Object-Oriented Programming  
Assignment 1  
Design Document

Jack Jones (22055565)

Contents

[Background 2](#_Toc157258368)

[UML Diagram 3](#_Toc157258369)

[Database Design 4](#_Toc157258370)

[Diagram 4](#_Toc157258371)

[Overview of Tables 5](#_Toc157258372)

[RoleTypes 5](#_Toc157258373)

[Users 5](#_Toc157258374)

[UserHashes 5](#_Toc157258375)

[UserBalance 5](#_Toc157258376)

[MessageStates 6](#_Toc157258377)

[Messages 6](#_Toc157258378)

[Flowcharts 7](#_Toc157258379)

[Log In 7](#_Toc157258380)

[Create User 8](#_Toc157258381)

[User’s Turn 8](#_Toc157258382)

[Computer’s Turn 9](#_Toc157258383)

[Screen Designs 10](#_Toc157258384)

[High Level Overview 10](#_Toc157258385)

[Low Level Overview 11](#_Toc157258386)

[Main Window 11](#_Toc157258387)

[Admin Home Page 12](#_Toc157258388)

[Detailed Add User Modal 13](#_Toc157258389)

[View Messages Modal 14](#_Toc157258390)

[Reset User Password Modal 14](#_Toc157258391)

[Set User Balance Modal 16](#_Toc157258392)

[Player Home Page 17](#_Toc157258393)

[Set Bet 18](#_Toc157258394)

[Main Game 19](#_Toc157258395)

[Low Level Create User 21](#_Toc157258396)

[Leaderboard Modal 22](#_Toc157258397)

# Background

The idea of my project is based around the card game Blackjack. The goal of the game is to get as closed to 21, without going over, which is known as going bust (Reynolds, 2020). Throughout the course of the game, both the dealer and the player will draw cards, and who ever has the highest score, without going bust, is the winner.

The game is player with a standard 52 card deck. The values of the cards are found in the below table.

|  |  |
| --- | --- |
| Card | Value |
| Ace | 1/11[[1]](#footnote-1) |
| 2-10 | Face Value e.g. 2=2, 5=5, 10=10 |
| Jack | 10 |
| Queen |
| King |

Table 1The Value of Cards in Blackjack

As well as the game itself, to adequately display principles of object-oriented programming, for example polymorphism and inheritance, I will also include a light administration system, allowing for the creation of users and their management.

# UML Diagram

Unified Modeling Language Diagrams (UML Diagrams) is a graphical language, used to document and visualise code/a software solution (Booch, Ivar Jacobson and Rumbaugh, 2005).

The UML for my project can be seen below, or at this link [here](https://drive.google.com/file/d/1vCR4C3DSONxBWwsfZVigrs1iTQHxVSES/view). It is also in the zip file provided under Design/Images/UML Diagram.jpg for clearer viewing.

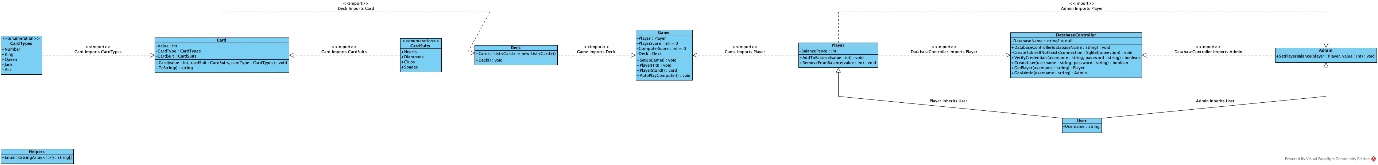


Image 1The UML Diagram for my Game

# Database Design

Due to wanting to incorporate users, and maintaining a scoring system, my project will need to include a database. As it meets my needs, and for added simplicity I have decided to use SQLite as my database solution.

SQLite is an open source, embedded, relational database solution which facilitates data storage, without needing to set up a database server (Kreibich, 2010). This means that it is much easier for me to write SQL queries and create the necessary tables, without needing to worry about setting up and maintain a server. The fact that my project is designed as a desktop application means that it is particularly well suited to using SQLite.

## Diagram

My database diagram can be found below, or at this [link](https://drive.google.com/file/d/1cvkjqGpKJUC_CjyzOghAFL8vSqPXOYwv/view). It is also found at Design/Images/Database Diagram.jpg in the provided zip file.

A computer screen shot of a computer

Description automatically generated

Image 2The Database Diagram for my Project Highlighting the Relationships Between Tables

## Overview of Tables

Below is a plaintext, description of the tables that will be used in my project.

### RoleTypes

* Id (Integer, Primary Key, Auto Increment)
* RoleName (Varchar (128), Not Null, Unique)

This table will store the types of users, at this stage, they will be:

* Admin
* Player

The table will maintain a one-to-many relationship with the [Users](#_Users) table.

### Users

* Id (Integer, Primary Key, Auto Increment)
* UserName (Varchar (128), Not Null, Unique)
* HashedPassword (Varchar (1024), Not Null)
* RoleTypesId (Integer, Foreign Key references RoleTypes.Id, Not Null)

This table will store users with the hash password, as well as the type of role they are.

The table will maintain the following relationships:

* A many-to-one relationship with the [RoleTypes](#_RoleTypes) table.
* A one-to-one relationship with the [UserHashes](#_UserHashes) table.
* A one-to-one relationship with the [UserBalance](#_UserBalance) table.
* A one-to-one relationship with the [Messages](#_Messages) table.

