should include:

- what you are testing
- descriptions of issues you encounter during testing
- descriptions of how you resolve these issues
- lists of references you use to resolve each issue

You could present your log as a table, using the above bullet points as column headings. E.g.

what you are testing	**descriptions of issues you encounter during testing**	**descriptions of how you resolve these issues**	**lists of references you use to resolve each issue**
Whether the link to the database is working.	Problem with sql not working, not able to identify tables and fields.	This is solved by having [] around each table name or each field name	Help from Teacher
Whether the card class is implemented correctly	Cards being displayed at the wrong coordinates or not showing up at all.	Add a method that sets the coordinates and displays the cards at the relevant place, using correct parameters	Stack Overflow, Microsoft VB Website - https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.picturebox?view=windowsdesktop-6.0

Whether the cards are shuffled into random order for each round	The cards having a repeating pattern	Researching a procedure that allows for an array to be shuffled into a completely random order every time (Fisher Yates Shuffle)	Fisher yates Shuffle - Stack Overflow
Whether the player details from the database can be successfully read into an array of records to store their details	Only one element has details read into it even though there is more than one player	Adding a running total in order to add the details from the database to each element that is required.	
Changing the value of an ace card that has been dealt from 11 to 1 when it is necessary	There wasn't a way to tell if an ace had been dealt and where it had been dealt to	Adding a new property and overriding the super-classes 'showcard' method to change the value of this new property, then using conditional statements to check which player has an ace and if their total is above 21.	
Whether the data is being read into the array of records in the gameUI form	There is one element being read in and then the rest aren't	Having a running total keeps track of how many elements have been read in so far and redimensioning the array when a new player is signing in.	

Testing the solution (15 marks)

Once you have fully implemented your design, you must carry out final testing on your solution. This testing should be systematic and comprehensive, and based on a test plan.

Final test plan (6 marks)

Create a plan of how you will carry out final testing of your fully implemented solution.

Your plan should be comprehensive, to ensure that your solution meets all the requirements identified at the analysis stage. It should include:

- all **requirements** you are testing
- a **description** of the tests you will carry out
- a **persona** and **test cases** to test the solution with an end user

Note: the tester can be another candidate, a teacher or a lecturer, who adopts the persona and carries out the test cases.

During Testing the Functional And End-User requirements that were stated earlier in the project document will be tested.

End-User Requirements Testing

1. "For players to be able to sign into their account and check the details of their account such as their account balance and games played." - This requirement will be tested by having an End-User play some games, while keeping track of the amount of games played, and the balance after the games. When logging in they will enter incorrect details to test the input validation, if their sign in is successful then the values shown in the program will be compared to the values taken by the end user and will show whether this function is carried out without error.

2. "For the program to give each player the option of whether to; Hit, Double Down or Stay during their turn of each round." - This will be tested by having multiple end users playing a game using the program, and checking if the action

which corresponds to the button clicked is being carried out each time.

3. "Each player should be able to bet up to £500 on each round and either lose what they bet, or win it back doubled." - This will be tested by a single end user playing multiple games, they will bet a different amount each time and calculate what their balance should be after winning or losing, and then comparing it to what they have (after either winning or losing a round).

4. "Each player should have the option to check the rules of BlackJack from the main menu, showing a list of the rules and a list of each card, its graphic and what it's value is." - to test this feature of the program all that must be done is to go onto the main menu of the program, click 'Rules', and check if the content on that form is correct and detailed enough for a person who has never played Blackjack to understand the outline of the game.

5. "Each user should be able to check a leaderboard which displays all the users with a rank depending on their balance the higher the balance the higher up on the leaderboard they are" - This can be tested by having the leaderboard form opened and then comparing the order to the order of an sql query that orders the balance of each account by descending order.

Functional Requirements Testing

1. "For the database to interact with the program after every round of the game, updating each player's information." - This can be checked by, after each round, opening and checking the contents of the database and how they compare with the values inside the program, more specifically the balance and games played of each player currently involved in the game.

2. "For the program to display a graphic of the cards that are dealt to the players and the dealer every turn." - This can be checked by simply going into a game, and placing a bet within the bounds, and seeing if cards have been dealt to your player and the dealer.

3. "Once a player has 'stayed' or has gone bust during that round the program should no longer show them a pop up window with each option that they could choose." - This can be checked by multiple accounts being signed into play a game, each of them either choosing to stay, to go bust on purpose, or carry on playing the round and checking whether after these actions, the pop-up window will show up or not , depending on which action they have taken.

4. "Once every player has 'stayed' or gone bust during that round, the program will cause the dealer's card that was face down to flip over and reveal itself, if the dealer's total is <=16 then the program will keep giving them cards until their total is >16" - this can be tested by multiple players playing a game, and keeping track of who is in and out of the round, when the program decides that they are all out of the round, it can be compared to whether they have noted down that they all

should've been out or not. And if that is correct then it can be observed whether the dealer's card that was face down has been flipped. It is also easy to keep track of the dealers total with the labels, and therefore can check if the dealer should or shouldn't have been dealt a card in a certain situation, using a breakpoint at the start of a conditional loop to stop the program and check the value of the cards

5. "The program should read in values from a database and be able to validate whether a players username and password are correct" - this can be tested by having a user create a new account, take note of the details, and then attempt to sign in with the 'account details' button, entering intentionally incorrect values to see whether the validation is correct or not, then when the correct data is entered it will be tested whether the user validation is working without error.

6. "The program should allow an unregistered user to create an account, saving the details of the account into a database and automatically setting their balance to £5000" - This can be tested by an end-user creating an account, trying to enter a username that has already been entered to check if the input validation works, and then creating an account with a unique username and checking if the correct values have been saved into the database as a new record.

General Test Plan

Each form will be tested while linked to each other (integrative testing), testing each form separately has been done as ongoing testing while the software is being implemented. It will be most effective to have users who have not used the program at all to try and use it to test any errors that could occur, this helps as people who have used the program will know where there are likely to be errors and avoid them to keep the program running smoothly. This is avoided when users who have not tried the software yet are introduced as they have to learn how to use it as well as navigate it. During this process mistakes are going to be made and this is when bugs will appear, helping identify any problems relatively easily. This will be carried out by having an End User use the program in accordance to the test cases that have been decided on. These will test every part of the program and will determine whether or not it is complete without errors.

Personas

1
- Teen
- Familiar with Blackjack
- Familiar With Software/Computers
- Enjoys Gaming

2
- Teen
- Unfamiliar with Blackjack
- Not adequate with computers
- Enjoys gaming

Test Cases

Every test case included in this list will be tested by an End User with the persona mentioned above

Case 1

- *An end user will create an account and play a few games with other players, after that they will check the leaderboard in order to find out where they rank out of all the other players* - This will test the link to the database, whether the data on the user is updated throughout the time they are playing a game, and if the sort algorithm that works out the order of the leaderboard is working correctly.

Case 2

- *An end user will create an account, play a few games, throughout the game all other players in the game will leave and they will stay, while keeping track of their balance and how many games they have played, then check their account details - entering the wrong username and password in order to test input validation - and compare the expected values of Balance and games played to the ones that are displayed*. - This will test the link to the database, the input validation, if the game still runs after some players have left and whether the data of the user is updated throughout the game.

Case 3

- *A new user will check the rules page, afterwards going to play a game and creating a new account, using a username that has already been used, then changing it after an error message has popped up, then playing a few games to check if the rules given are adequate enough for a new player to understand the the game to an extent after reading them*. - This will test how useful the rules page is and will further test the game form as well as testing the input validation of the 'Create Account' form.

Requirements testing (6 marks)

Test your solution and provide evidence of each end-user and functional requirement test identified in your plan.

The problem you are solving will determine the evidence you require. The following may be suitable:

- screenshots of program inputs and outputs (including errors if any are generated)
- printouts of database output tables generated by SQL statements
- screenshots of form data being entered, along with subsequent results (for example: table updated and errors returned)

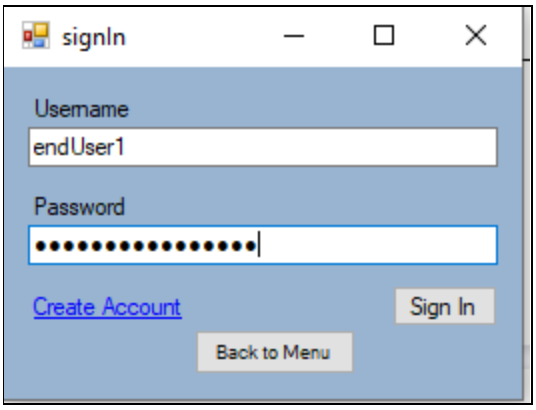
End-User Requirements Testing

1. "For players to be able to sign into their account and check the details of their account such as their account balance and games played."

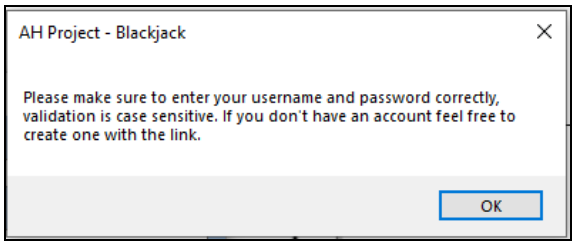
Games Being Played: 5
Expected Balance: £4379

Option Chosen From mainMenu - Account Balance

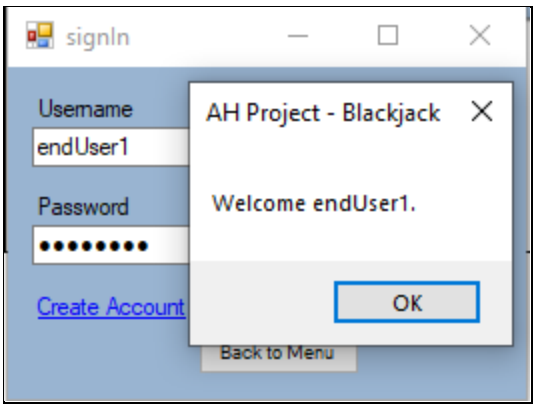
Sign In Form - Incorrect password Input



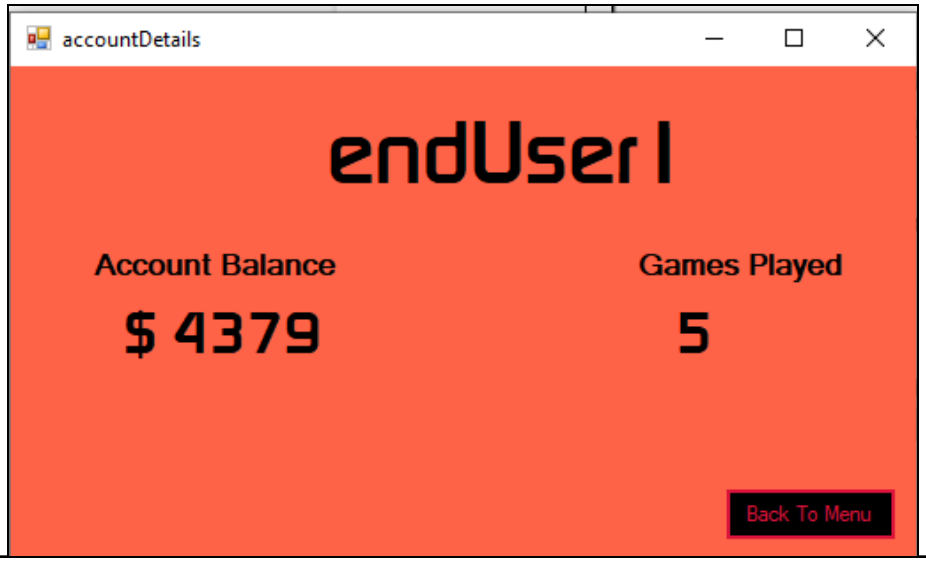
Error message shown due to incorrect input



Message box shown when correct values are input



Account Details Form Displays with Correct values for Games Played and Account Balance

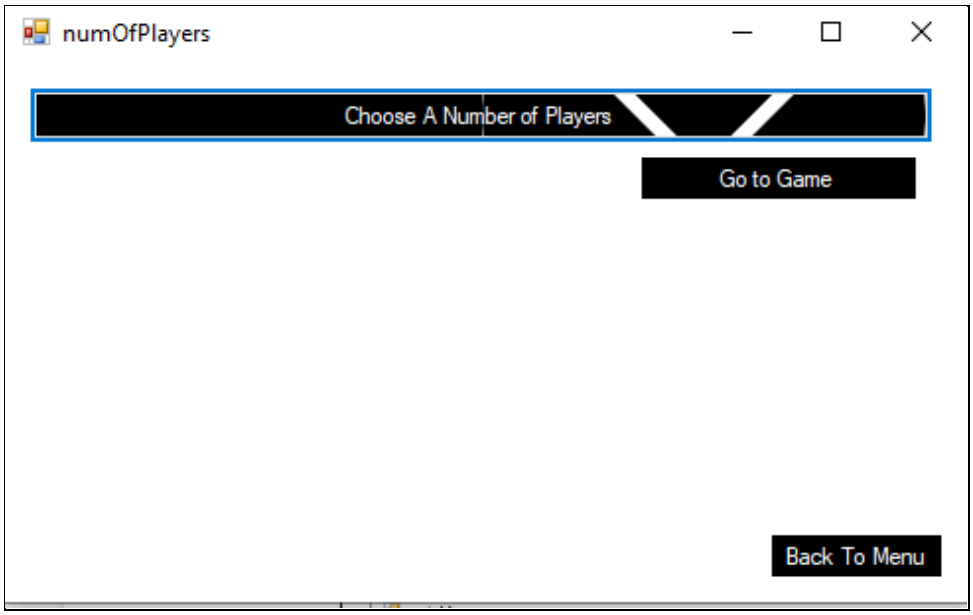


With input validation working and the expected outcome being the one that was carried out, this requirement has been met.

