                          Mongoose Games Computing NEA

Discussion

Introduction

I have been hired by Mongoose Games to design and develop a system with a core functionality of buying and selling games to an array of customers. More importantly I am required to test and evaluate the finished system to establish if it conforms to the prescribed user requirement. The client currently employs 3 members of staff and commands a portfolio of 250 different games between different consoles. The client has an average of 500 customers. As customer numbers are increasing, the current system that has been running since the store was first opened (15 years ago) no longer fit its purpose with the size of the company. Currently most calculations are done on paper such as coming up with buying and reselling prices which is particularly time consuming and as there are shift changes, different employees have different opinions on the suggested prices which can lead to loss of money detrimental to the profitability of the business. The new system the client requires will be able to calculate all these prices by using complex algorithms which take various factors into account in order to ensure profit and eliminate human error. This extra money will help the company increase the number of games in order to appeal to a wider range of people and in turn increasing business.

Aims of the project

This section of the project will describe the broad aims of the computer system that I propose to design and develop for Mongoose Games. Bullet listed below are the aims of the proposed project.

The proposed solution should:

* Contain a preset algorithm implemented into the system which is used when coming up with selling prices for certain games.
* Contain a preset algorithm implemented into the system which is used when coming up with buying prices for certain games brought in by customers.
* Allow authorised members of staff to login using their preset logins
* Allow users to login to their existing account or create a new account to allow them to access the features that a customer account offers such buying or selling games.
* Allow users to be able to change their credentials by entering whether they want to change their username or password, and then the data they wish to change it to.
* Allow users to view all of the information that the system has stored about them
* Allow employees to sell games to customers by entering the game ID, customer ID and Price which will update the system.
* Allow employees to buy games from customers by entering the game ID, customer ID and Price which will update the system
* Allow employees to add games that have been bought from customers into the system by entering their customer ID, game ID and Price.
* Allow employees to enter new games into the database.
* Allow games to be deleted from the database once they are sold.
* Allow games to be deleted for other valid reasons (e.g. lost or stolen).
* Allow the viewing of games that have previously been bought by customers.
* Allow the viewing of games that have previously been bought from customers.
* Allow employees to check the quantity of each game in store
* Allow the front covers of each game to be shown when being viewed by anyone.
* Allow employees to view and update customer information.
* Allow anyone to view all the games currently in stock.
* Allow anyone to search through the games currently in stock.
* Allow anyone to order all available games by a certain field (e.g. Price, Release date, Genre, Publisher)
* Allow employees to check game information (e.g. Release date, Publisher, Number in franchise, Genre).
* Allow customers to reserve games from home to later pick up in store.
* Allow customers to receive a quote for their game.
* Allow games that are bought from customers to be put up as available to be bought in the system instantly but for a higher price (determined by a set algorithm).
* Develop the program so that customers can access the system from home via an app to set up a sale for a game they have or buy games for them to collect from the store.

When the new system is created it will help the company to save a lot of time with monotonous tasks such as manually inventing selling and buying prices. Overall the company will be more efficient with the new system.

Limitations

* When employees are shown games that the customer wants to sell or if they are selling a game to a customer, they will only have the game title and ID on their monitor. It would be helpful if they would see the cover of each game to confirm if it’s the right one. Currently the system will not have this feature and will have to be added in the new system.
* It could prove difficult to be able to develop an app that will allow customers to be able to connect to the system at home. This is due to the fact that there are two databases which will be needed which connect with each other. Moreover, the application would have to be able to run on multiple devices and so it would have to be compatible with them. It may be hard to achieve within the time frame given.
* When games are bought from customers the amount they should be given in pounds will be determined by taking the release date, popularity and damage to the case into account in the algorithm that determines this. A similar algorithm then needs to be applied to the price it was bought from the customer to determine how much it should be increased by before being listed for sale on the system to ensure a realistic price while maintaining profit. Currently the system does not have these algorithms and they will have to be added.

Possible limitations of a solution to the problem

This section will describe the potential limitations I proposed to solve Mongoose’s Games current problems.

