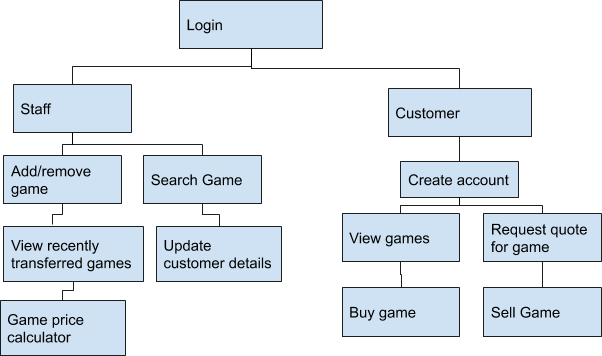
Design

**Introduction**

In this section I will discuss the various input and output data that needs to be entered into the proposed system and also the data that will be outputted from the system. I will also design well annotated user interfaces (i.e menu, data entry screens, output reports, etc.) that will be replicated in the program. In addition, this section will also describe processes that will be required to transform the data into the required output. Below is a detailed explanation and illustrations of the data input required.

Breaking down of the proposed system



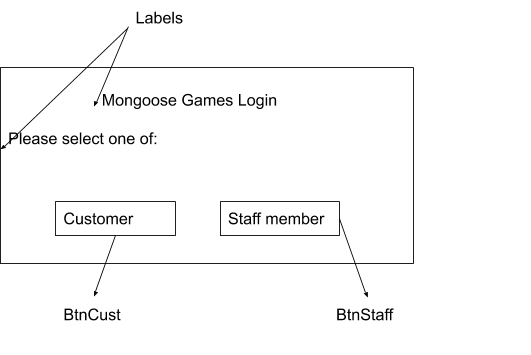
The diagram above is an illustration of my proposed solution which has been broken down into manageable sub-programs. I have chosen to decompose my system in order to program certain parts individually and ensure they’re working before moving to the next one, instead of doing all of them at once and being faced by errors which could be produced from any sub-program. It’s a lot more feasible to develop the system in this manner, with each sub-program representing a module, it allows me to manage my workload and maintain my code in a more organised and professional way. I have generalised some modules into one sub-program to simplify the decomposition diagram and since they fit under a similar procedure to develop.

**Login details**

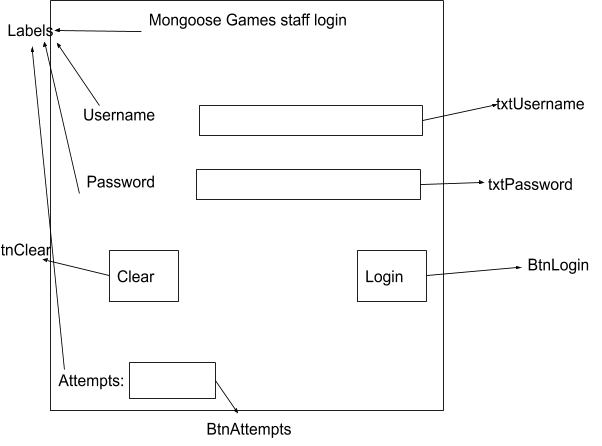
The layout below represents the design of the screen that both staff members and customers will see at first. From this screen the user can select if they are an employee or a customer. They will then be directed to their respective login screen depending on their selection.

|  |  |
| --- | --- |
| Data | Explanation |
| Username | This data will be needed by either the customer or staff member in order to log into their account. This would need to match with the password in order to grant the user access to the system |
| Password | The password needs to be inputted into the system so that either a staff member or customer can access their accounts. The system will check that the password matches up with the username inputted and then it would either grant or deny access to the system depending on if their credentials are correct or not |

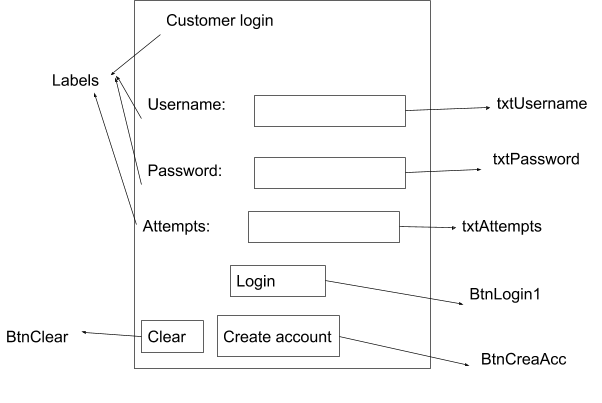
This is a screen that both staff members and customers will see at first. From this screen the user can select if they are an employee or a customer. They will then be directed to their respective login screen depending on their selection.