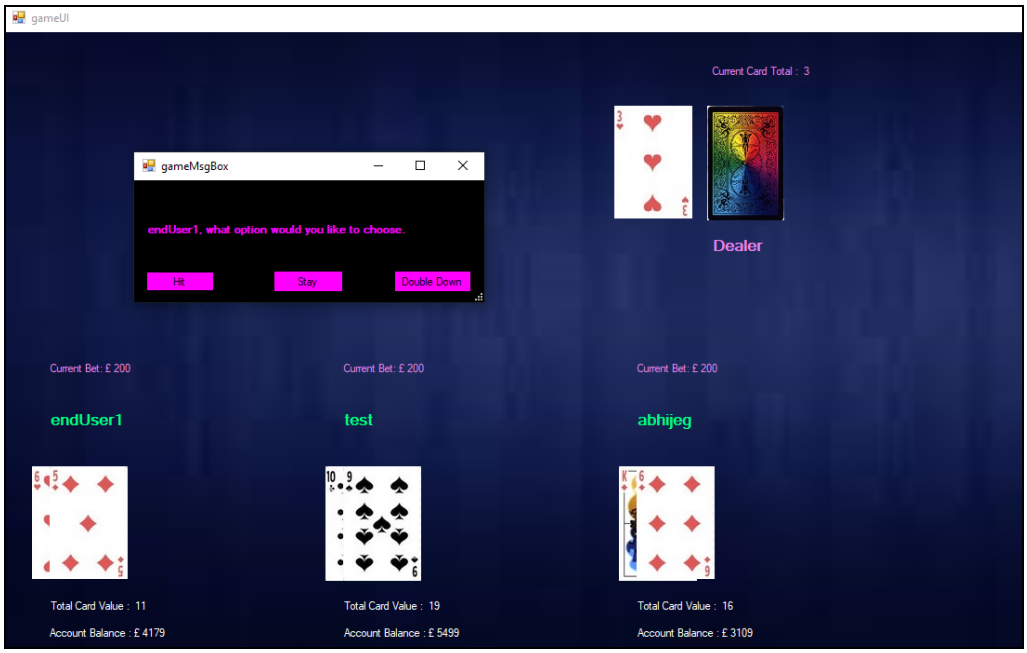
2. “For the program to give each player the option of whether to; Hit, Double Down or Stay during their turn of each round.”

mainMenu option chosen - Start Game

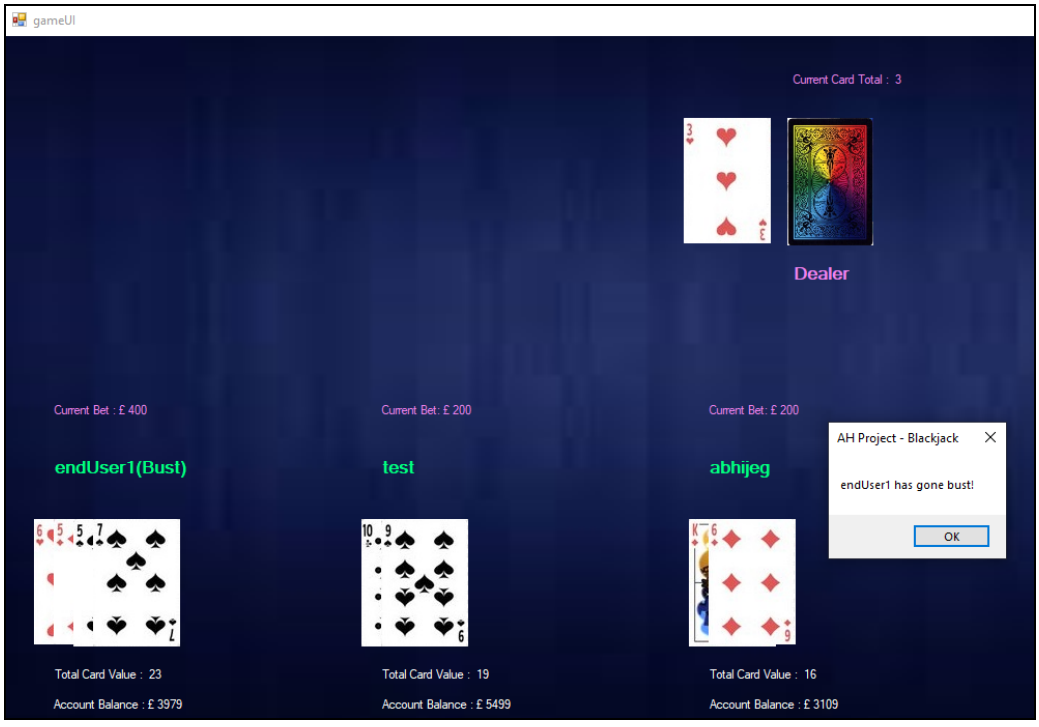
Number of Players Chosen - 3



All bets are placed, and endUser1 is given his options, they choose to Double Down (Which means getting dealt 2 cards in return for doubling your bet)

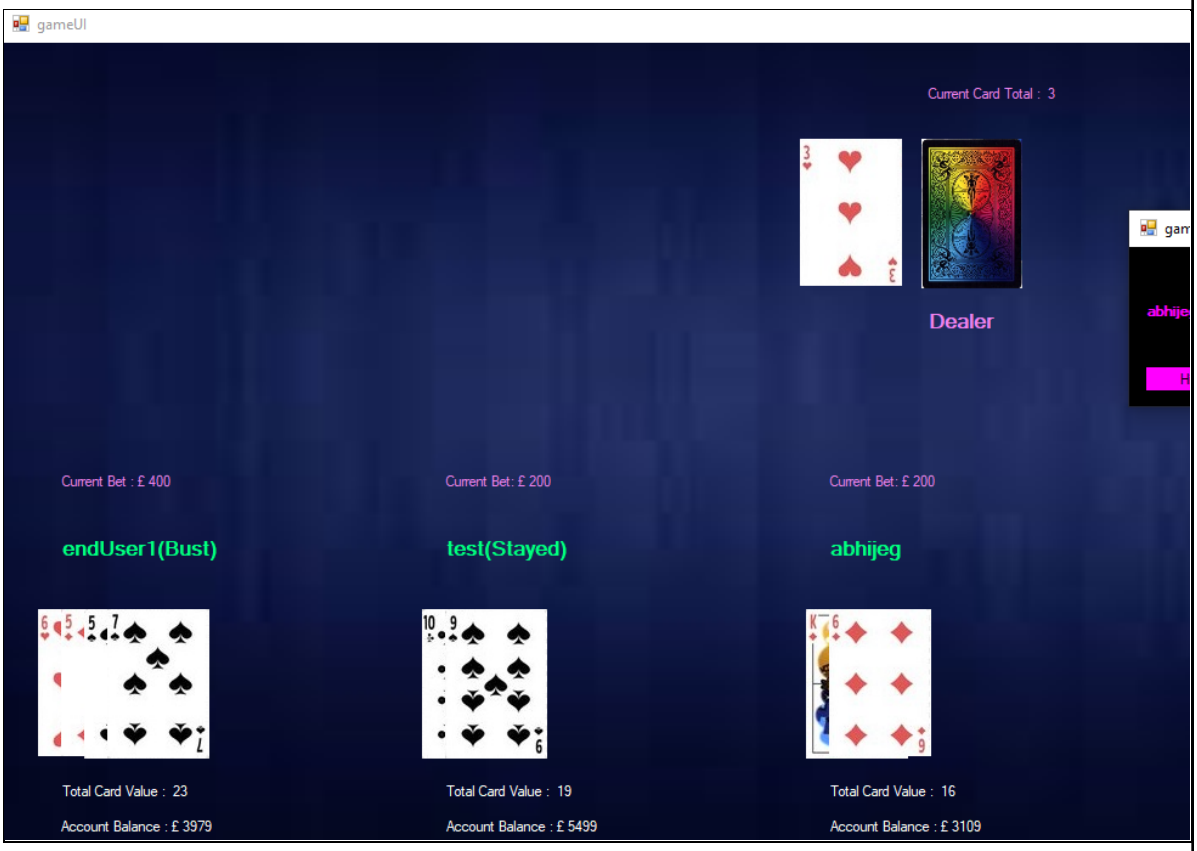


As is shown, their bet is doubled from before and they have been dealt 2 cards instead of one, even though in this example they have gone bust.



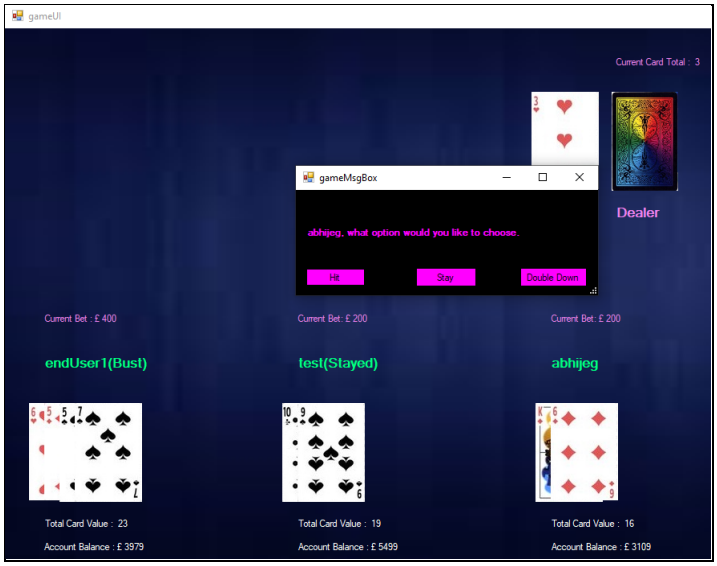
(£400 instead of £200)

test gets options and chooses to stay (Not ask for any more cards and keep the cards he has until they are compared to the dealers cards at the end of the round)

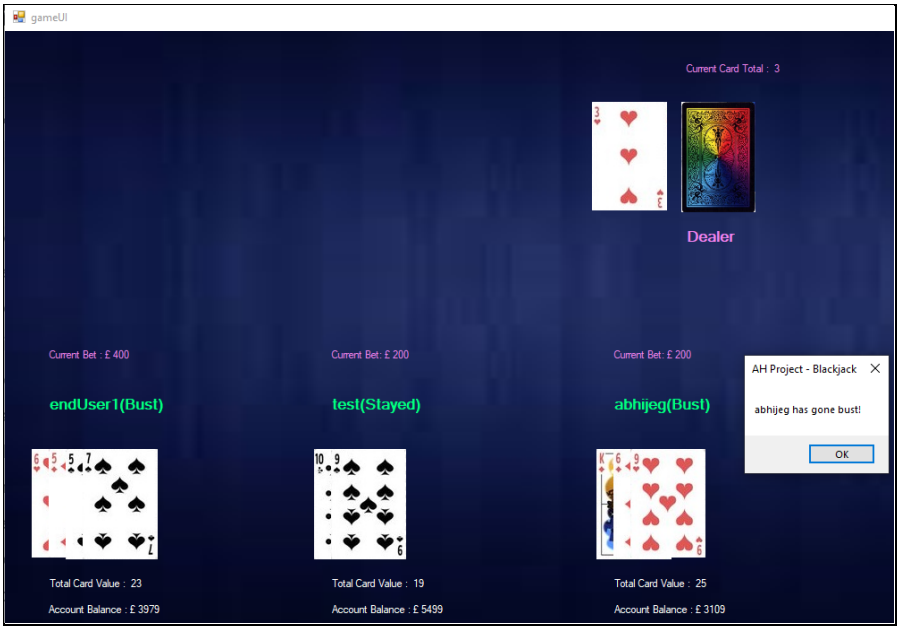


As an extra feature the status of a player is given beside their name if they are out of the round for any reason.

abhijeg chooses to hit (Asks for a single card to be dealt to him, no changes to bet or anything else apart from card value)



Another card was dealt, unfortunately it caused the player to go bust, but a single card was dealt without affecting any of the other running totals in the program.