* When trying to make both a system for the store and an app for the customers at home and on the go, the problem is that the app and the main system will both have to be programmed in VB.NET 2010 for them to be able to communicate correctly. They will also both need to be linked together securely over the internet in order to function properly. This could be time consuming and security is also a required feature so the user’s devices cannot be hacked into.
* The system would need to be compatible with a range of devices and operating systems to appeal to all customers and to function properly. As I will use visual studio it would be challenging to install this software on some mobile devices due to compatibility issues.
* Storing pictures of all the games on the database may not be feasible as it is a lengthy process. The storage space to store high quality images also needs to be taken into consideration as it will be high which could in turn affect the speed of the program.
* The algorithms that will be used to calculate prices for games need to be very accurate, considering a wide range of factors to ensure that reliable prices are given.

Feedback

The feedback that I have received from the client was that the system idea was excellent and the possibility of adding an application for users to log in at home although being deemed ambitious given the time constraints to deliver the project, it’s an exceptional idea. The client has also mentioned that it would be essential if I would be able to come up with accurate and appropriate algorithms to come up with prices to offer customers for their games as well as resell prices to reduce the need of human opinion to come up with the prices which has proven to be inaccurate and detrimental to the profitability of the business. I had identified this issue within my limitations section and it will be further explored.

Investigation Section

**Introduction**

In this section I will describe all of the methods that I will use to understand how the current system works with a view to design and create a system that delivers the prescribed user requirements, this will be done thoroughly in order to gather more insight on how it works in more detail and if there is anything else that may need to be implemented to the proposed system in order to provide a system that meets the client’s requirements. During my investigation I will use multiple data gathering techniques such as questionnaires, observation and interviews. I have also briefly used document analysis which required me to physically examine existing documents. Furthermore I will also examine how the current system processes data.

Described below are the methods that I will consider using in investigating the current system.

Observation

Observation is a data gathering technique which involves inspecting staff members using the current system to be able to gain an insight as to what areas need to be improved or not, it  is a very useful technique to help gather data about how the current system operates. Moreover, in the process, it will also show if there are any recurring issues that need to be investigated further.

The interviewing and questionnaires play a vibrant role in making valuable feedback available from the user based on the questions asked of them. However, there are times when these techniques may be limited in gathering useful observable scenarios such as observing a user interacting with the current system. Hence the observation technique is better suited in gathering observable features which are useful in helping me develop a full grasp of how the current system works and how the surroundings affect the user interaction with the current system.

On the contrary, observation is a very lengthy process which in turn delays the data collection process and there is also a high chance that the store might operate differently knowing that their every move will be registered which could cause some issues to not be brought to my attention.

**How I used this method:** When observing the store, it was clear that the notepads that were being used were sometimes inaccurate and at times there was confusion by members of staff as to which notepad they should write in. I have taken note of this while observing and will be designing the new system to ensure that this issue will not persist.

Questionnaires:

This method allows me to collect information from many people in a relatively short amount of time. As the store has around 500 customers this would be a lot of information to process if everybody filled out and responded to the questionnaire. Moreover, questionnaires can only provide a limited amount of information, especially when the questionnaire is comprised of close ended questions e.g. *“Do you like the current system”* which could only produce a yes or no answer whereas questions such as *“What current features of the current system do you enjoy”* which would allow for a possible range of answers. Questionnaires can be answered by customers at a time that is convenient to them which will make them more likely not to rush it as they won’t feel pressured for time due to having other commitments. In addition, customers are more likely to give an honest opinion due to their anonymity. However, due to the inability to ask customers to elaborate on their answers, and their ability to skip questions, despite the sample size, detailed responses won’t be achieved.

**How I used this method:** I created a questionnaire and gave it out to Mongoose Games customers who were buying/selling a game(s). This has allowed me to gather an array of customer’s views about the store and how it currently works. This information will be helpful to me as it can be used when thinking of designs and features for the system.