### UserHashes

* UserId (Integer, Foreign Key references Users.Id, Not Null, Unique)
* Hash (Varchar (1024), Not Null)

The hash will be a randomly generated string generated on creation and should be unique.

The table will have a one-to-one relationship with the [Users](#_Users) table.

### UserBalance

* UserId (Integer, Foreign Key references Users.Id, Not Null, Unique)
* BalancePence (Integer, not null, defaults to 10,000)

The balance will be stored using pence to avoid any issues that could occur due to rounding errors. The balance will be able to be negative to represent debt.

The table will have a one-to-one relationship with the [Users](#_Users) table.

### MessageStates

* Id (Integer, Primary Key, Auto Increment)
* StateName (Varchar (128), Not Null, Unique)

This table will store the different states a message can be in. At this point they are the below:

* Approved
* Declined
* Unread

The table will have a one-to-many relationship with the [Messages](#_Messages) table.

### Messages

* Id (Integer, Primary Key, Auto Increment)
* UserId (Integer, Foreign Key references Users.Id, Not Null, Unique)
* MessageStateId (Integer, Foreign Key references MessageStates.Id, Not Null, Unique)
* DateTime (Timestamp, not null, default current timestamp)

This table will have the following relationships with:

* A one-to-many relationship with the [Users](#_Users) table.
* A one-to-many relationship with the [MessageStates](#_MessageStates) table.

# Flowcharts

Flowcharts are a type of diagram used to plan, document, and analyse different process flows (Myler, 1998).

The flowcharts for my project can be [here](https://drive.google.com/drive/folders/1oQZa3A3GXAJYRrEXq-B3e-gVmvYxKClz?usp=drive_link), in the zip file under the folder Design/Images/Flowcharts and will be explored in the below subsections.

## Log In

The process flow for user log in can be found [here](https://drive.google.com/file/d/14rDFf4ADxGzydauoGpo8OIG3YlaL2LFH/view?usp=drive_link) or in the Flowcharts directory in the zip file as Login.png

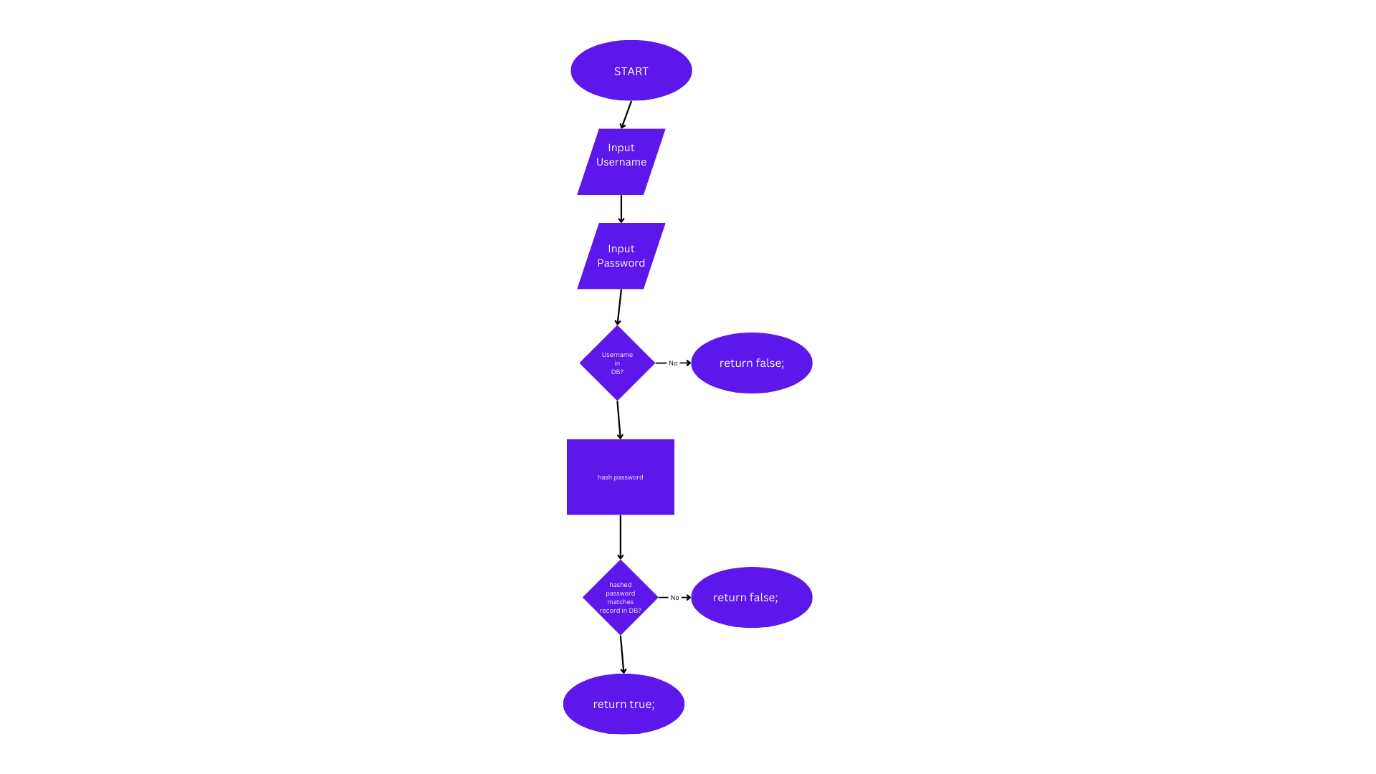


Image 3Login Flowchart

## Create User

The process flow for creating a user can be found [here](https://drive.google.com/file/d/1rJEppGvOk9zf3pssYcTstcceukTgK4DY/view?usp=drive_link) or in the Flowcharts directory in the zip file as CreateUser.png