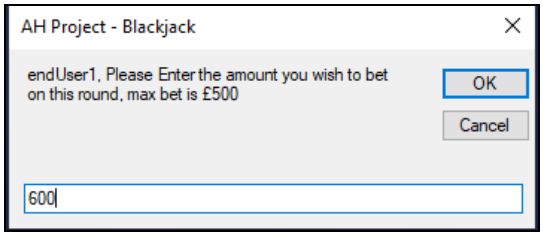


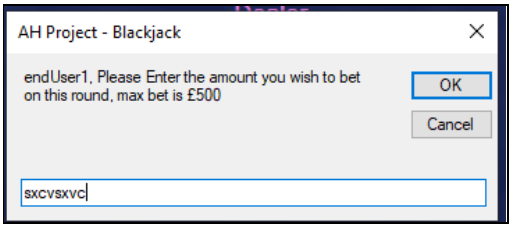
All three of these features working correctly means this requirement has been met.

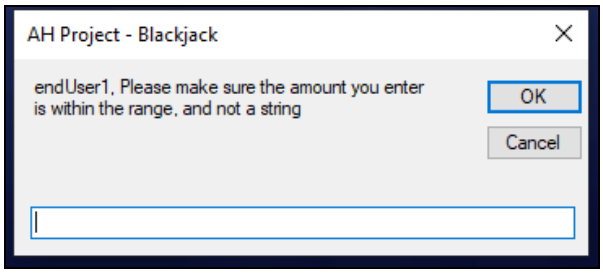
3. “Each player should be able to bet up to £500 on each round and either lose what they bet, or win it back doubled.”

mainMenu option ‘Start Game’ is chosen, it is decided that there will be one player in the game

When an input that is out of the range of the bet, or is a different datatype an error message is shown

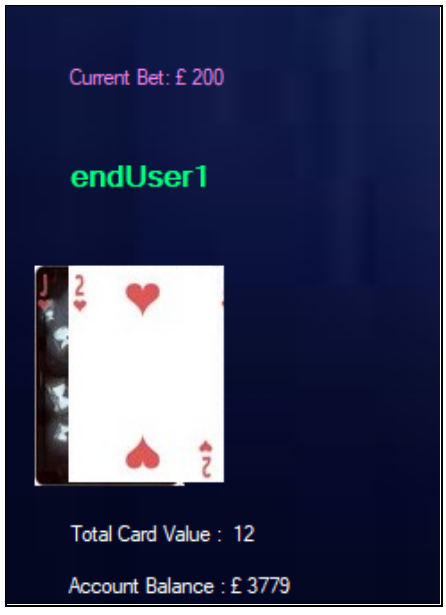




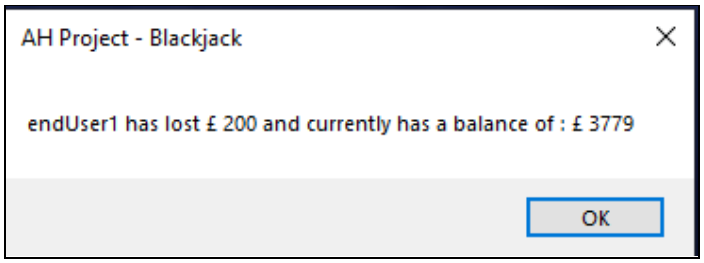


When a valid bet is placed, the value of it is displayed and the balance of the user is updated along with it

<u>Round 1</u>

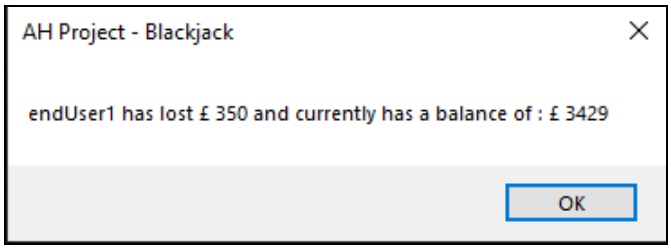


Win - Balance = £4179
Lose - Balance = £3779

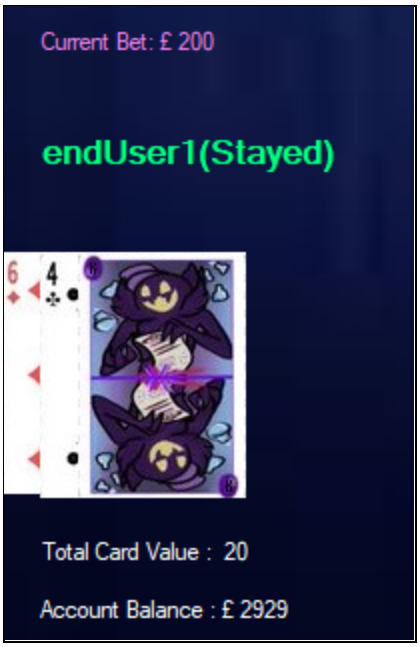


Round 2

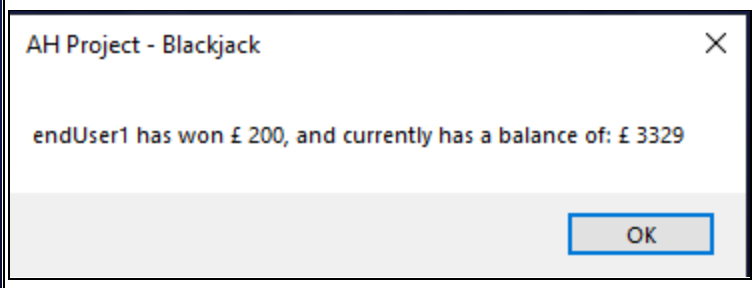
Win - Balance = £4129
Lose - Balance = £3429



Round 3



Win - Balance = £3329
Lose - Balance = £2929

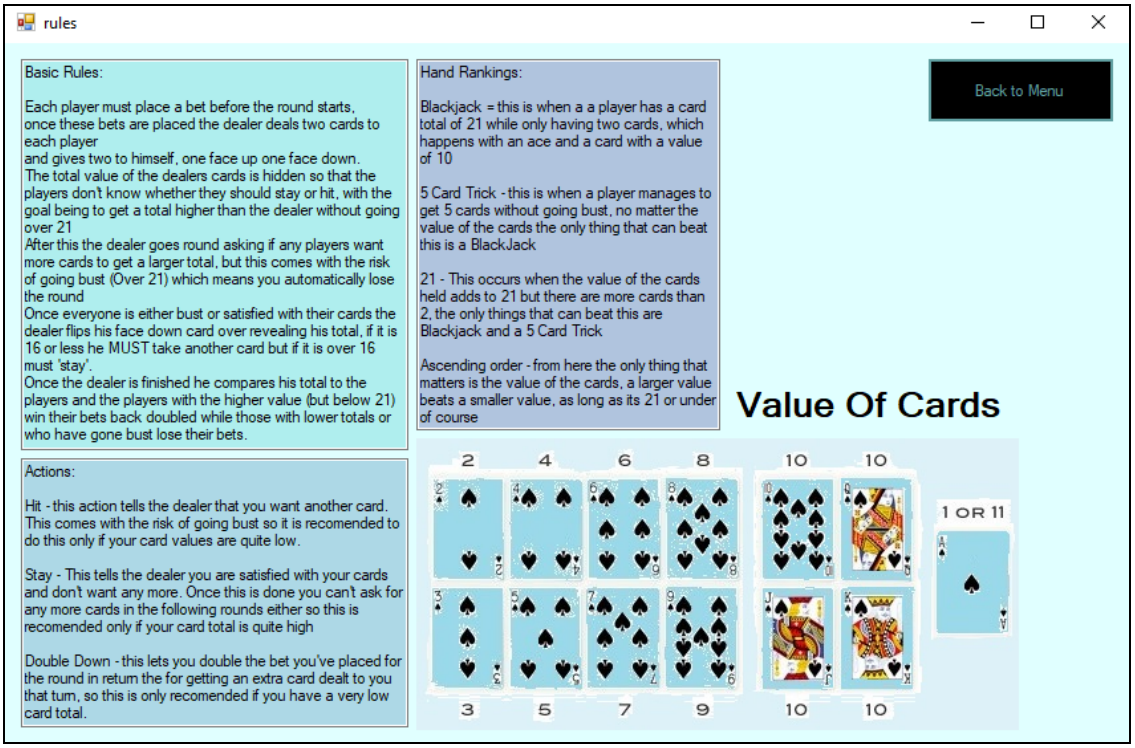


With the the input validation working well, all possible outcomes of the program running fine, and the expected balances for every round being correct this requirement has been fulfilled.

4. “Each player should have the option to check the rules of BlackJack from the main menu, showing a list of the rules and a list of each card, its graphic and what its value is.”

mainMenu option that is chosen is the ‘Rules’ page

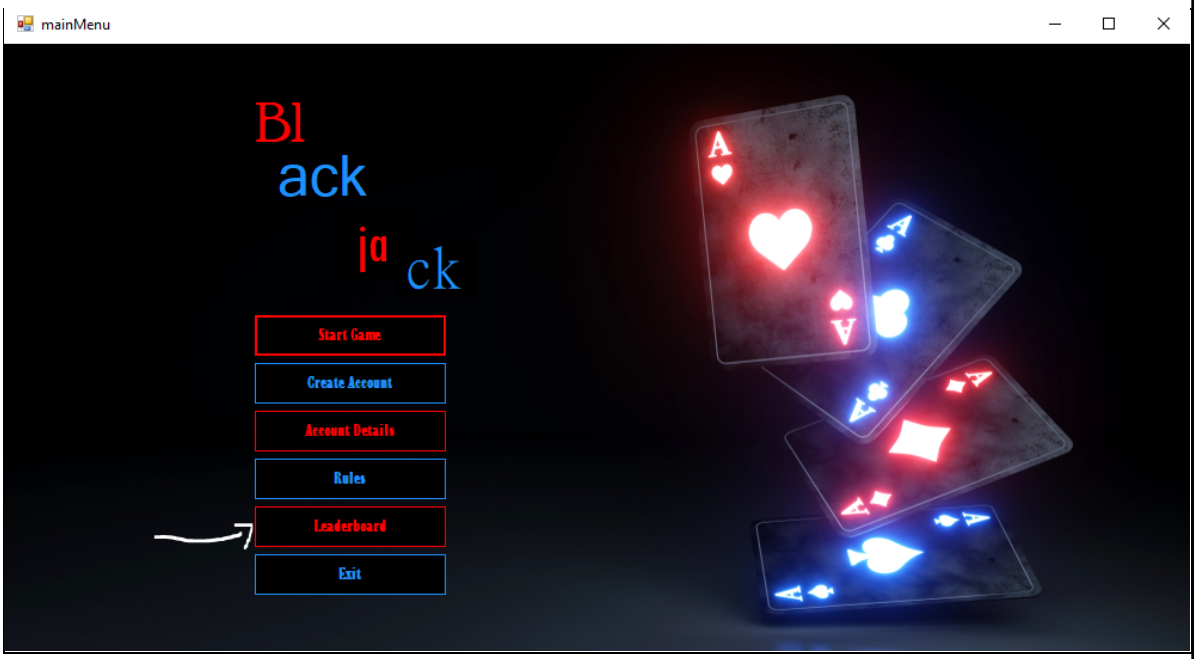
The form that shows up when this is clicked is the following.



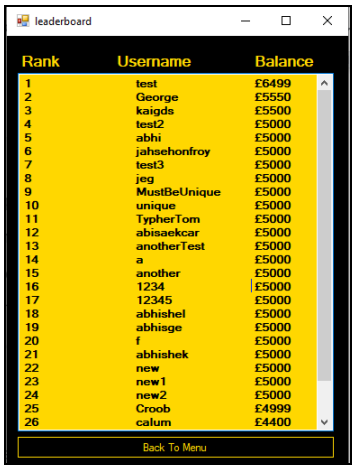
It explains the way each round is carried out, any actions the player can take during the round, and how to tell whether you have won or lost, while displaying a picture of all the cards in a deck as well as their corresponding values, meaning that this requirement has been completed.