Interviews:

Unlike questionnaires, interviews involve having a formal meeting with a customer in order to ask them a range of questions in search for detailed responses. Furthermore, during interviews it will be possible to ask different questions depending on what the customer answers to the previous ones, to be able to get an even deeper insight into their opinion on the system. However, the weaknesses of interviews are that it’s a very time consuming process and so therefore only a select few of customers can be interviewed. Moreover, as interviews are formal meetings, time, location and date will have to be arranged prior to the interview which suits both the interviewer and the interviewee. This further prolongs the process of the data collection. Unlike questionnaires, there is no anonymity and so this could lead to the interviewee not giving their honest opinion on the current system and only saying what they think the interviewer wants to hear, which will lead to biased results. However, if the interviewee maintains sincerity, then the results will be high quality and new recommendations to the system may also be mentioned.

**How I used this method:** I conducted an interview with my client (as although is not the only stakeholder, has the highest interest in the success of the proposed system) to ask them certain questions and also to hear feedback about what they would like to be included in the project. Being able to speak formally with the client was useful as it gave me a first person insight into how the client feels about the way the store is currently run. From this I could foresee system features that may make the store more efficient and give a better service for the customers.

Document analysis

Document sampling will show me how a single document can represent the whole population of data in the system. Due to the lack of time available to examine all documents it’s more realistic to pick out one document from a sampled population to represent the whole system. For example, from a pile of 250 documents only one or 2 would be picked for convenience. The advantages are that the documents can be examined in detail within a short time frame and this will give enough of an indication as to how the current system functions and what can be done to improve efficiency.

However, the problem with document analysis is that it doesn’t show how the data is used by the members of staff either looking to find or input a record. Moreover, not all areas of the documents can be examined. For example, if staff members make hidden notes to help them process data which if they would not be able to be retrieved for the document analysis, it would result in a less accurate indication as to how the current system processes data.

**How I used this method:** I looked through all the documents that the store has been using to store and process all of their current games and customer information (included on the notepads). From looking at the documents it allowed me to further understand the way in which the current system is working and also ideas on how to make the system more secure and speed up the process of selling and buying games.

Given the analysis of the pros and cons of the data collection techniques discussed in this section I have decided that I will continue with the following techniques; Observation, interviews and questionnaires to examine in detail how the current system works with a view to identify areas of improvement. In my opinion, these three techniques will yield sufficiently reliable and accurate data to enable me to see all of the current faults, inefficiencies and limiations of the system, some of which will have not been aware to me, and to be able to witness first-hand the operations of the system.

Investigation of the Current System

Observation:

The current system is paper based and uses notepads for the staff to note down when people have bought and sold games.

However, this system could be seen as unreliable as certain staff members could have hard to understand handwriting causes problems when another member of staff wants to update a record as they might not know what the previous staff member wrote. This issue is also present when staff members come up with a price to buy a game from a customer for, they often check back in the notes of when the same or similar game was bought to be able to get an idea of how much they should offer the customer yet if they won’t be able to know the price from the previous game sold, they will have to go off their own opinion. Moreover, another issue with a paper based system is that the current system takes up a lot of space over time when the notepads become full and with hundreds of customer numbers this could prove to be a large issue. Accessing records from a while back is also a very time consuming process as a member of staff would have to look back note by note to find, input or read data. This is very harmful to the business as customers could be put off from coming to the store due to long waiting times, especially if they’re in a rush.

**Questionnaire that was issued to customers as they were waiting to buy/sell a game:**

Question 1: Would you like to be able to view all available games from home?

**Answer 1:** I guess it would be a good idea, however for me I don’t really mind to pop in to question a game’s availability as I live 5 minutes away.

**Answer 2:** Yes

**Answer 3:** It would be good if I were able to receive alerts when games are put in stock so I don’t miss out on new releases or cheap deals for older games.

**Answer 4:** Yes as it would save me the disappointment of visiting the store to then find out the game I want isn’t in stock.

**Answer 5:** Yes I think this feature would be very helpful as it would save me a trip to the store.

**Overall response:** In conclusion, it has been seen in good light the possibility of carrying out this function from home. Moreover, an idea has been brought to my attention that I had previously not thought about, the idea of issuing alerts to customers when a game is introduced into the database. I will consider these ideas when designing the system.