The layout below represents a screen is the login screen for the staff members. Their username and password are necessary for login.

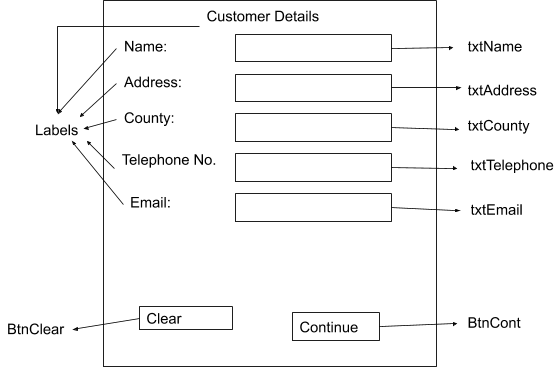


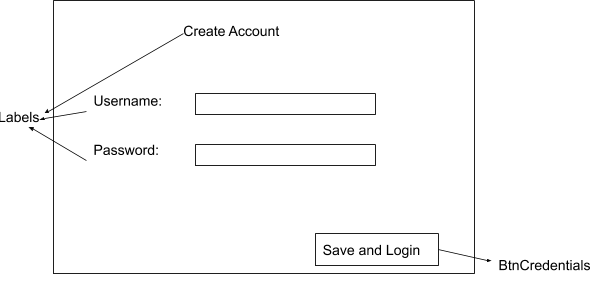
The screen design below is the login screen for the customers. The functionality to create an account will be added on the Create Account button when a customer accesses the system for the first time(form shown after). Once they create their account they are able to login using their respective username and password.



|  |  |
| --- | --- |
| Data | Explanation |
| Name | The customer’s name will be required to be entered into the system in order for an account to be created for them so that the system can distinguish each customer by their name. |
| Address | The system will also require the member’s address in order for the store to be able to contact them by post about any matter |
| County | The county would be entered so that the store can understand where the customer comes from in order to have an idea for their travel distance. |
| Phone Number | The customer’s phone number is essential to be stored in the system so that members of staff can contact them with ease, which is much more convenient and effective than by post. |
| Email | The customer’s email address will be needed so that general information can be sent to the customer such as when a game on their wishlist comes in stock and replying on a quote for a game(s) they have. |

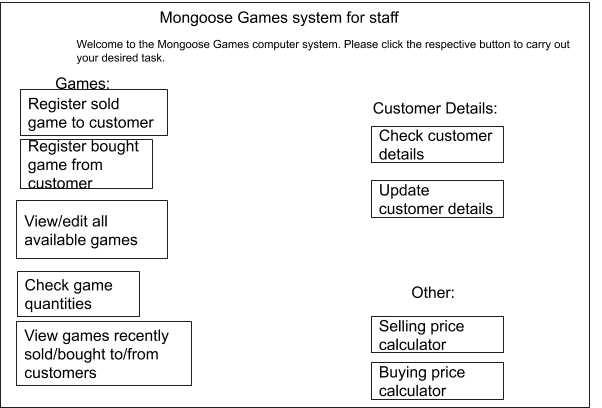
The details that are entered will go into a database that stores all of the member’s data. When BtnCont is clicked, the form will be hidden and the next form will appear prompting the customer to create their account with their desired credentials. These credentials can then be used prior to grant them access to the system.





After the save and login button is pressed, the user will be redirected to the customer login page in order to use their newly created account to access the system.

This is the main menu page that is viewed by staff members and allows them to perform various tasks that had previously needed to be done manually on notepads. They choose their required task by clicking on the respective button.