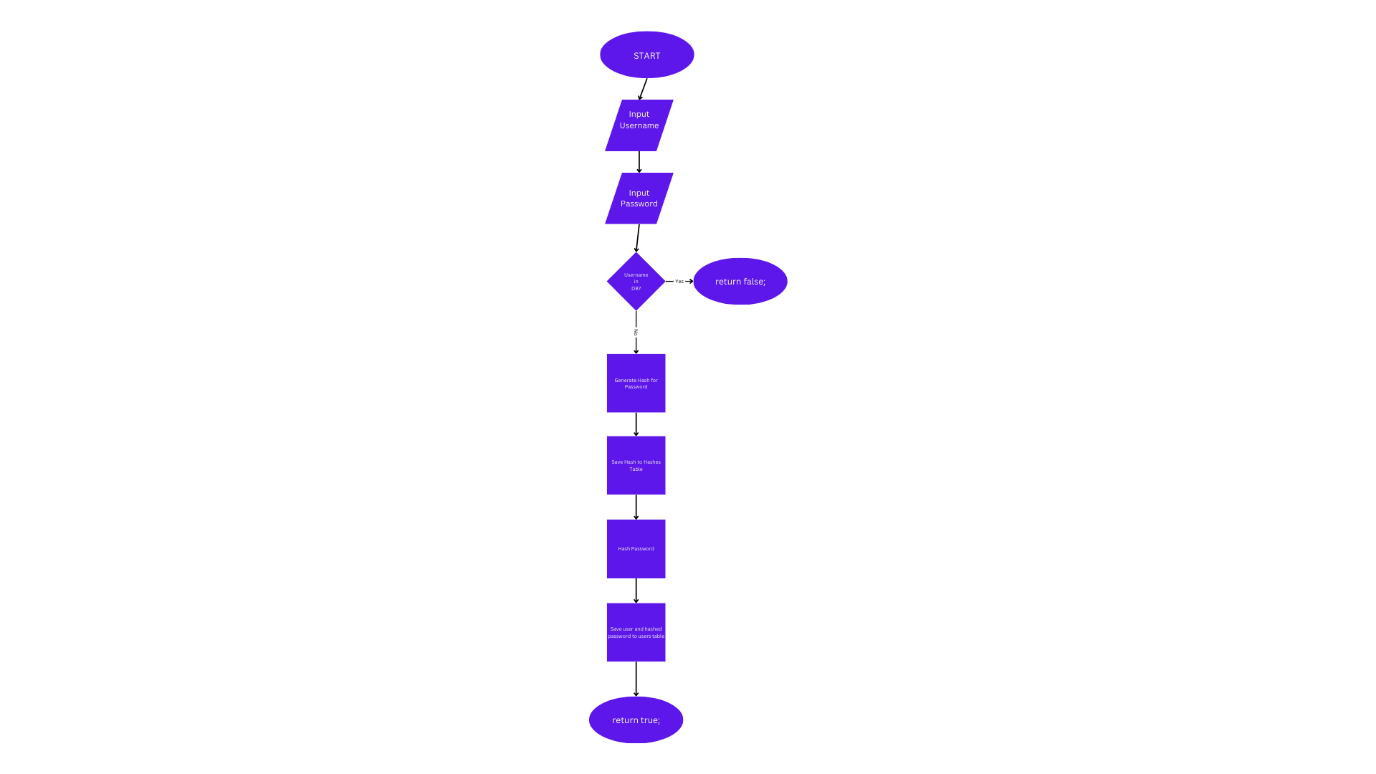


Image 4Flowchart to show Create User Flow

## User’s Turn

The process flow for the user’s turn can be found [here](https://drive.google.com/file/d/1QLuSxZ2o9nJPsoF0Xo42G8N8t4Dqad00/view?usp=drive_link) or in the Flowcharts directory in the zip file as UsersTurn.png