Question 2: Are there any areas you think that Mongoose Games could improve on?

**Answer 1:** The time it takes for members of staff to process games is very time consuming and I usually have to wait at least 5 minutes to buy/sell a game.

**Answer 2:** The possibility of ordering the next game in the series if I enjoyed the game should be added (if there is a sequel of course). For example, if I enjoyed the first Call Of Duty game, I could pre-order the next game from home. This would be convenient for me and save me a trip to the store.

**Answer 3:** I would like to be able to send a pre order request for games that aren’t yet available in store so that the game can be bought by Mongoose Games and delivered to the store. Then I could be contacted saying that the game is ready for collection by email.

**Answer 4:** I think that the consistency of the business should be improved. For example, my friend sold his game to Mongoose Games for £25 and told me about it, as I owned the same game I came into store to sell it and I was only offered £17 for it. This could be due to the fact that the quotes offered to the customers are just made up by the staff members which I believe to be unacceptable and extremely unprofessional.

**Answer 5:** Speed of service, especially when selling games to the store.

**Overall response:** A few features identified that I will consider when designing the system some of which I had already considered however answer 3 could be quite farfetched to implement as complex validation will have to be utilised.

Question 3 - Would you like to be able to receive a quote for a game(s) you wish to sell from home?

**Answer 1:** Yes, I would as although I normally call them to receive a quote, the amount I get offered for the same game varies due to different staff members picking up the phone at different times which all have distinct opinions on the value of the game which makes this method unreliable.

**Answer 2:** I am always using my computer for work purposes and so having this feature will allow me to quickly get a quote on my game and then continue working.

**Answer 3:** Yes I would very much enjoy this feature as it’s very often that I come to the store and get a disappointing quote for a game and then have to go home empty handed as well as having wasted a considerable amount of time.

**Answer 4:**  I usually call Mongoose Games in order to receive a quote from the staff member that picks up the phone which is a generally fast process.

**Answer 5:** Yes this will be a good feature as I would not have to visit the store in order to ask how much they would give me for a game.

**Overall response:** The majority of the customers questioned said that they liked the ability of being able to receive a quote of a game from home and would be used often to allow the customer to decide if they want to come into the store to sell it.

Question 4 - On a scale of 1 to 10, how satisfied are you with the efficiency of the current system?

**Answer 1:** 3

**Answer 2:** 6

**Answer 3:** 5

**Answer 4:** 2

**Answer 5:** 3

**Overall response:** The overall response was negative which implies the speed of service is very slow, this is definitely due to the fact that the system is paper based which increases the waiting times for the customers, negatively affecting their satisfaction.

Question 5 - Would you like to be able to reserve a game to then later collect in store?

**Answer 1:** Yes I would like this feature as it would allow me to take my time to get a game by being able to pop in another day to calmly collect it without having to rush to the store to buy it before it goes.

**Answer 2:** No, I think it wouldn’t be necessary, it’s a lot more simple to just go into the store and buy the game, I think that the whole process is just time consuming.

**Answer 3:** Yes I would very much enjoy this feature as I get paid monthly and if a new game gets released near the end of the month and I don’t have the funds to buy it, this will allow me to reserve it for when after I get paid.

**Answer 4:** Yes I would appreciate this feature as when new games come out, the store is usually packed with people wanting to buy it but this will allow me to calmly reserve the game from home to come in to collect it.

**Answer 5:** Yes, and I think that it will tie in very well with the proposed feature of being able to view all available games, it’ll make for an excellent system!

**Overall response:** Many customers liked the idea of being able to reserve a game from home due to the convenience it would bring. I have gathered that this is a feature that would be desired by many of the customers.

Interview:

I decided to carry out an interview with my client to help me further understand the current system. I had to create an appointment with the client at a date and time that suited us and on said day I attended the store in order to interview him. I made sure that it was very detailed to ensure that I wouldn’t miss out any possible system requirements and also for the client to provide me with any others.