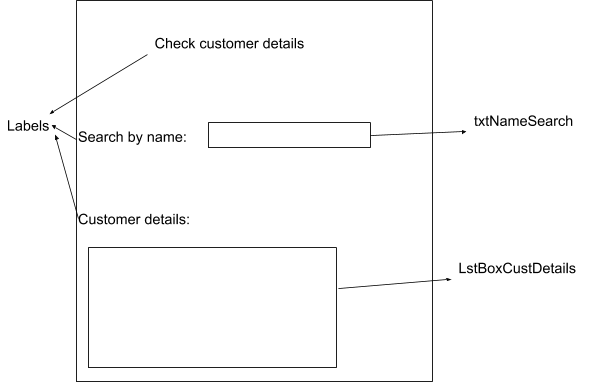
**Output**

**Introduction**

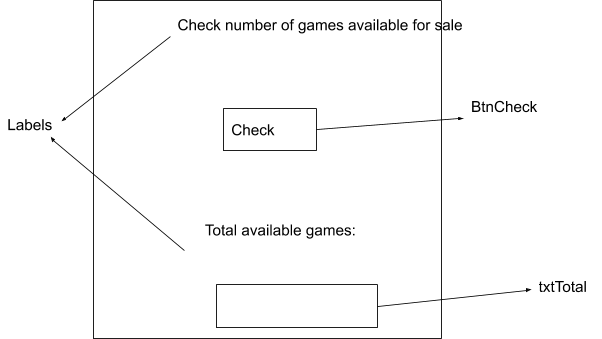
This section will describe the outputs that the proposed system should produce. These will be generated by the system and then show when they are requested e.g. the total number of available games will be shown when the staff member wants to check this data.

|  |  |
| --- | --- |
| Data | Explanation |
| Output all customer details | This output will show the customer details of the certain member that has been searched. This data will help a member of staff to find out the relevant information they need about a member in order to contact them. For example, the member of staff may need a customer’s email to let a customer know that a game they preordered is ready for collection. |
| Output all available games | This output will show the staff member all of the games currently on the system |
| Output all games games recently sold/bought to/from customers | This output shows the member of staff all of the games that have been sold to or bought from customers. This is important information for the store so that they can keep track of what games are being sold and bought in order to prevent overstock/ scarcity of games |
| Selling/buying price calculator. | The buying/selling price will be calculated based upon what the member of staff enters into the system. For example, if a game is 10 years old then this would be taken into account into the algorithm as well as other factors and have an impact on the price e.g. If a game was released 10 years ago at the price of £50, it will be divided by 10. This would then give a total price of £5 that the store would have to pay for a game owned by a customer. However, after a game is bought by a customer, it’s not sold at the same price as it wouldn’t produce profit, and so it will be automatically incremented by a set amount e.g. if a game was bought from a customer for £5, the selling price will be multiplied by 1.5x when sold. This would then give a total price of £7.5 a customer would have to pay. |
| Output all game quantities | This output gives the number of games that are available to be bought. This is important information for the store so that they can keep track of the number of games they have available to ensure a sufficient amount is in stock. |

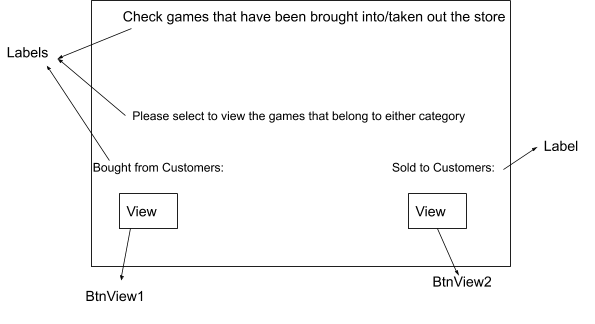
The layout below is the design of an interface which allows the staff member to search a customer by name and then view their relevant information



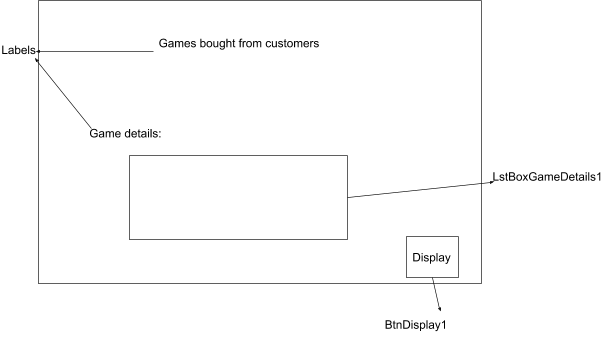
This screen allows the staff member to check the quantity of games currently in stock