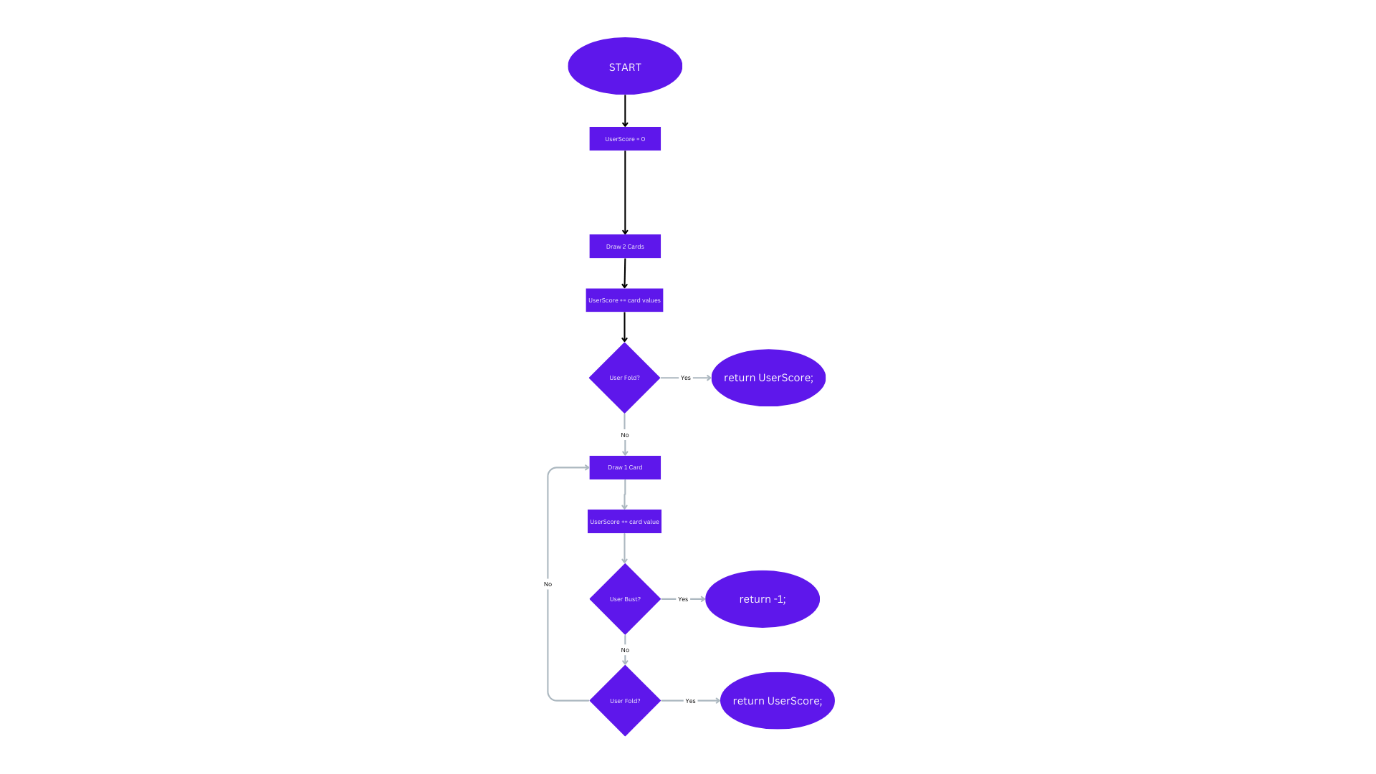


Image 5Flowchart to show the Player's Turn

## Computer’s Turn

The process flow for the computer’s turn can be found [here](https://drive.google.com/file/d/1mJNvxYiuZPrEHChl9UqiXWwCODKx_LWY/view?usp=drive_link) or in the Flowcharts directory in the zip file as ComputersTurn.png

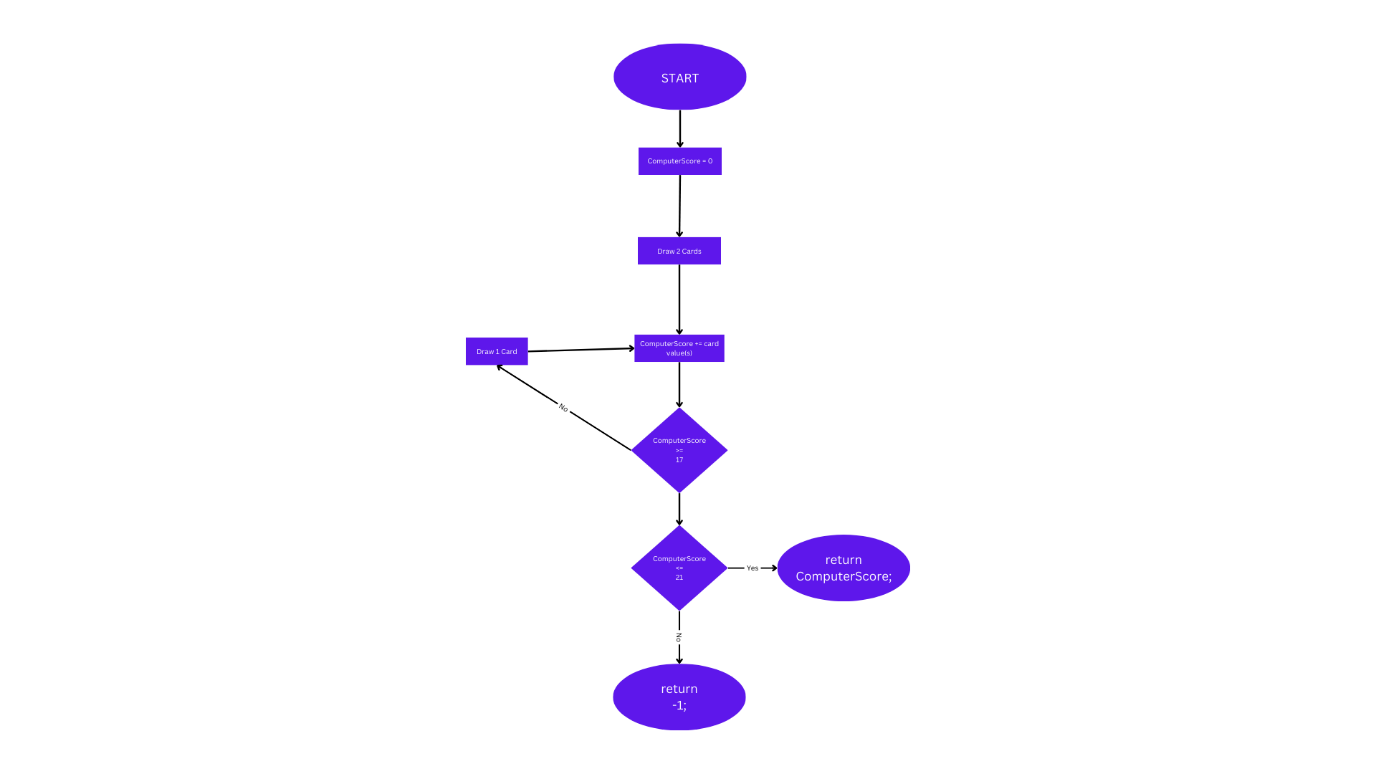


Image 6Flowchart to show the computer's turn.