5. “Each user should be able to check a leaderboard which displays all the users with a rank depending on their balance the higher the balance the higher up on the leaderboard they are”

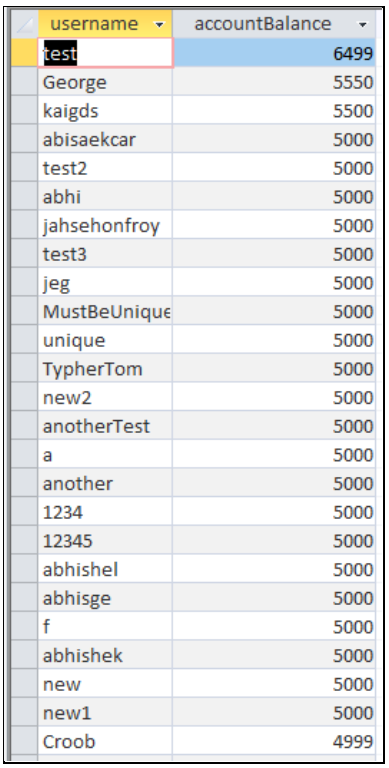
To test this the option chosen on the main menu is ‘Leaderboard’



When this option is clicked the leaderboard form is opened and a sort algorithm is carried out to order the values.



```
SELECT username, accountBalance
FROM Players
ORDER BY accountBalance DESC;
```



username	accountBalance
test	6499
George	5550
kaigds	5500
abisaekcar	5000
test2	5000
abhi	5000
jahsehonfroy	5000
test3	5000
jeg	5000
MustBeUnique	5000
unique	5000
TypherTom	5000
new2	5000
anotherTest	5000
a	5000
another	5000
1234	5000
12345	5000
abhishel	5000
abhisge	5000
f	5000
abhishek	5000
new	5000
new1	5000
Croob	4999

With the sql query showing the same order as the leaderboard form and the values for balance matching the usernames the same as in the leaderboard form, this requirement has been met.

Functional Requirements

1. “For the database to interact with the program after every round of the game, updating each player’s information.”

mainMenu option that is chosen is ‘Start Game’

The amount of players chosen will be 2

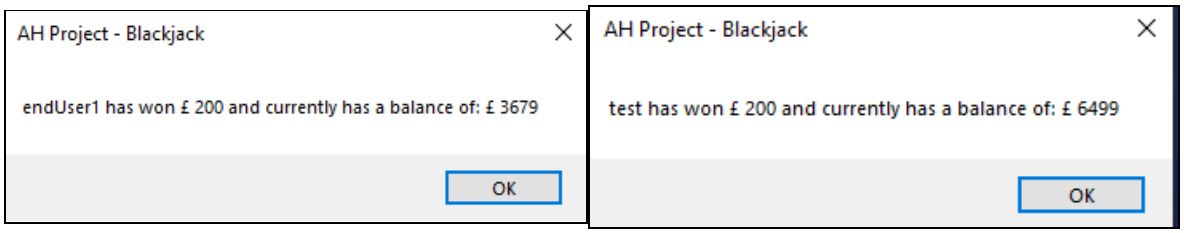
The balance of each account in the program matches the balance of each account in the database

endUser1	password	3479	15

test	random	6299	9

The number in the last field is the number of games played, so if the database is updated successfully then not only should the balance be updated but the value of games played should increase by one.

After one round the balances given are:



The updated rows in the database show:

	endUser1	password	3679	16
	test	random	6499	10

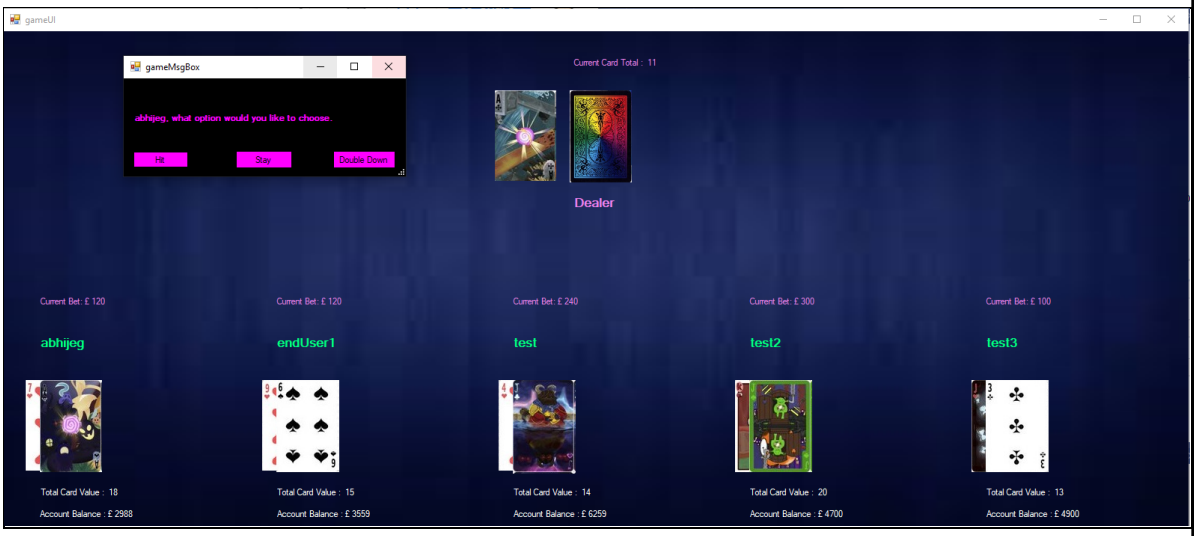
With both balances being updated accordingly and the number of games played increasing by 1 for each row, this requirement has been fulfilled.

2. "For the program to display a graphic of the cards that are dealt to the players and the dealer every turn."

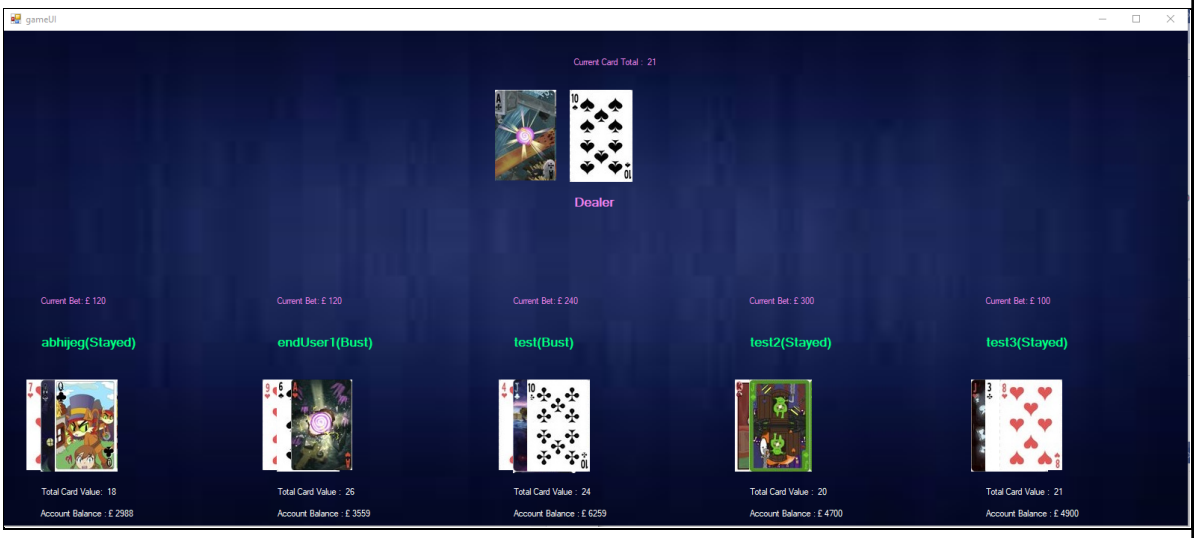
The mainMenu option chosen for this is 'Start Game'

The amount of players chosen is 5

Once all bets are placed the cards are shown to each player, with the dealer having one card shown and one hidden



Once every player has either stayed or gone bust, the card that is hidden flips around revealing the dealer's card



Since each round, card graphics are displayed after bets are placed, throughout the round depending on the options that players choose, and the dealers cards are revealed at the end of the round, this requirement is fulfilled.

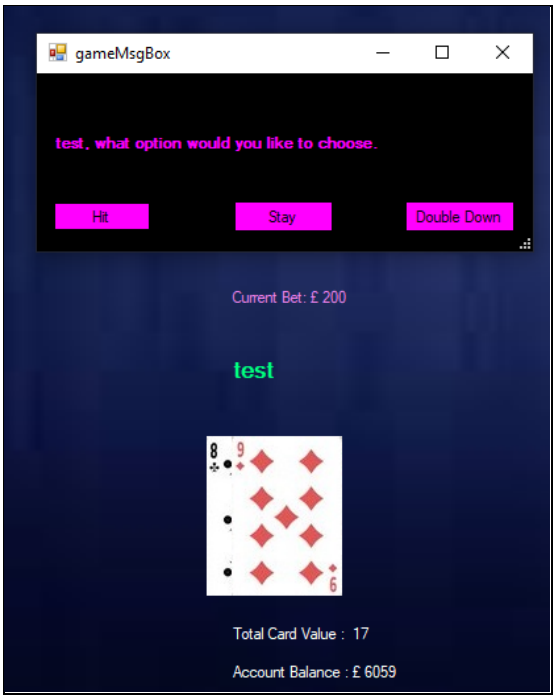
3. “Once a player has ‘stayed’ or has gone bust during that round the program should no longer show them a pop up window with each option that they could choose.”

To test this the ‘Start Game’ option has been chosen

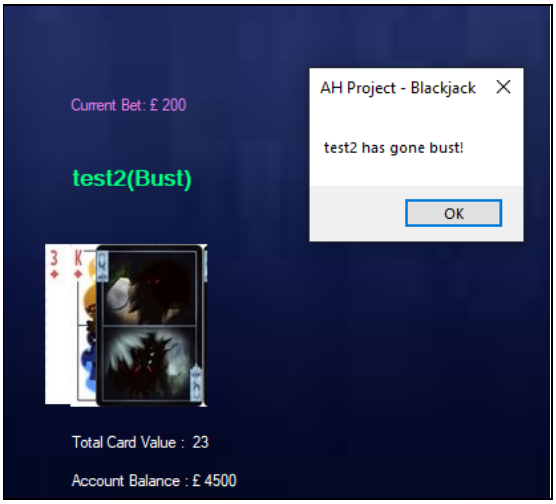
The amount of players will be 3

This player will choose to ‘stay’

This player will chose to ‘hit’ and is still in the game



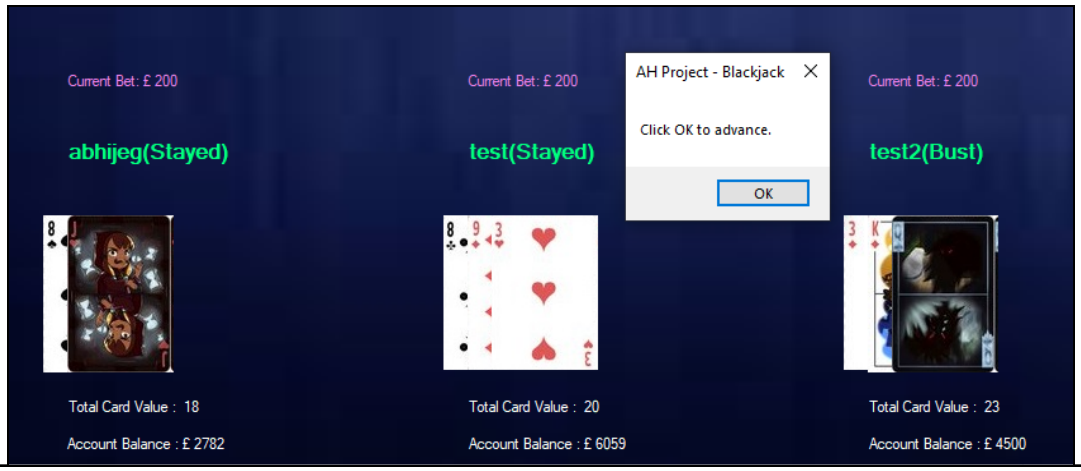
This player chose to ‘hit’ and went bust



With ‘abhijeg’ stayed, and ‘test2’ bust, when the next pop up window shows it should address ‘test’

They choose to stay.