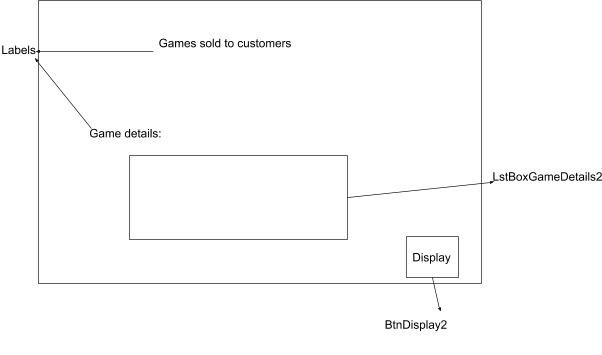
This interface below allows the staff member to view recently sold or bought games, by clicking the respective button.



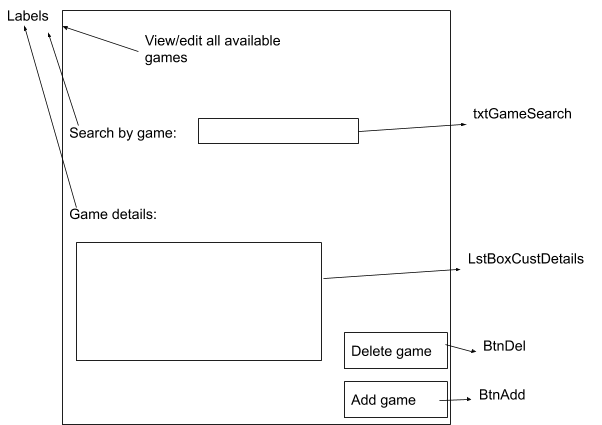
This screen is shown when “BtnView1” is pressed



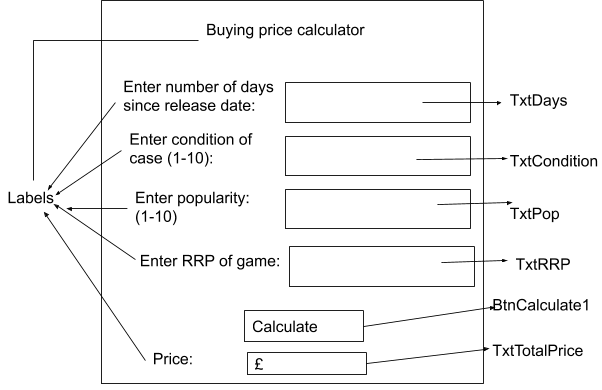
This screen is shown when “BtnView2” is pressed and shows all games sold to customers



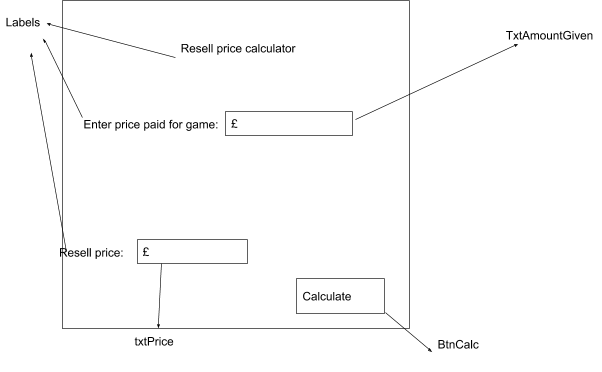
This screen allows the staff member to search for a game by name and then view its respective details, they can also delete or add a game but that will be covered in the input section.



This screen allows the staff member to receive a price for a game after entering a few factors of the game that determines this price.



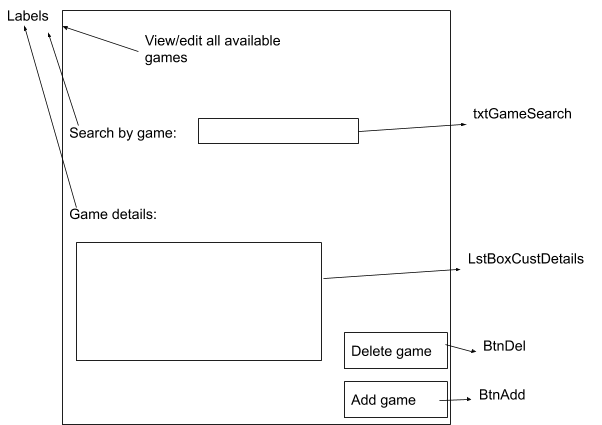
This screen allows the staff member to receive a resell price for a game that was bought from a customer