# Screen Designs

## High Level Overview

Below is an overview showing how all the forms are connected. The [Low level Overview](#_Low_Level_Overview) section will go through the actual design of the forms. The file can be found in “Design/Images/ High Level Screen Designs” in the provided zip file, or:

* As a PNG in the directory as HighLevelScreenLayout.png or [here](https://drive.google.com/file/d/17XedrITdBviASapFyZgfd2EUlGKatrxv/view?usp=drive_link)
* As an SVG in the directory as HighLevelScreenLayout.svg or [here](https://drive.google.com/file/d/17XedrITdBviASapFyZgfd2EUlGKatrxv/view?usp=drive_link)
* As an Excalidraw file in the directory as HighLevelScreenDesign.excalidraw or through [drive](https://drive.google.com/file/d/1bq1HEqE0rBhrs3WiH_UhcXaNtTxUSOWZ/view?usp=drive_link) or on [Excalidraw](https://excalidraw.com/#json=bp5Jly7S8FrNZQ4hU3UZp,zwBIQCzs9VC4hl93lBPNsw).

A screenshot of a computer

Description automatically generated

Image 7High Level Screen Layout

## Low Level Overview

The design of the screens, where possible will follow the recommended best practice. For example, throughout the project I will use size 24 pixels font using Segoe UI, as outlined by Microsoft (Microsoft, 2021).

I will also ensure that the keyboard will be a viable input method where possible. For example, using the TabIndex and TabStop properties to control the order that a user can tab through a form (Microsoft, 2022).

The individual designs of each screen can be found in Design/Images/Screen Designs in the provided zip file, or [here](https://drive.google.com/drive/folders/1i0bZnH-yzjR100rUoYh5Ow3kUvEQIRFd?usp=drive_link).

### Main Window

* Dimensions (Width, Height): (800, 460)
* Can Resize: Only Minimise
* Image Path: MainWindow.png or [here](https://drive.google.com/file/d/1hj0yI8pTMU_29MW5M2XyUUQ05v9iQTTM/view?usp=drive_link)

A screenshot of a computer screen

Description automatically generated

Image 8The Screen Design for the Main Window

The main window screen is the first screen a user will see on load of the application. All text on this page is based on the standard sizes, and in the standard font.

There will be two labels, named **lblUsername** and **lblPassword** respectively. These will be used to label the form inputs.

One of the form inputs will be **txtUsername** which is a TextBox. This will take the username that the user will attempt log in as. It will be accessible through the tab key and will have a tab index of 0, meaning it will be the first item accessible through the tab key. It will be approximately 300 pixels in length. It will have the Wrap property set as NoWrap, meaning that it will not allow the text to line break.

The other form input will be **pbPassword** which is a PasswordBox input. A PasswordBox has been selected over a TextBox, as it easily allows for masking of the input, for added security. This input will take the user’s password. It will be accessible through the tab key and will have a tab index of 1, meaning it will be the second item accessible through the tab key. It will be approximately 300 pixels in length.

The form will also contain four different buttons. They will each have a width of around 200 pixels, and a height of around 60 pixels.

The first button will be **btnLogIn**. This button will begin the log in process. It will check the inputted credentials, and if they are valid will allow the user to log in. This will mean that if the attempted log in is that of an admin it will show the [Admin Home Page](#_Admin_Home_Page), whereas if the user is a player, it will show the Player Home Page. If the credentials are incorrect, it will show a pop-up message showing the reason. It will be a tab stop and have a tab index of 2, meaning it will be the third item accessible through the tab key.

The second button will be **btnCreateUser**. This button will open the form to begin the process of creating a player user by opening the [Low Level Create User form](#_Low_Level_Create). If the user has already entered a password, this will be carried across to the create user page. It will be a tab stop and have a tab index of 3, meaning it will be the fourth item accessible through the tab key.

The penultimate button will be the **btnViewLeaderboard**. This button will open a [leaderboard modal](#_Leaderboard_Modal) showing the top 5 players ranked by balance. It will be a tab stop and have a tab index of 4, meaning it will be the fifth item accessible through the tab key.

The final button on this form will be the **btnViewGuideBook**. This button will open the PDF guidebook using the machine’s default PDF viewer. It will be a tab stop and have a tab index of 5, meaning it will be the sixth item accessible through the tab key.

### Admin Home Page

* Dimensions (Width, Height): (800, 500)
* Can Resize: Only Minimise
* Image Path: AdminHomePage.png or [here](https://drive.google.com/file/d/1QUR3XSn2ZQMuEeSFuj5WIpUcVdf0lXNr/view?usp=drive_link)

A screenshot of a computer

Description automatically generated

Image 9The Form Design for the Admin Home Page.

The Admin Home Page form contains functionality that would be useful for an admin. All text on this page is based on the standard sizes, and in the standard font.

The form will contain a single label called **lblWelcomeMessage**. This will show a welcome message to the logged in admin in the form “Welcome {Username}” where {Username} is the username of the logged in admin.