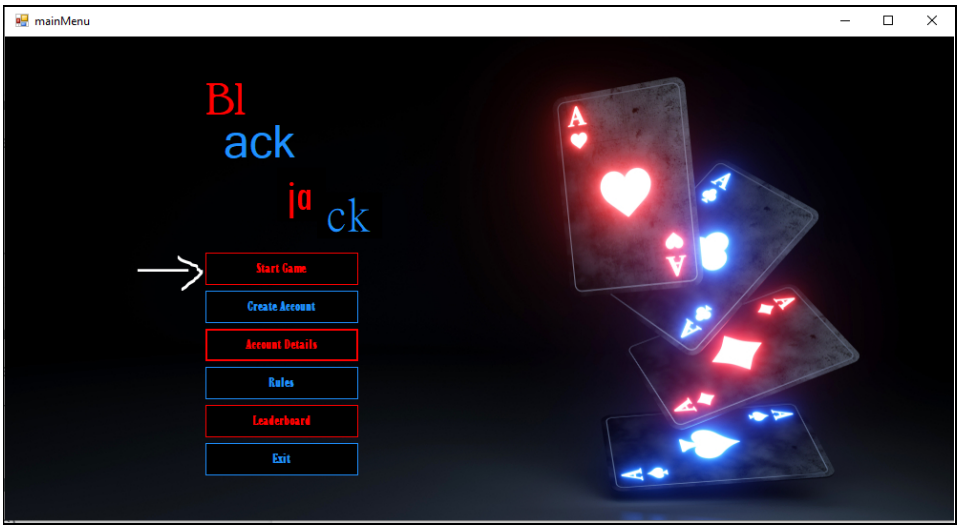
With every player out of the round by either staying, or going bust, no more option windows show up and the round moves on to the phase where the players cards are compared to the dealer.



With the option windows no longer showing up for the players who are out of the round, and then stop showing up altogether when the round is complete this requirement has been met.

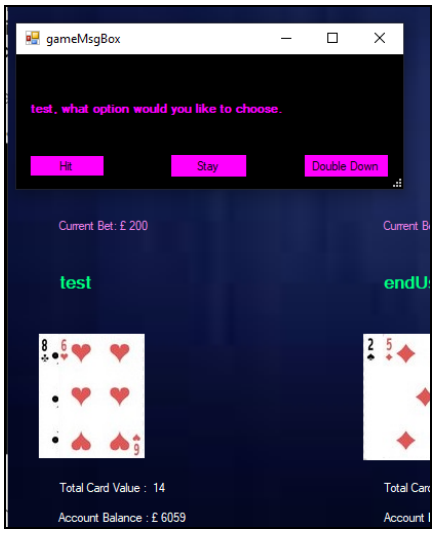
4. “Once every player has ‘stayed’ or gone bust during that round, the program will cause the dealer’s card that was face down to flip over and reveal itself, if the dealer’s total is <=16 then the program will keep giving them cards until their total is >16”

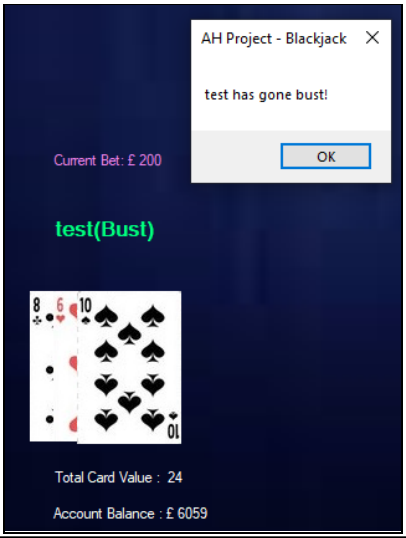
The mainMenu option chosen will be ‘Start Game’



The amount of players chosen will be 3

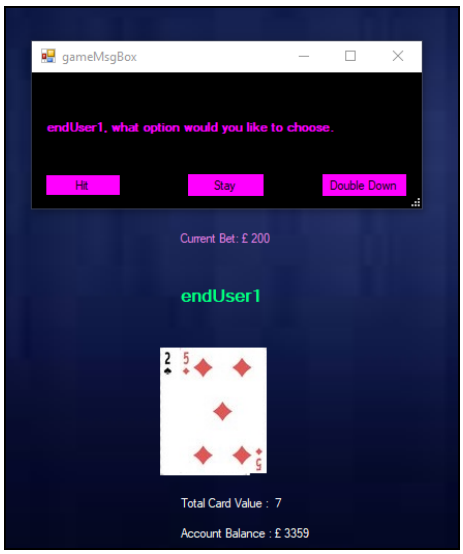
Test chooses to ‘hit’ and goes ‘bust’

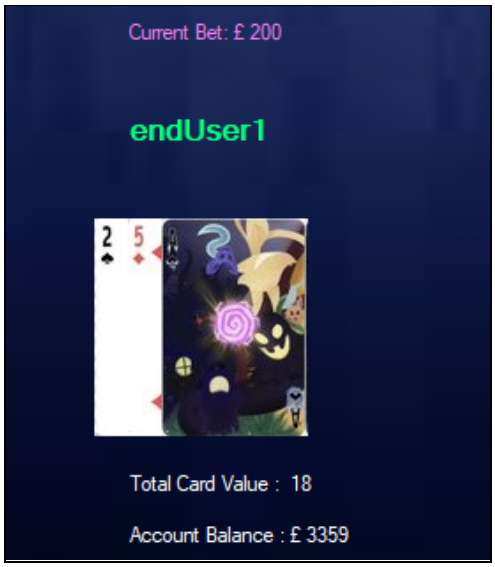




Test - bust
EndUser1 - in
Test2 - in

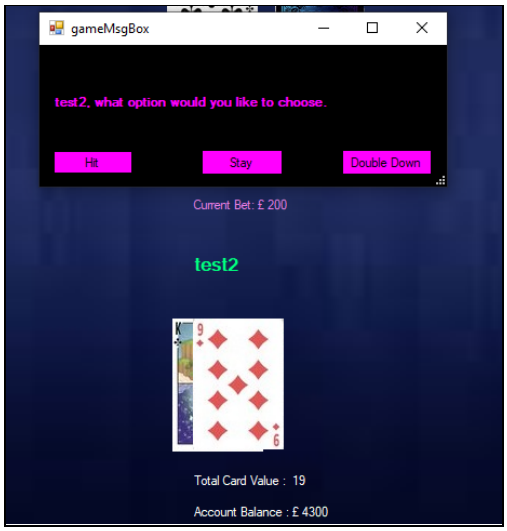
endUser1 chooses to ‘hit’ and stays in





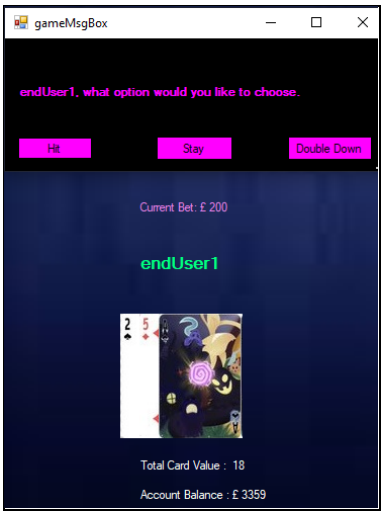
Test - bust
EndUser1 - in
Test2 - in

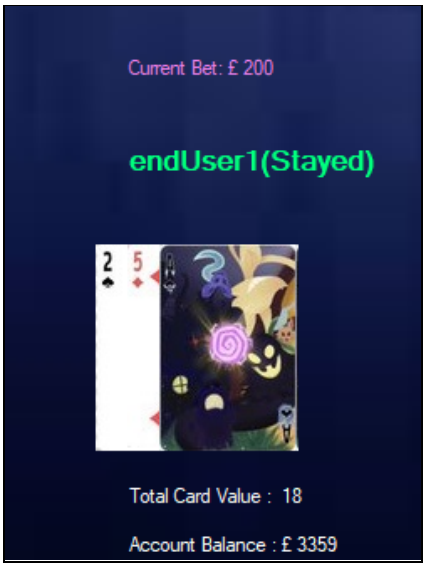
Test2 chooses to ‘stay’



Test - bust
EndUser1 - in
Test2 - stayed

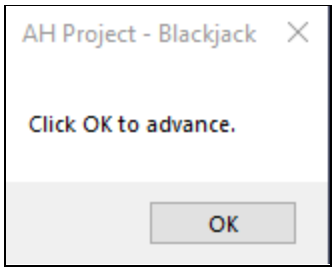
EndUser1 chooses to ‘stay’





Test - bust
EndUser1 - stayed
Test2 - stayed

With everyone out of the round (either bust or stayed) the game moves on to its next part with this pop up



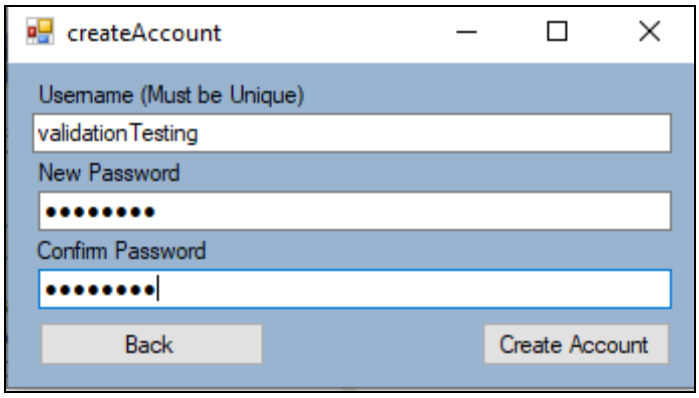
With everyone out the dealer is dealt his cards

As shown he initially had a 10 of clubs, and a face down card. With the card flipped over and it being a 4 (less than 16) another card is dealt, which takes his total to over 16, stopping the cards being dealt.

With every player having kept track of their status and the round ending when all the players are out, and when the round is supposed to end, this requirement has been met. With the dealer's card flipping over and him getting dealt cards until his total is over 16, this requirement has been met.

5. "The program should read in values from a database and be able to validate whether a players username and password are correct"

An account has been created



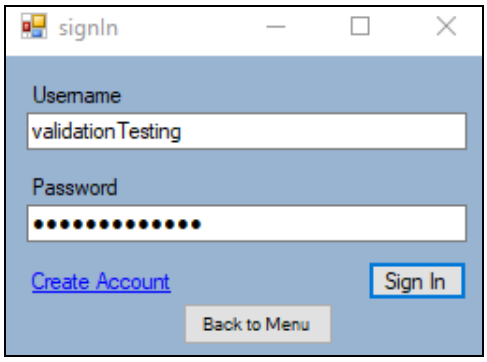
Username - validationTesting
Password - pass1234

This causes a new record to be created in the database holding the information on this account.

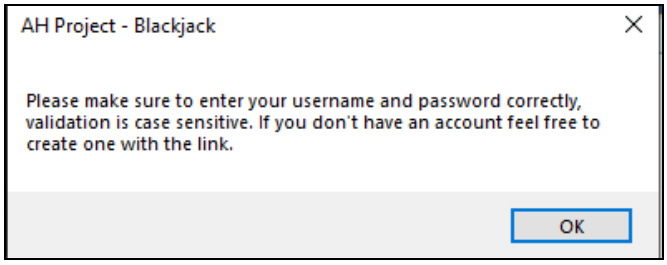
With the account made the mainMenu option chosen will be 'Account Details'

In this form all usernames and passwords are read in from the database and compared to the values input by the user, if they are valid then the form closes and the account details form opens, if not then an error message is shown.