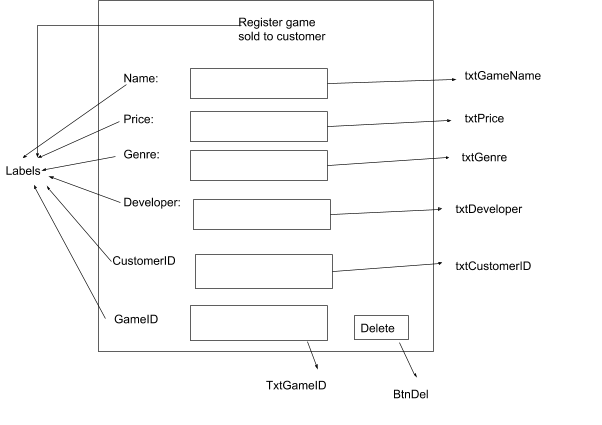
**Input**

|  |  |
| --- | --- |
| Data | Explanation |
| Register sold game to customer | This input will enable the staff member to register that a game was sold to a customer which will subsequently delete it. This function is necessary as it ensures that all sold games are deleted from the database to avoid confusion, it also allows the staff member to keep track of what games have been sold. |
| Register bought game from customer | This input will enable the staff member to register that a game was bought from customer and add it to the system. This function is necessary as it ensures that all games bought from customers are appended to the database in order for them to be sold again but at a higher price. |
| Update customer details | This input will enable the staff member to update the details of a customer. This function is necessary to ensure that up-to-date customer details are stored in the system in case of a change of address/telephone number to maintain contact |

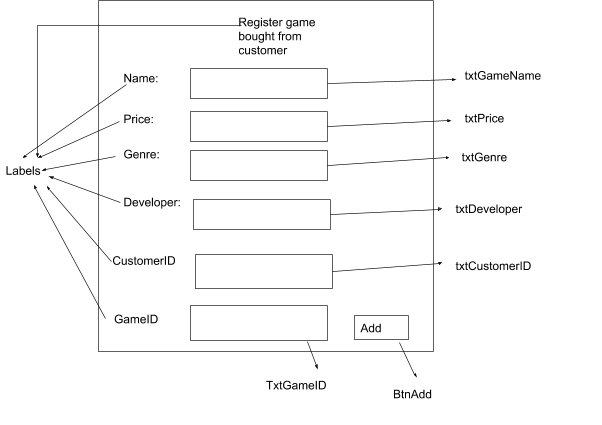
This screen allows the staff member to search certain games (covered in the output section) and then allows them to delete or add new games.



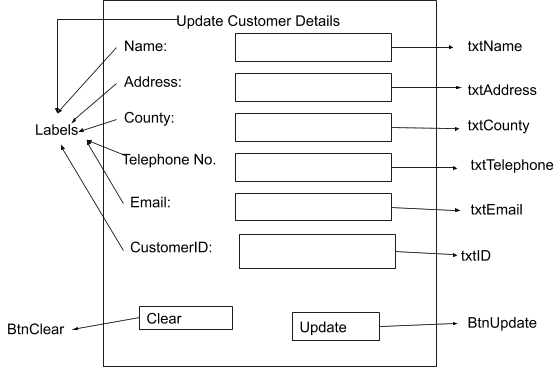
This is the screen that shows up when the staff member clicks “Delete game”



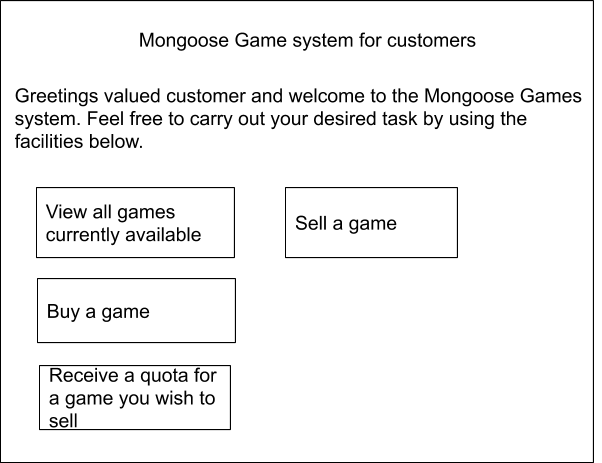
This is the screen which shows up after the staff member clicks “Add game”



This is the screen which shows when a staff member wants to update the information regarding a customer.