I started the interview by asking: Apart from the issues that I have identified and decided to solve in the proposed solution, do you think there are any more that need to be addressed? The response I received from the client was that there was nothing that I have missed out from what he’s previously mentioned and that he’s looking forward to the new system. However, he did comment that I should ensure that the proposed system will be user-friendly as it will be a big shock for the staff and certain customers to suddenly switch to a computer based system after more than a decade using a paper based one. We later discussed how the proposed system should look like and the different user interfaces I will have to design. The presence of a registration screen for new customers, login screen for staff and customers as well as how the actual system will look like were all identified. Furthermore, we talked about what information will be inputted and outputted from the proposed system which included: customer name, customer ID, game title, publisher, bought price and sold price. This is the data that will be entered into the system by the staff members when games are being sold or bought.

Another important topic that was discussed was security. The client expressed how he would like each member of staff to have preset logins to ensure that their credentials are sophisticated and secure. He also commented that as customers will be creating their own credentials, that validation should be implemented to ensure that their passwords will be advanced and less prone to brute force attacks. These features will ensure that high security standards are set for the system which is important as the system will be storing personal information of the customers such as address and name and therefore the store should comply with the Data Protection Act 1998 to prevent legal action being taken in the case that they were to lose customer information if the system would get leaked.

This implementation that the client has outlined and that the current system is lacking which will be changed in the proposed system and will help the client keep their customer’s information secure.

Stakeholders:

The owner of Mongoose Games is called Mongoose Hanks (not a real name) and he is responsible for managing the members of staff within the store. Hanks’ priority is to increase the efficiency of the store which will give the customers a better service which will lead to an increased popularity of the store and will attract more customers.

The second most important stakeholder is the members of staff. The proposed system will allow them to quickly process game transactions which is currently time consuming and hand-operated. Staff members will be able to type the record regarding each transaction into the system instead of having to write down each game that has been sold/bought onto paper. The new system will facilitate their jobs which will in turn increase productivity. The time saved can then be spent completing other tasks to improve the business.

Lastly, the customers also have a great interest in the store as it’s a place they can use to get money for their unwanted games or to buy new ones.

The new system will benefit impatient/busy customers that normally have to wait a while for the members of staff to find their records due to the quicker and more secure method that will be used that processes games when they are being bought or sold.

Existing solutions to similar problems:

My local newsagents has recently changed from a paper-based system to maintain stock held in the store to a computer-based system. Before the change, the store owner had to physically write each item of stock down on a piece of paper to acknowledge that it had been received. Now this process is much simpler as it allows the shop owner to scan the barcode on a product which automatically counts how many have been received into the shop. This saves a lot of time and money less time is wasted performing this daily task, which in turn increases the efficiency of the business.

The system can also show the owner if there are any food products in the store that are past their sell by date as well as if a new edition in a magazine comes out. When stock arrives it gets scanned into the system and as it scans it prompts the store owner to enter the sell by date of the food product/edition number of a magazine. When the sell by date is nearing, it prompts the store owner and the price of said item is then reduced subsequently and when an edition of a magazine gets surpassed, they are also notified and in turn, the new edition is automatically ordered. Finally, when a food product reaches its sell by date, it’s then thrown out.

Similarly, in my proposed system I will implement a similar way of entering customer information so that it won’t have to be done by hand and instead be typed in. This method has shown to increase efficiency and convenience as well as accuracy and thus will be used in the system.

This could also be developed on in the future so that the system would be connected to a scanning device which would scan the barcodes of games similar to how the shop owner scans their products which would be a much more rapid method than entering the game ID manually.

**Existing solution to similar problem**

My local library has also changed from a paper-based book control system to a computer based system. Before the change, the library staff had to manually write the

Data collected by the current system:

Data in the current system:

Shown below is an example of an entry that may be entered by a member of staff

It would be written onto paper to then be put aside until the record needs to be looked at again.

Below is shown that the Game ID 73 has been bought from William Mireles for £12. The price would have been devised by the employee.

**Member name:** William Mireles **Customer ID:** 1337 **Game ID:** 35 **TransactionType:** Bought **Price:** £22

This would then be written on a page of a notepad and it would then be put in a pile of similar records.