The form will also contain four main buttons, each of a width of approximately 180 pixels. The first button is the **btnResetAllCredit**. When this button is pressed it will prompt the user to make a confirmation about whether they truly wish to reset all balances of players to £100. If the user selects “Yes”, this change will be made. It will be accessible through the tab key and will have a tab index of 0, meaning it will be the first item accessible through the tab key.

The next button will be the **btnCreateUser**. This button will open the modal [Detailed Add User Modal](#_Detailed_Add_User). This differs to the Low Level Create User, as it allows for admins to be created, and to set a custom balance for player users. It will be accessible through the tab key and will have a tab index of 1, meaning it will be the second item accessible through the tab key.

The penultimate button will be the **btnLogOut**. This button will return the logged in user to the [Main Window](#_Main_Window). It will be accessible through the tab key and will have a tab index of 3, meaning it will be the fourth item accessible through the tab key.

The last button on this form will be the **btnViewMessages**. This button will the modal View Messages Modal, which will show balance reset messages from users, and allow the admin to approve or decline them. It will be accessible through the tab key and will have a tab index of 4, meaning it will be the third item accessible through the tab key.

The form will also contain a list view named **lvPlayers**. This will be populated with the player users. It will have a width of approximately 675 pixels, and a height of around 150 pixels. It will have a tab index of 2 meaning it will be the third item accessible.

The list view will contain three columns, “Username”, “Reset Password” and “Set Balance. The username column will simply contain the username of the user. The buttons however will open modals, one for a [password reset](#_Reset_User_Password) of that reset, and one to [specify the balance of that user](#_Set_User_Balance) respectively.

### Detailed Add User Modal

* Dimensions (Width, Height): (600, 450)
* Can Resize: No Resize
* Image Path: DetailedAddUserModal.png or [here](https://drive.google.com/file/d/1TMgFlWpxvJMnA2Z7O6Xg_1KrtevaM2E-/view?usp=drive_link)

A screenshot of a computer screen

Description automatically generated

Image 10Form Design for Detailed Add User.

The detailed add user modal is used by admins to create a user of either an admin, or a player with a custom balance. All text on this page is based on the standard sizes, and in the standard font.

The modal contains 5 different labels: **lblUsername**, **lblPassword**, **lblPasswordConfirm**, **lblRoles**, and **lblBalance**.

There are also several input fields on this form. One of the input fields is a text field **txtUsername**. This will be used to set up the username of a new user. It will have a length constraint that it must be more than six characters long and must also be unique. It will have a tab index of 0 and so will be the first item accessible via the tab.

There will be two password fields **pbPassword** and **pbPasswordConfirm**. These will be masked to improve security. There will be several constraints based on length, casing, containing certain characters, and matching each other. They will have a tab index of 1 and 2 respectively.

There will also be a combo box **cbRoles**. This will allow the user to select one of the user role types from the database. A combo box has been chosen, to add a constraint that the value must be one of the allowed values. It will have a tab index of 3.

Th final input will be a DoubleUpDown control from the WPF Toolkit (Xceed, 2020). The control will be named **dudBalance**. This control, allows the user to input a number (specifically a double), using up and down arrows. This control type has been chosen as it allows a specification of a minimum value (0), a maximum value (100) and an increment (0.01). It also enforces that the value is a double. The tab index will be 4.

There will be two buttons on this form **btnCancel** and **btnSave**. Each of these buttons will have a width of around 170 pixels. The first button will close the current form and return to the [Admin Home Page](#_Admin_Home_Page), whereas the second button will validate the form, if it passes validation save the user, and then clear the form. They will have tab indices of 5 and 6 respectively.

### View Messages Modal

* Dimensions (Width, Height): (800, 450)
* Can Resize: Can Minimise
* Image Path: ViewMessagesModal.png or [here](https://drive.google.com/file/d/1w31zEW5qI5hSbQV8HYN5O2PVqSCk1Il0/view?usp=drive_link)

A screenshot of a computer

Description automatically generated

Image 11The Form design for the View Messages Modal

The View Messages Modal will be a very simple modal off the admin home page. It will contain a single item, a list view named **lvMessages**.

This list view will be populated from the messages table. One column will contain the timestamp of the message in the form yyyy-MM-dd HH: mm. There will then be two columns for the message sender’s username, and their current balance. The third column will contain two buttons, one to approve the message and one to decline it. On message approval, the users balance will be reset to £100.

### Reset User Password Modal

* Dimensions (Width, Height): (700, 450)
* Can Resize: Can Minimise
* Image Path: ResetUserPasswordModal.png or [here](https://drive.google.com/file/d/1VJ55Sa0yNecStbZ37iVvtVEqBJs0SjzB/view?usp=drive_link)

A screenshot of a computer screen

Description automatically generated

Image 12Form Design for Reset User Password Modal

The Reset User Password modal will be deployed from the list view on the [admin home page](#_Admin_Home_Page). There will be four labels on this modal, three of which will be labelling the input fields (**lblUsername**, **lblPassword**, **lblPasswordConfirm**), whereas the other (**lblUserHolder**) will be populated with the username of the selected user from the list view.

There will be two password fields **pbPassword** and **pbPasswordConfirm**. These will be masked to improve security. There will be several constraints based on length, casing, containing certain characters, and matching each other. They will have a tab index of 0 and 1 respectively.

There will be two buttons on this form **btnCancel** and **btnSave**. Each of these buttons will have a width of around 170 pixels. The first button will close the current form and return to the [Admin Home Page](#_Admin_Home_Page), whereas the second button will validate the form, if it passes validation save the user, and then return to the [Admin Home Page](#_Admin_Home_Page). They will have tab indices of 2 and 3 respectively.