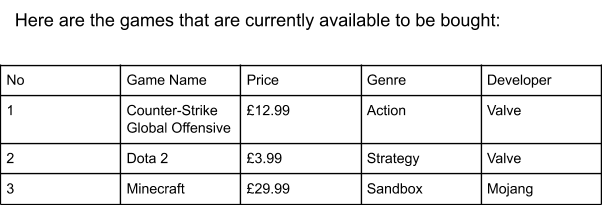
This is the main screen that is shown to the customers of Mongoose Games. The customer can then navigate through the system by clicking on the respective button to carry out their desired task.



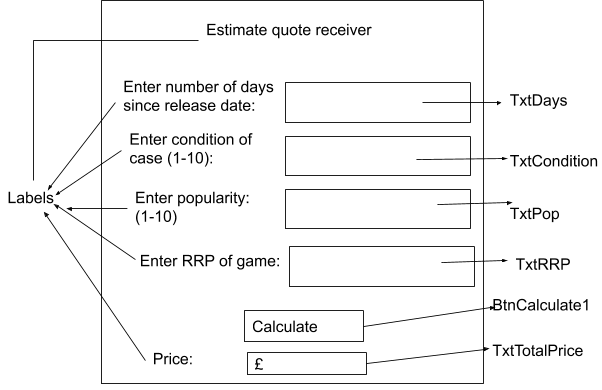
**Output**

The table below describes the outputs the system will produce for the customer during their use of the system.

|  |  |
| --- | --- |
| Data | Explanation |
| View all games currently available | This output will allow the customer to view all of the games that are available to be bought. This is necessary for when the customer wants to buy a game. |
| Receive a quote | This output will let the customer know the worth of their game by using the preset algorithm. This is necessary as it allows the customer to know how much they will get for their game without commiting to selling it. |
| Price | This output will show the customer how much a game they are buying is worth |
| GameName | This output will show the customer the name of a game they’re buying |
| Genre | This output will let the customer know what to expect from the game by looking at the genre |
| Developer | This output will let the customer know who made the game which could lead to their purchase |



This screen will allow a customer to check how much they will get for selling their game in store. They might want to do this as it will give them an idea for how much they will get for their game which will allow them to decide if they want to sell it or not. Due to the fact that it’s a subjective scale, or the customer might lie, the price will always ultimately be decided by a staff member.



**Files/ Data Structure**

This section will describe the file structure that will be used to store the data that will be entered into the system as well as the structure of the arrays that will be used to temporarily store data in RAM.

**Login File**

This file will store the key information that is needed to give members access to their account. It is not integrated in the member file as when the user first registers they will have the option to update all of their details in later however the login file will rarely change.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Size | Data type | Description | Validation (Y/N) | PK or FK |
| Customer ID | Auto | Integer | Uniquely identifies each customer | Y | PK |
| Customer Password | Auto | String | It is necessary that the user’s password is stored in the database for when they login it can be checked with their username and will grant/deny them access respectively | Y | FK |

**Customer File**

This file will be used to store all of the customer’s details in a database. I will use MS Access to link the data that is entered through the program with the database. Each individual customer of Mongoose Games will be recorded in the file. The table below shows the file structure that will be used whilst creating the solution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field size | Data type | Description | Validation (Y/N) | PK or FK |
| Customer Name | Auto | String | The user’s name is essential to connect the person with their account | N | FK |
| Address | Auto | String | Enables staff members to contact the customer if need be | N | FK |
| County | Auto | String | Forms part of the address | N | FK |
| Email Address | Auto | String | Allows a member of staff to send a direct instant message to a customer regarding new games / updates to the system | Y | FK |
| Phone Number | 11 | Integer | Allows the member to be contacted | Y | FK |