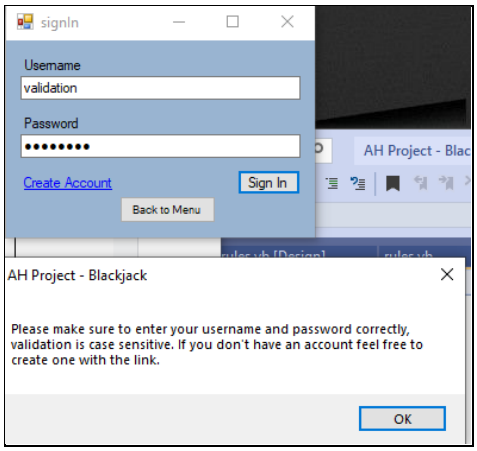
The correct username but incorrect password is entered



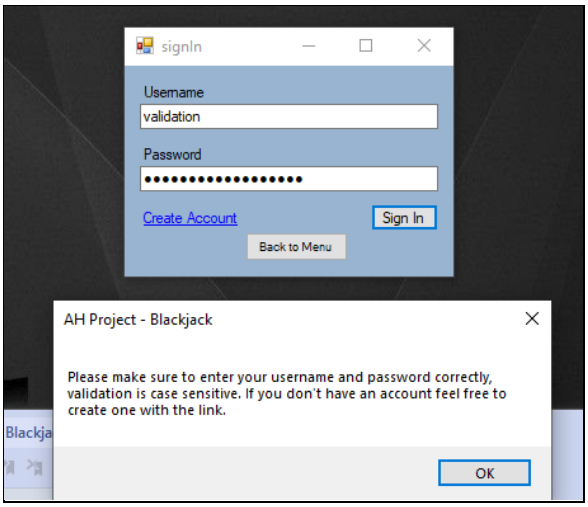
And this error message is shown



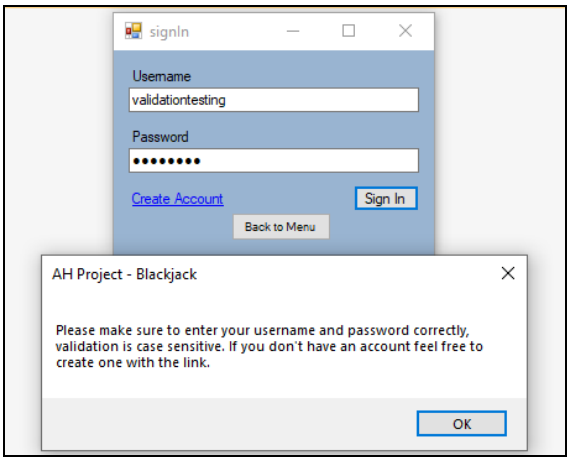
The incorrect username but correct password is entered and the same error message is shown



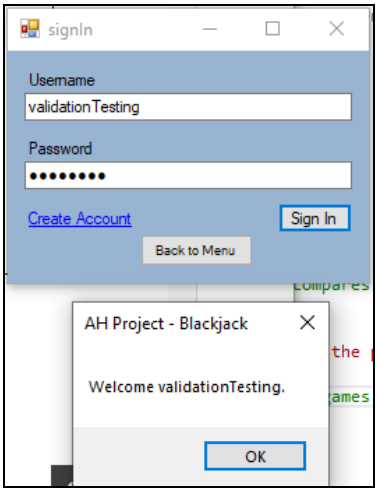
This time both values are entered incorrectly and again, the same message is shown



Now both values are entered in the wrong case and the error message is shown



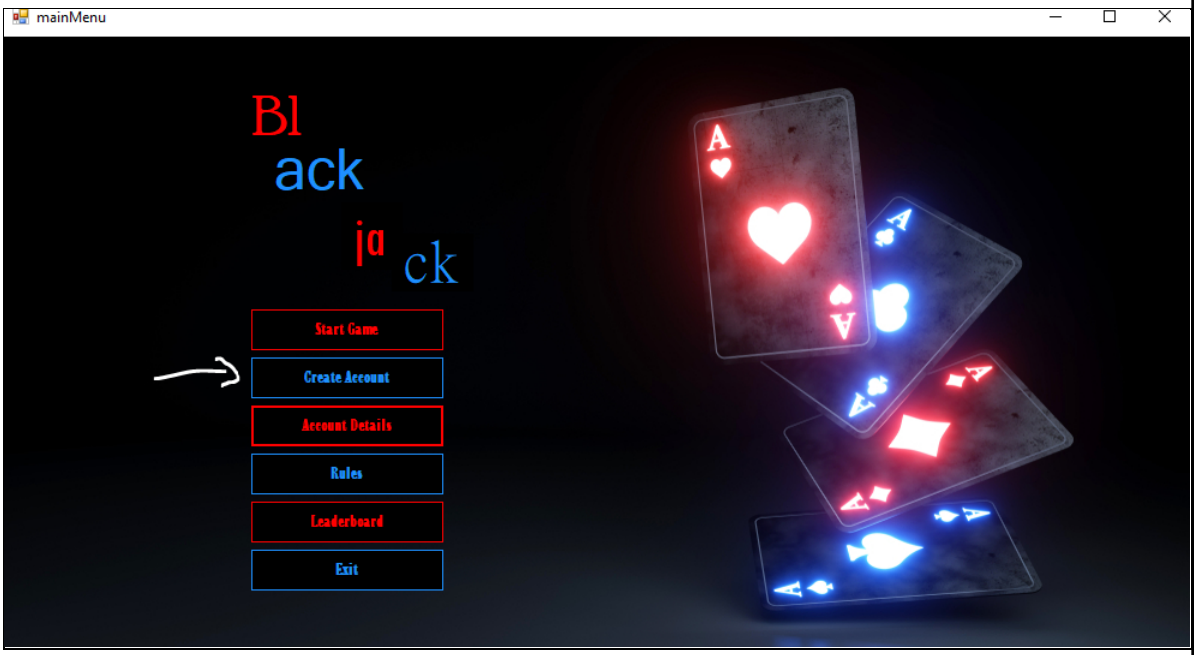
Now both values are entered correctly and the ‘Account Details’ form is opened



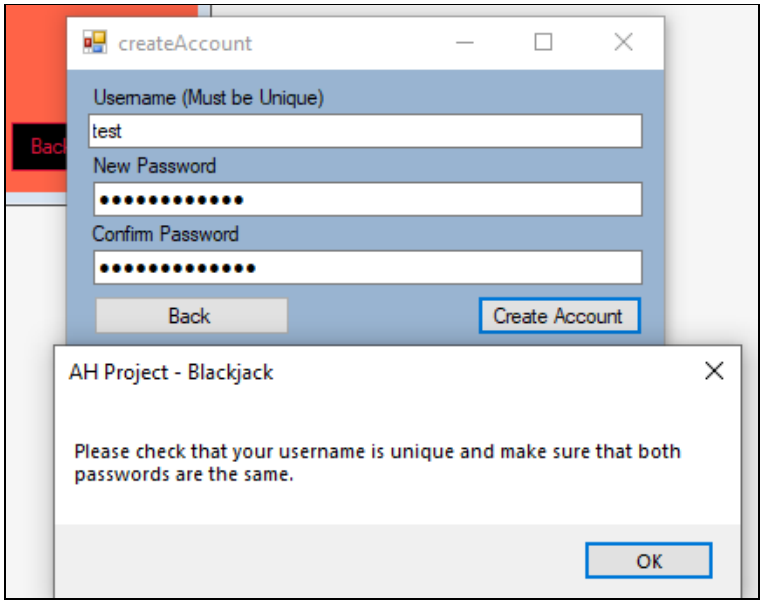
With the validation working to the extent where it is case sensitive, and the values being read in from the database any time any validation is required, this requirement has been met.

6. “The program should allow an unregistered user to create an account, saving the details of the account into a database and automatically setting their balance to £5000”

The mainMenu option chosen will be ‘Create Account’



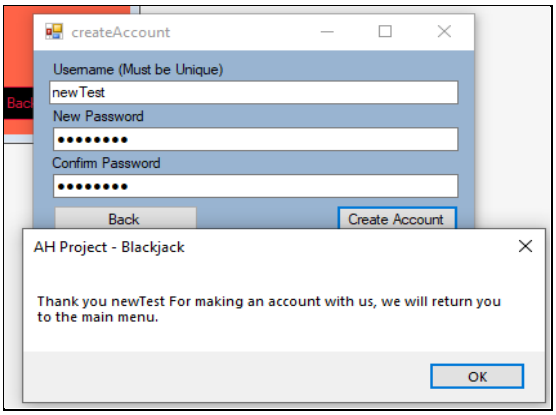
A username that has already been taken will be entered to test input validation



This error message is shown

When both passwords are entered as different values then this same error message appears

Username - newTest
Password - 1234word



In order to check if the correct value of £5000 has been entered into the accounts details then we can check the database.

Username	password	Balance	games Played
newTest	1234word	5000	0

With the correct balance entered, the new account being written into the database, and the unique username validation check working, this requirement has been met.

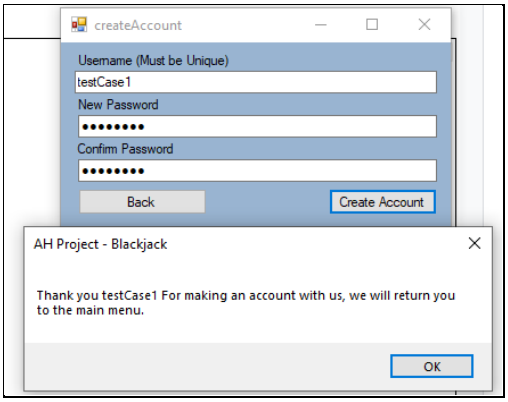
Testing with persona and test cases (3 marks)

Test your solution using the **persona** and **test cases** identified in your plan.

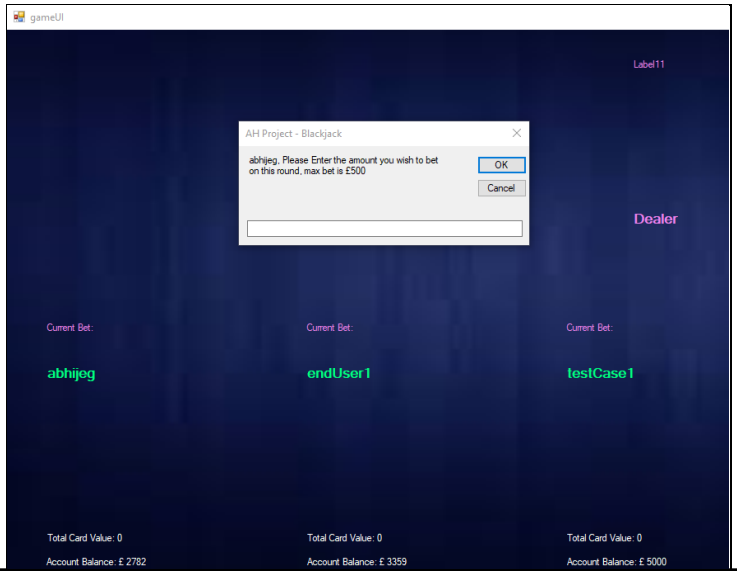
Describe the results of each test case — this could be in the form of a short report or a table.

Case 1
Persona 1

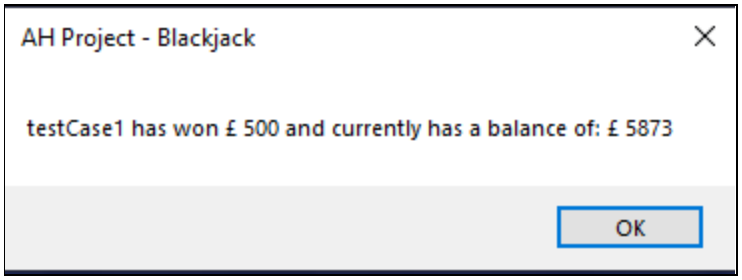
A new account is created called ‘testCase1’