### Set User Balance Modal

* Dimensions (Width, Height): (500, 450)
* Can Resize: Can Minimise
* Image Path: SetUserBalanceModal.png or [here](https://drive.google.com/file/d/1lSVqZVruxzcuFz7Y6a8YWYfxx5rpbQ54/view?usp=drive_link)

A screenshot of a computer

Description automatically generated

Image 13Form Design for the Set User Balance Modal

The Set User Balance modal will be deployed from the list view on the [admin home page](#_Admin_Home_Page). There will be three labels on this modal, two of which will be labelling the input fields (**lblUsername**, **lblBalance**), whereas the other (**lblUserHolder**) will be populated with the username of the selected user from the list view.

The form will also have an input that is a DoubleUpDown control from the WPF Toolkit (Xceed, 2020). The control will be named **dudBalance**. This control, allows the user to input a number (specifically a double), using up and down arrows. This control type has been chosen as it allows a specification of a minimum value (0), a maximum value (100) and an increment (0.01). It also enforces that the value is a double. The tab index will be 4.

There will be two buttons on this form **btnCancel** and **btnSave**. Each of these buttons will have a width of around 170 pixels. The first button will close the current form and return to the [Admin Home Page](#_Admin_Home_Page), whereas the second button will validate the form, if it passes validation save the user, and the return to the [Admin Home Page](#_Admin_Home_Page).

### Player Home Page

* Dimensions (Width, Height): (800, 450)
* Can Resize: Can Minimise
* Image Path: PlayerHomePage.png or [here](https://drive.google.com/file/d/1qTHGS30XQB36qWoDQTTfhoYJF7PMTJgT/view?usp=drive_link)

A diagram of a user

Description automatically generated with medium confidence

Image 14Form design for the Player Home Page

The Player Home Page form contains functionality that would be useful for players. All text on this page is based on the standard sizes, and in the standard font.

There will be three labels on this form. One of these labels will be **lblWelcome** which will show a welcome message to the current user. There will be a control **lblCurrentBalance** as a descriptive label for **lblcurrentBalanceValue** which will be populated with the current user’s current balance.

There will also be three buttons on the form. The first being **btnLogOut** which will have a tab index of 0. This button will always be visible and enabled. On click it will return the user to the [main window](#_Main_Window).

The other two buttons are dependent on the user’s balance. The first of these is **btnPlay** which will have a tab index of 1. When this is pressed it will send the user to the Set Bet page. When the user’s balance is less than 0, this button will be disabled.

The final button on this page will be **btnSendMessage** which will have a tab index of 2. This button will be hidden if the user’s balance is greater than 0. This button when pressed will send a balance reset request message to admins.

### Set Bet

* Dimensions (Width, Height): (500, 450)
* Can Resize: Can Minimise
* Image Path: SetBet.png or [here](https://drive.google.com/file/d/1wzN-vWMnAC3Le1ik61b1kkBcSB5ni4CV/view?usp=drive_link)

A diagram of a computer program

Description automatically generated

Image 15Form Design for Set Bet Screen

The set bet screen comes from the [player home page](#_Player_Home_Page). It is used for the player to be able to set their bet amount.

It will have two static labels **lblUsername** and **lblBet.** There will also be **lblUsernameValue** which is populated with the logged in user’s username.

There is also a slider **sBetAmount** which will have a tab index of 0. This control is used for the user to select the bet amount of the user. Around this are three labels **lblMinimumValue**, **lblCurrentValue**, **lblMaximumValue**, which will be used to show the minimum, current, and maximum values of the slider. The maximum value of the slider will be the entire balance of the current user. The minimum value will be zero, and the default current bet amount will be the floored (Graham, Donald Ervin Knuth and Patashnik, 1989) value of half the user’s balance.

The form will also have two buttons **btnCancel** and **btnSave** which will have tab indices of 1 and 2 respectively. The cancel button will return the user to the [player home page](#_Player_Home_Page), whereas the save button will set the user’s bet and take the user to the main game.

### Main Game

* Dimensions (Width, Height): (750, 500)
* Can Resize: Can Minimise
* Image Paths:
  + MainGameOnLoad.png or [here](https://drive.google.com/file/d/1D2NVK06pLMsDW8RkaKv3b5SngVAHB8J6/view?usp=drive_link)
  + MainGameOnPlay.png or [here](https://drive.google.com/file/d/1RUAMTTkieCTxEFS-ToU-3CTzquZv-Fj1/view?usp=drive_link)

A close-up of a white rectangular object

Description automatically generated

Image 16Form Design for the Main Game on Load

A screenshot of a computer screen

Description automatically generated

Image 17Form Design for the main Game after the play button is pressed.

The main game window is where the game of blackjack will be played. On initial load all elements except **btnPlay** will be hidden. This button will have a tab index of 0. When this is pressed the play button is hidden, and all other elements have their visibility set to visible.

Post play button press, the screen will roughly be able to split into around 4 sections: The Computer Cards, the separator, the player cards, and the action buttons.

The computer card section has several labels. One of these is **lblComputerScore** which is used to label **lblComputerScoreValue**, the latter of which is used to showcases the computer’s current score. Before the computer’s turn, this will be shown as a question mark.

There will also be the **lblComputerCardsValue** which will show the cards of the computer. This label will have a font size of 100 pixels. It will be populated using Unicode card characters (Unicode, 2018). When the play button is pressed, it will show two cards, one of which will be face down. There will also be a separator named **separator** with a width of 800 pixels, which is separator the computer section and the player section.