**Game File**

This table will store all of the game information regarding each game. This data is needed as all of the games need to be recorded in the system in order for them to be monitored whilst selling games out to members or buying games from them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | GameID | Price | Genre | Developer |
| Call of Duty Modern Warfare 2 | 1 | £2.99 | Action | Infinity Ward |
| OSU! | 2 | £6.99 | Rhythm | Peppy |
| Mount and Blade: Warband | 3 | £12.99 | Strategy | TaleWorlds Entertainment |
| Trine 2 | 4 | £6.99 | Puzzle | Frozenbyte |
| Portal 2 | 5 | £14.99 | Puzzle | Valve |
| Half-Life | 6 | £9.99 | Action | Valve |
| Ricochet | 7 | £1.99 | Action | Valve |
| Fifa 20 | 8 | £49.99 | Sports | EA |
| Tom Clancy’s Rainbow Six Siege | 9 | £7.99 | Action | Ubisoft |
| Far Cry | 10 | £10.99 | Adventure | Ubisoft |
| Continued... |  |  |  |  |

**Validation Table:**

The validation table will show the different types of validation checks that will be used throughout the system in order to ensure that only valid entries can be submitted into the database and therefore prevent them from getting entered into the database and disrupting it’s functionality.

|  |  |
| --- | --- |
| Name of Validation Check | Example |
| Length | When validating the phone number in the Customer file the length check will ensure it’s 11 digits. This is done as all phone numbers are 11 digits so by doing this check it will ensure only valid numbers are entered |
| Type | The type check can be used to validate the GameID from the Game file as only integers are needed therefore only integers should be expected. |
| Range | When a customer’s name is entered it’s important that the customer’s name is between 0 and 1 characters. This prevents members from entering their full name. |
| Format | When the price is entered into the Game file a format check will ensure it’s in the format of £XX.XX. This will help consistency in the member file database. |

**Method of file access**

The method of file access that will be used in the proposed solution is sequential. This method works by starting at the beginning of the file, and then locates the record by checking each record one after the other in order. I have decided to use this method due to its high speed of access as well as its ease of implementation into the program, as all that is needed is a counter that increments by 1 each time it finds data in the database, which is a very convenient combination.

However, what must be taken into account is that there  needs to be data in every field of the database or there will be issues with searching for data as it might stop looking for a record once it reaches a blank space, thinking it’s at the end. To counter this I will need to ensure that I design the database in a manner that there are no blank spaces. If a file is needed to be added to the database then it would have to be added at the end. Moreover, random access is not possible when using sequential file access and therefore all records have to be searched before the correct one is found.

**Array Data Structure**

The proposed solution will employ array data structures. The structure below demonstrates an array structure for holding member data read from a file stored on disk.

For example:

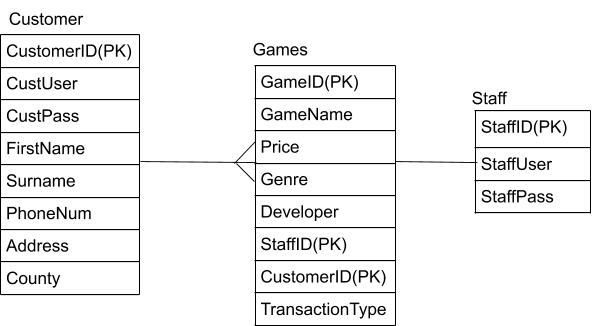
Customer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| Customer ID | CustomerName | Address | County | Email | Phone number |

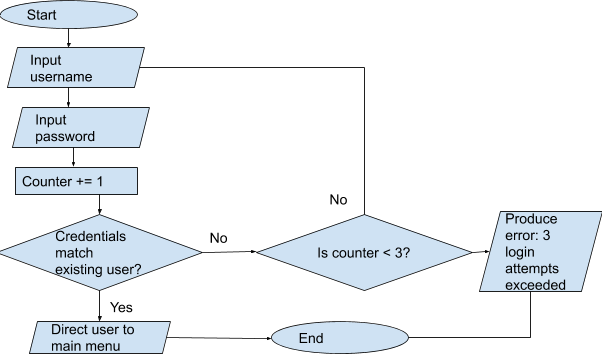
1. 42
2. William
3. 100 Tudor Rose
4. Greater London
5. [William21@gmail.com](mailto:William21@gmail.com)
6. 07827384332

Entity Relationship Diagram of the proposed database

This is an entity relationship diagram (ERD) for my main tables in the database that the system will be connecting to. This shows the relationship between how my tables are related via common keys.