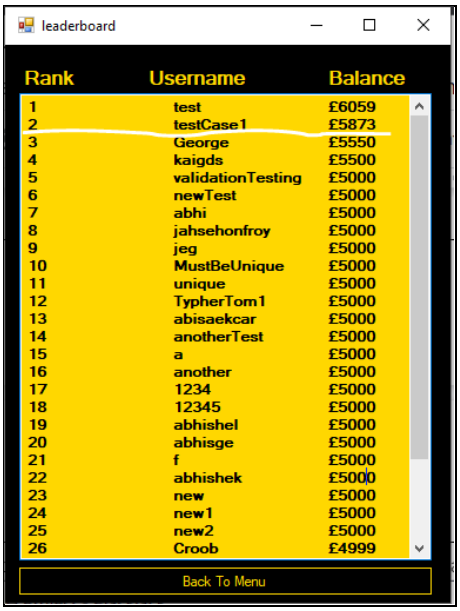


A new game is started with 3 players playing, 5 rounds are played



After the 5 rounds the expected balance of testCase1 is £5873

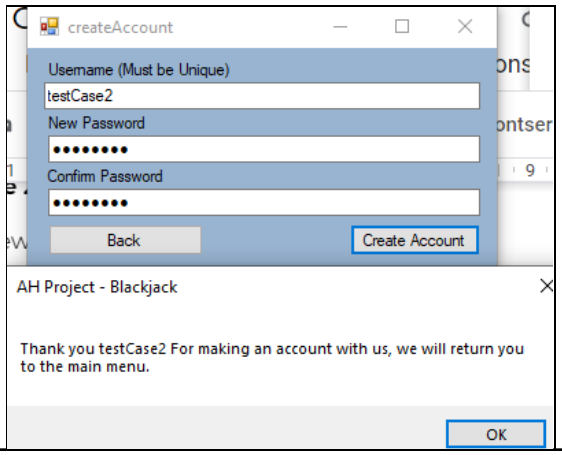




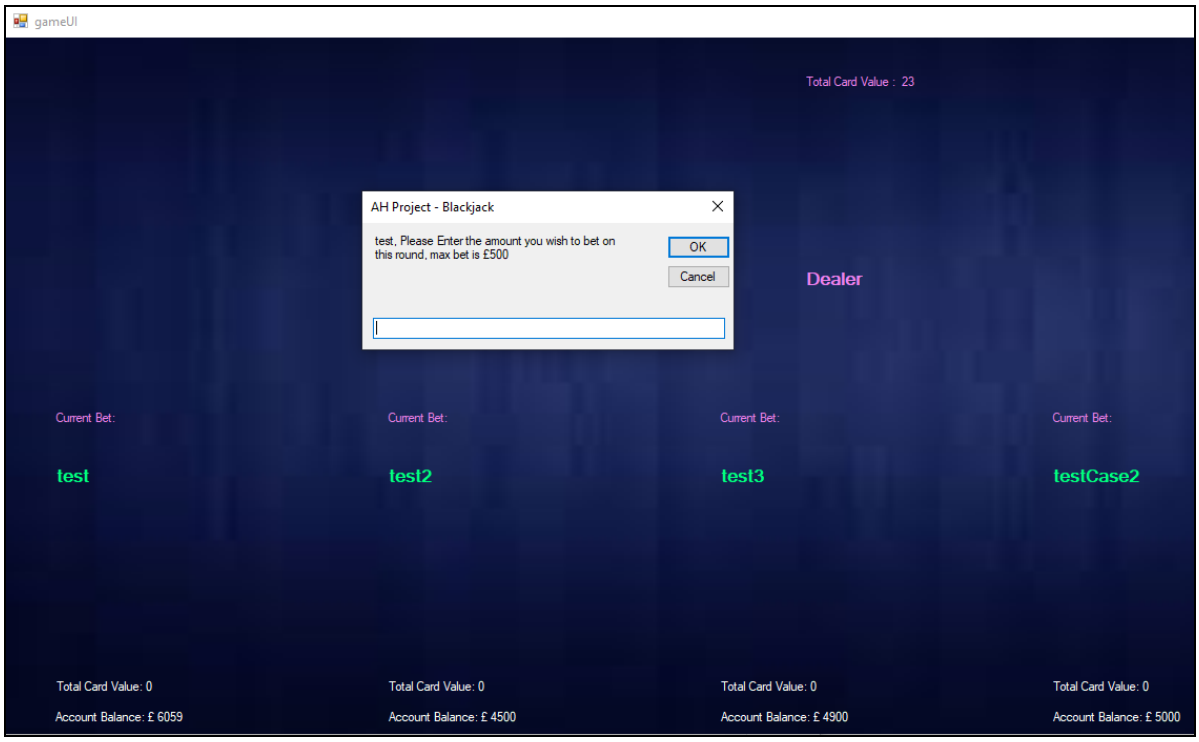
With the expected balance matching the actual balance, and the player's rank being correct in accordance to the other players' balance, this test case is a pass.

Case 2
Persona 1

A new account is created called 'testCase2'

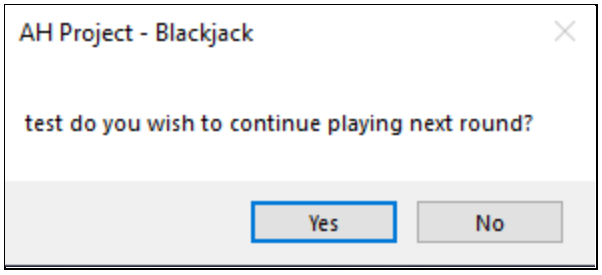


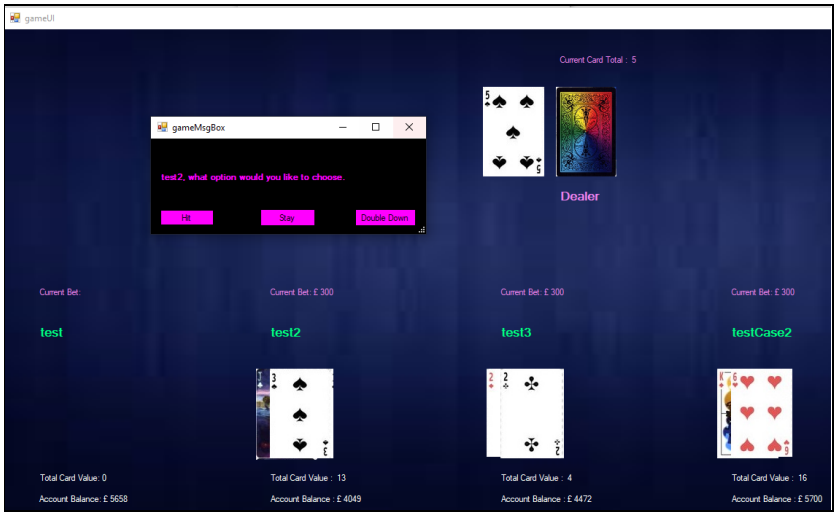
A new game is started with 4 players



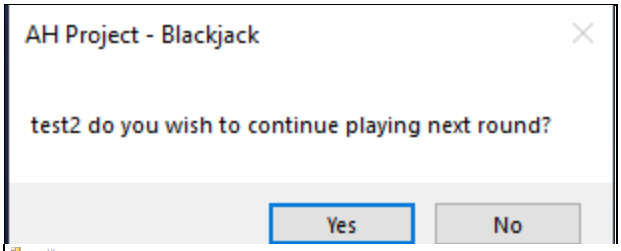
7 games are played and throughout that all of the players except from testCase2 leave

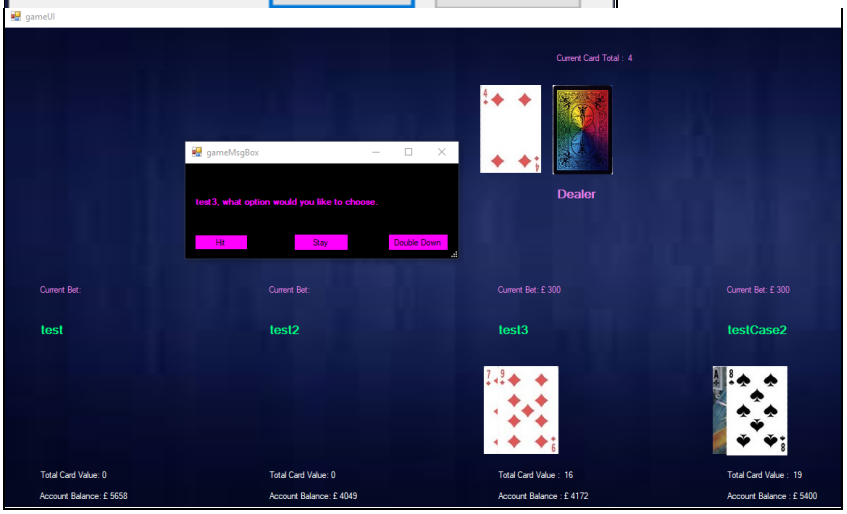
Round 3 test leaves



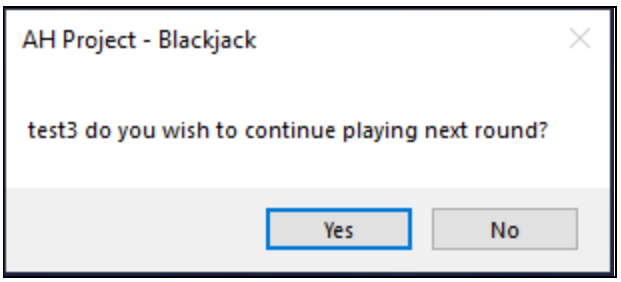


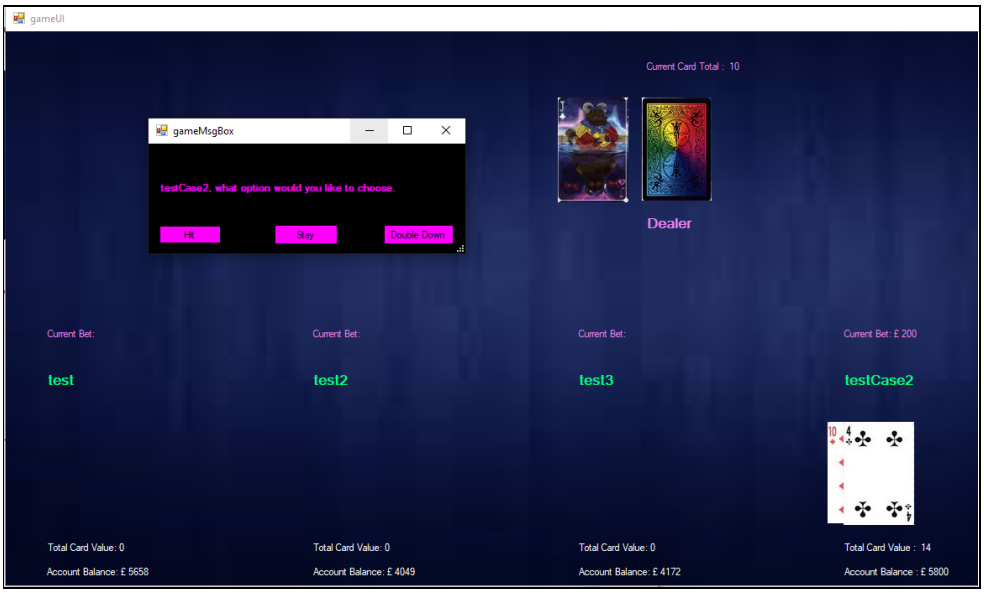
Round 5 test2 leaves



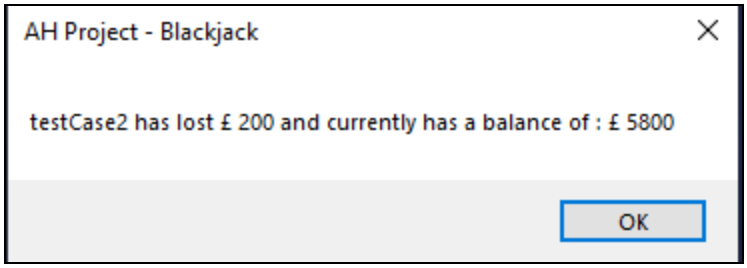


Round 6 test3 leaves

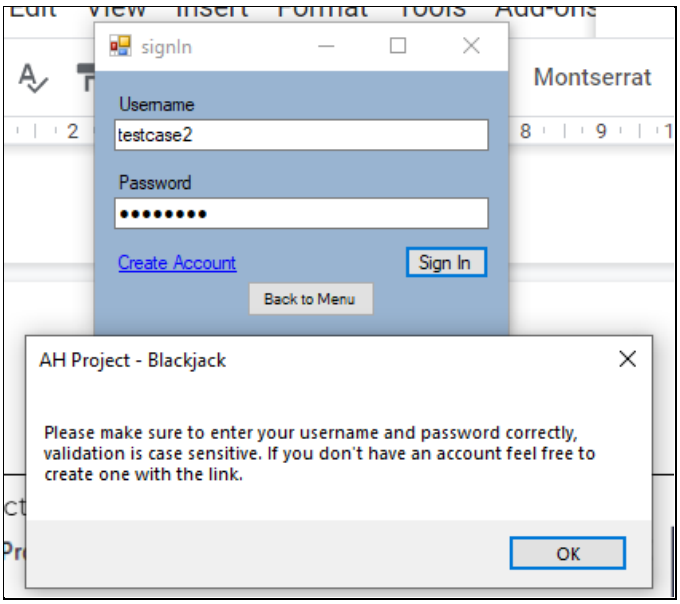




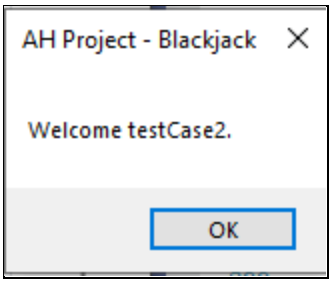
Expected Balance is 5800

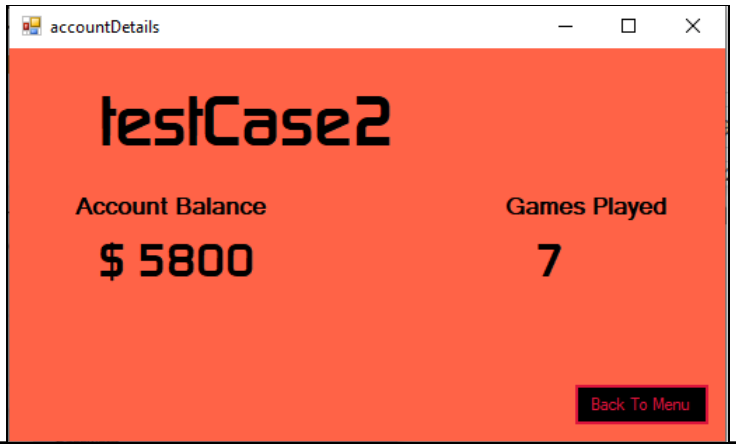


When signing into account details, the incorrect details are entered and an error message is shown.