The player section is very similar to the computer section. It has the same types of labels but named appropriately as **lblPlayerScore**, **lblPlayerScoreValue**, **lblPlayerCardsValue**. A differentiation is that when the play button is pressed, both the cards dealt to the user are face up.

Finally, there are three buttons. The first of these is **btnFold**, which when clicked will return the user to the [player home page](#_Player_Home_Page), and as a penalty reduce the player’s balance by half the bet amount. This will have a tab index of 1.

The next button is **btnHit**, which will deal the user another card. It has a tab index of 2. The user can request as many cards as they want until either they hit 21 and win, press the stand button, or go bust and lose. On a win, the player is rewarded with their bet amount, and returned to the [player home page](#_Player_Home_Page). On a loss, the player loses their bet amount, and returned to the [player home page](#_Player_Home_Page). When a player receives a new card, the **lblPlayerScoreValue** and **lblPlayerCardsValue** will be updated to show the new values.

The final button is **btnStand**, which will end the user’s turn and start the computers. It will have a tab index of 3. The computer will be dealt a card, and as per the rules of blackjack, they must continue until they go bust or reach 17 or more, at which point they must stand. Like with the player, the computer’s **lblComputerScoreValue** and **lblComputerCardsValue** will update. On a win, the player is rewarded with their bet amount, and returned to the [player home page](#_Player_Home_Page). On a loss, the player loses their bet amount, and returned to the [player home page](#_Player_Home_Page).

### Low Level Create User

* Dimensions (Width, Height): (800, 450)
* Can Resize: Can Minimise
* Image Path: LowLevelCreateUser.png or [here](https://drive.google.com/file/d/1Xdn93OkeB8qOwd-yZsh5ne-s-36JGjag/view?usp=drive_link)

A screenshot of a computer program

Description automatically generated

Image 18Form Design for Low Level Create User

The low level create user form, allows the creation of a new player without being logged in as an admin. It comes from the [main window](#_Main_Window).

On this form there will be five labels. Four of these labels (**lblUsername**, **lblPassword**, **lblPasswordConfirm**, **lblRoleType**), will be used to label different form inputs. The final label is **lblRoleTypeValue** which will be hardcoded to Player, to show that the form is being used to create a user.

There are also three input fields on this form. One of the input fields is a text field **txtUsername**. This will be used to set up the username of a new user. It will have a length constraint that it must be more than six characters long and must also be unique. It will have a tab index of 0 and so will be the first item accessible via the tab. If the user had entered a username on the main window, before clicking the create user button, it will be prepopulated with that value.

There will be two password fields **pbPassword** and **pbPasswordConfirm**. These will be masked to improve security. There will be several constraints based on length, casing, containing certain characters, and matching each other. They will have a tab index of 1 and 2 respectively.

There will also be two buttons on the form. One will be **btnBack** which will have a tab index of 3. When pressed, it will return the user to the [main window](#_Main_Window). The other button is **btnSave** which will validate the form, and if it is valid save the user, before returning to the [main window](#_Main_Window), where the username field will be populated.

### Leaderboard Modal

* Dimensions (Width, Height): (800, 450)
* Can Resize: No Resize
* Image Path: LeaderboardModal.png or [here](https://drive.google.com/file/d/1ZUXPwupa2uBZjrhvJIPC27IXDCOU8TOG/view?usp=drive_link)

A screenshot of a computer

Description automatically generated

Image 19The Form Design for Leaderboard Modal

The leaderboard modal will come from the [main window](#_Main_Window), and will be a simplistic form. It will contain a single element, a list view named **lvLeaderboard** which will have a tab index of 0.

The leaderboard will have three columns, a rank, the user’s username, and their balance. As the database will store the balance in pence, the balance column will require a mask to conform it to pence. The leaderboard will take up around 80% of the form’s width.

# References

* Booch, G., Ivar Jacobson and Rumbaugh, J. (2005). *UML: The Unified Modeling Language user guide: [thoroughly updated - the ultimative tutorial to the UML from the original designers]*. Upper Saddle River, Nj: Addison-Wesley.
* Graham, R.L., Donald Ervin Knuth and Patashnik, O. (1989). *Concrete Mathematics*. Addison Wesley Publishing Company.
* Kreibich, J.A. (2010). *Using SQLite*. Beijing; Farnham: O’Reilly.
* Microsoft (2021). *Typography in Windows apps - Windows apps*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/windows/apps/design/style/typography.
* Microsoft (2022). *Control.TabIndex Property (System.Windows.Controls)*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/dotnet/api/system.windows.controls.control.tabindex?view=windowsdesktop-8.0 [Accessed 26 Jan. 2024].
* Myler, H.R. (1998). *Fundamentals of engineering programming with C and Fortran*. Cambridge, U.K.: Cambridge University Press.
* Reynolds, T. (2020). European Blackjack: how to play and where to play it in the UK. *The Telegraph*. [online] 14 Feb. Available at: https://www.telegraph.co.uk/betting/casino-guides/blackjack/european-blackjack/.
* Unicode (2018). *Playing Cards*. [online] www.unicode.org. Available at: https://www.unicode.org/charts/nameslist/n\_1F0A0.html.
* Xceed (2020). *DoubleUpDown*. [online] GitHub. Available at: https://github.com/xceedsoftware/wpftoolkit/wiki/DoubleUpDown.

1. An ace will be valued at 11, unless it would cause the player/dealer to go bust, in which case it is valued as 1. [↑](#footnote-ref-1)