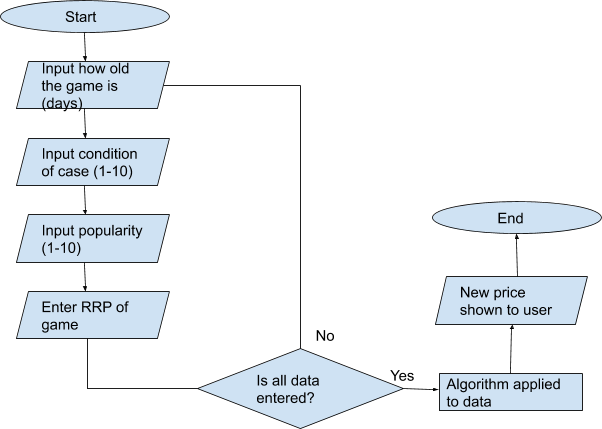


Login screen flow chart



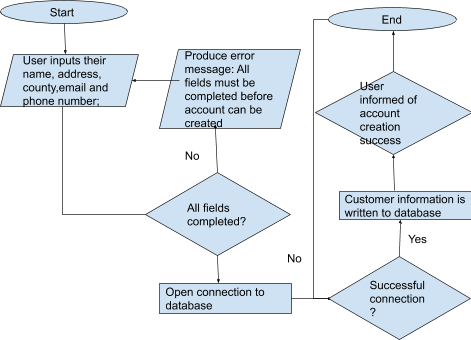
This flow chart represents the generic login functionality for when both staff members and customers request access to the system.

Flowchart for buying price calculator



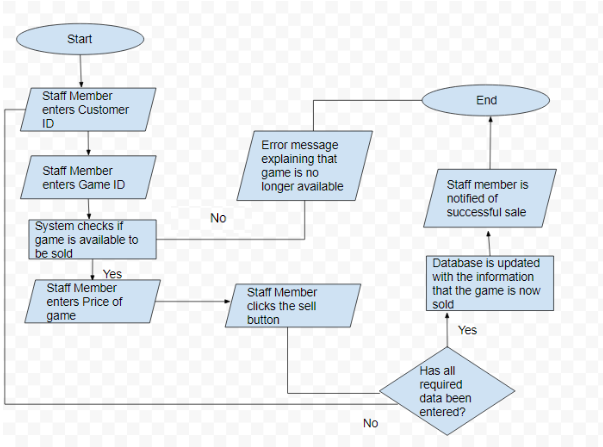
This flowchart represents how the buying prices for games brought in by customers are calculated, taking into account a few important factors in order to give an accurate and business orientated price.

Flowchart for adding new customer to database



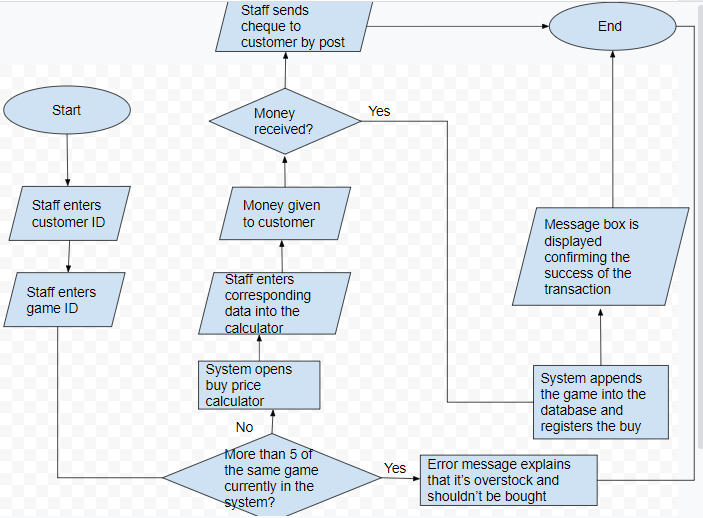
This flowchart represents the process of adding a new customer into the database, ensuring all fields are entered and a connection with the database is established.

Flowchart for selling game to customer



This flowchart represents how the system handles a sale of a game to a customer, with a detailed consideration of all possible paths of execution.

Flow chart for buying game from customer



This flowchart represents the process of when a game is being bought from a customer, taking precautions such as if its already overstocked to ensure a sensible business decision is made.