When the correct details are entered a pop up is shown and the 'Account Details' form is shown

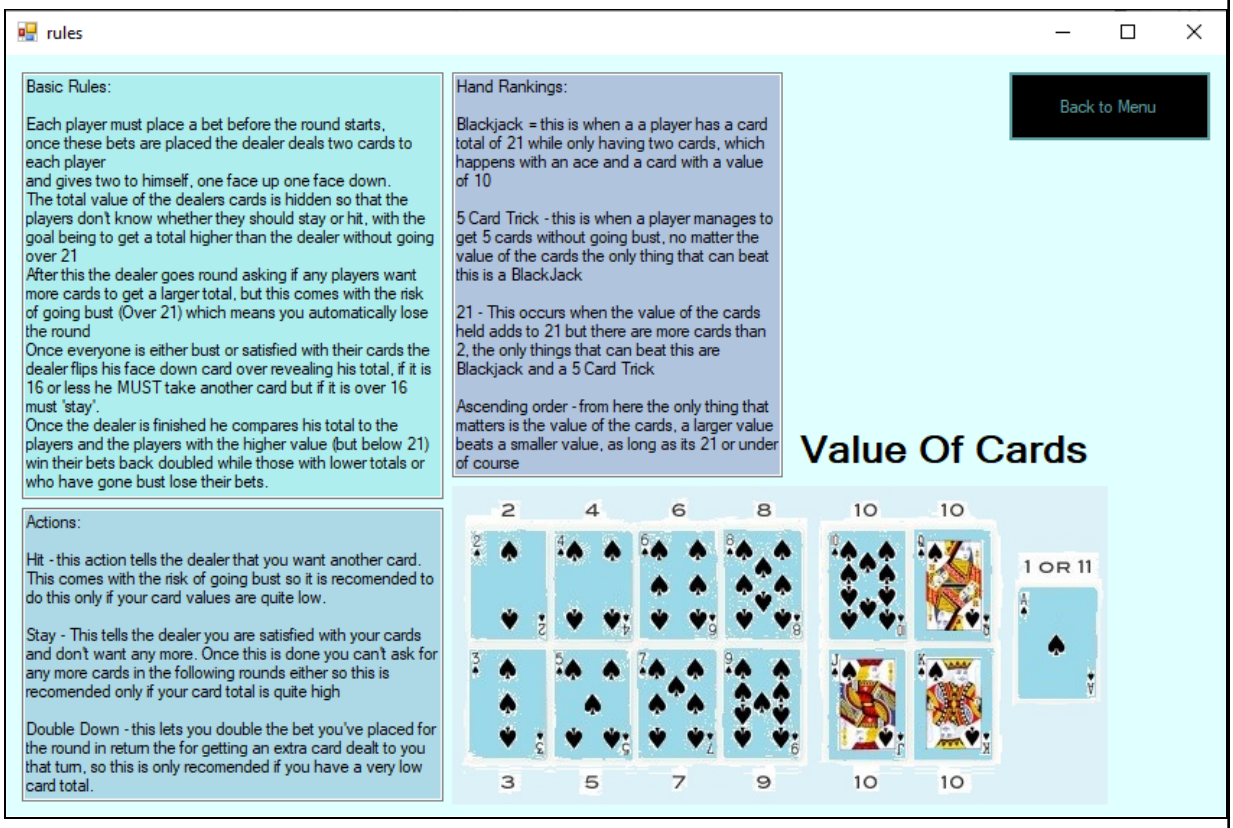




With the game continuing completely fine with some players leaving, the data from the database being updated every round, the input validation correctly validating incorrect and correct details, this tes case is a pass.

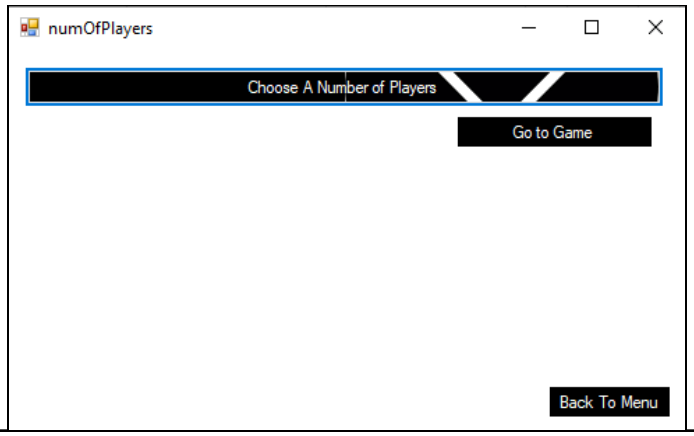
Case 3
Persona 2

Having read through the rules page the player thinks that they have enough

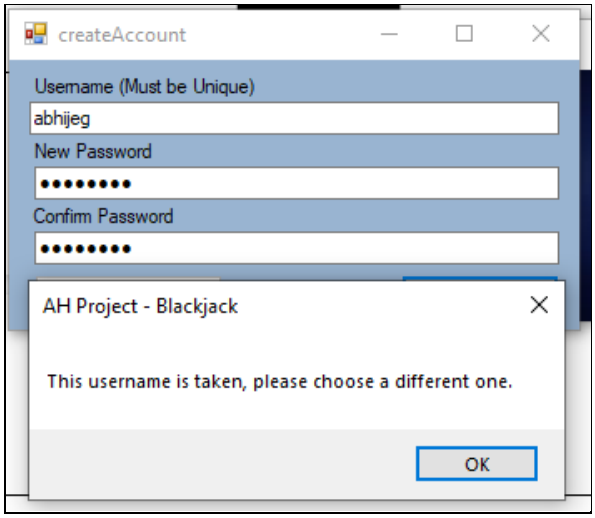


understanding of Blackjack to play a game

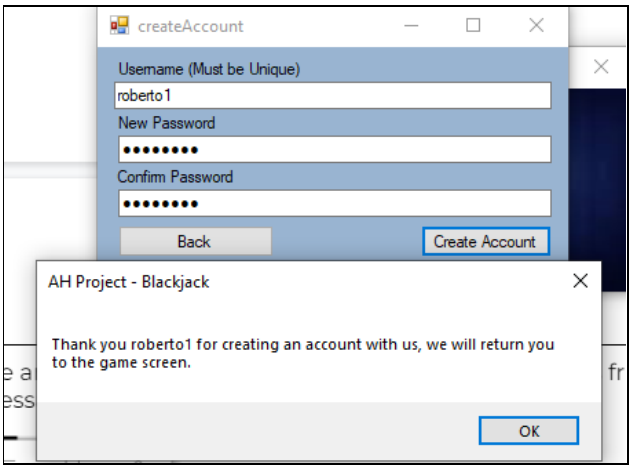
He goes to start a game with his friend but realises he doesn't have an account

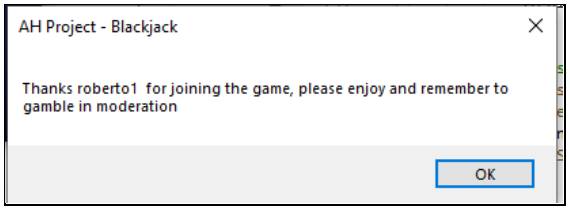


His friend signs into the game and he then goes to create an account, copying his friends name. The following error message is shown.

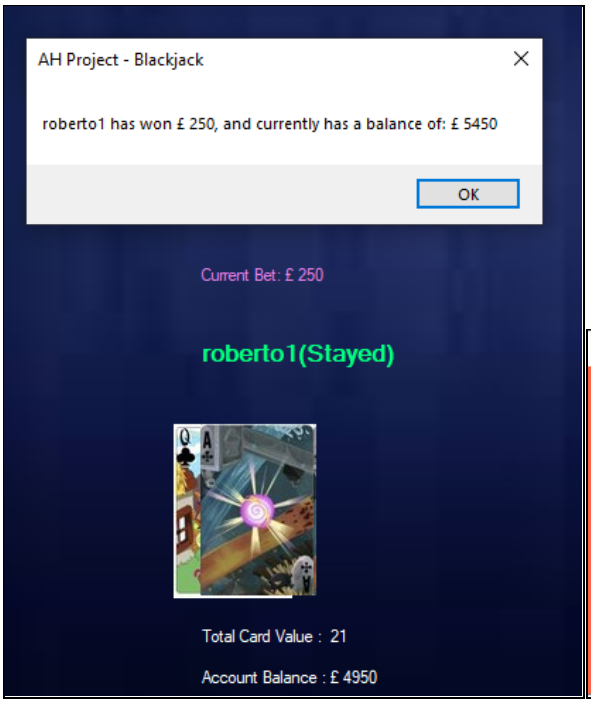


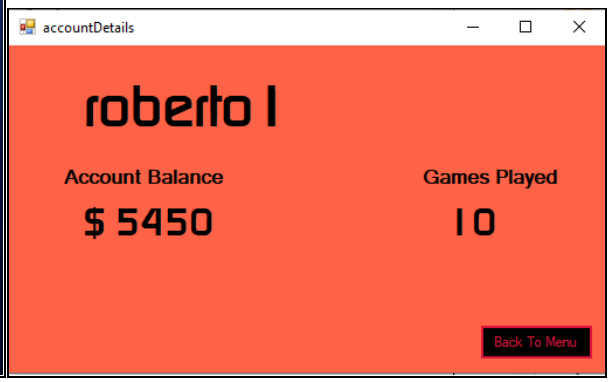
He then creates his own account and then signs in.





Then they play a few games and roberto is able to grasp and understand the game more by playing it further, gradually understanding when to take what action.





Since the 'Create Account' form's validation is working, and the rules page gives enough info for a new player to be able to understand the game, and the user interface is friendly enough for non-expert users to navigate easily.

Evaluation of the solution (5 marks)

You must now evaluate your solution.

Evaluation report (5 marks)

Produce a report to evaluate your solution. This should include:

- the **fitness for purpose** of your solution, discussing:
    - how closely your solution matches the **functional requirements**
    - the **results of your testing**
- the **maintainability** and **robustness** of your solution

During the implementation a number of errors that could have caused the program to be unfit for purpose were found. These were tested as the implementation was ongoing - Component Testing - and were all fixed testing stage. The functional requirements each being met without problems shows that the software is fit for purpose. It also meets all End-User requirements, which were tested by a group of end users with personas that were decided in the test plan, showing that the program not only works technically, but is User-Friendly and easy to learn. The testing with test cases can further prove this as a player who wasn't experienced in Blackjack was able to learn the game by reading over the rules and playing a few games to get the hang of it.
Each test case tests a different aspect of the program, everything that can be done in the program was done, and tested, within the 3 test cases. Each one with an End-User with the persona mentioned in the test plan.

Once the implementation was almost complete there was a requirement that was not met. A requirement specifying that an Advanced Higher algorithm must be included in the project. In order to fulfil this requirement the leaderboard form was added, it allowed a Bubble Sort to be added into the program without altering the design of the initial project by much. The requirements testing and test cases have shown that the leaderboard is a working part of the program and therefore meets the requirements set.

In terms of scopes and boundaries the program has met them , with the scope being slightly edited (Number of players in a game decreased) due to the speed and capabilities of the computers that were used, and the boundaries being maintained throughout.

The program is extremely robust, everytime the user has an option to type in a value instead of clicking a button, there is input validation to prevent the program from terminating due to an execution error. To sign in or create an account the user's inputs must meet certain conditions, if these are not met then an error message is shown explaining what the problem was, helping the user to input values that meet those conditions. When placing a bet in a game, if the value entered is too big or a value that is not an integer is entered an error message is shown, telling the user what the problem with their input was. Other than that the inputs for the program are provided by buttons with specified actions, leaving little possibility of execution errors.

The program is maintainable, it has internal commentary, briefly explaining what each section of the program does. It's modular, with most major algorithms or functions being contained inside a module. The code uses white space and indented making it easily readable for editing. A problem would be the amount of global variables used, it was necessary in order to pass data from one form to another but they reduce to portability of the program, making maintenance slightly more difficult, mainly component testing, (Due to separate forms not being usable unless they are connected to each other because of data being passed and held by each one), though in this circumstance it is not an issue as the program has already been fully tested.