Outputs from the current system:

When a record has to be retrieved, a member of staff would then have to look back into the notepad and find the correct record. This would be a significantly time-consuming process and the customer would have to wait in the meantime. This is very poor customer service as if a possible customer would be in a rush, they would be deterred from going to the store due to the assumed delay the transaction would cause them. Moreover, this loses potential business which is a serious issue as it affects the success of the store.

After a different member of staff has accessed it could also be retrieved incorrectly e.g.

**Member name:** William Mireles **Customer ID:** 1337 **Game ID:** 36 **TransactionType:** Sold

**Price:** £8

In this case, the number of the Game ID has been read incorrectly by the other member of staff and the wrong game would have been marked as sold to a customer. This would cause problems for the store to keep track of sold games.

Another problem with retrieval of data from the current system is that for example if an employee has to devise a price for a game brought in by a customer, he would try to go off the price a similar/same game was sold for. If the last time the same/similar game was sold 4 months ago, it would be a very complicated process in order to retrieve the record to see the previous price. This is caused because the store would have moved onto multiple new notepads within the time causing the original one to be lost in a pile.

Currently in the system to check what games are overstocked and therefore shouldn’t be bought from customers is a prolonged process due to the fact that the staff member would have to search the notepads with data stored in to take down a tally of the quantity of each game.

In order to give a customer a quote for their game, if the same/similar game would have not been sold previously, the employee would have to devise the price from their own opinion. This method can be inaccurate and could result in the customer being given more or less than what they should receive which affects business. The new system will have a built in algorithm which enables the employee to reliably and efficiently calculate the suggested price.

**Receipt given to customer when they buy/sell game**

Limitations of the current system:

A main limitation of the current system is that the store relies on the employees to devise sensible quotes for games brought in by customers. There is no current system in place with a functionality of calculating this for them. This costs the store a lot of money each year as with hundreds of customers there is bound to be people that will take advantage of the lack of this function and try to lie to or persuade the employee to get a better deal for their game.

Purpose of the Project:

The main purpose of this project is to create a system for Mongoose Games which will help them become more organised as well as more efficient. The system will help sort the day to day time consuming processes of using notepads in order to store customer information and game details.

The new system will be entirely electronic and there will no longer be a need to store a pile of notepads which is a security risk as it’s very insecure and could lead to a breach in the Data Protection Act 1998 if customer details are leaked in any way.

Human error will also be reduced by using the new system as it will no longer be possible for a staff member to make a mistake due to bad handwriting and this will keep the business’ database accurate. It will also be further reduced via the implementation of the algorithms which will calculate responsible prices for games that are bought from customers. The business would benefit from this as profitability will be increased which will help them buy new games etc.

Objectives of the project:

* Firstly, functionality to login with a username and password will be needed to be able to grant access to the system’s functionalities, there will also be a different login for customers and staff members to ensure that at the same time customers can make use of the system without having administrative features such as adding records and maintaining data.
* The new system will be able to solve the current issue of the time it takes to do basic tasks such as buying games from customers then adding them to the system and selling games to them. It will help to reduce the time the customer has to wait and it will help to provide improved customer service.
* When the system is fully implemented it will enable the staff members to check the quantity of each game in store which will help them to ensure that they don’t accept any more of the same game from customers once it’s overstocked enabling the supply and demand to be maintained.
* The proposed system also creates many more features that the store could not currently do. For example, a member of staff could not easily contact a customer to inform them of a new game released because they would have to look through a large amount of notepads in order to find that particular customer’s details. With the new system, this process can be carried out by typing in the customer ID number to then find all of their respective details.
* The proposed system will also make it easier to maintain data e.g. update prices by just searching the game and editing its information instead of having to go through the notepads to find that game’s details
* Moreover, deleting unwanted/obsolete data will also be facilitated as instead of having to search through the notepads and throwing out unneeded data, records can be searched and subsequent data can be deleted.
* The proposed system should also have intuitively simple user-friendly interfaces to ensure that the system can be navigated by all kinds of people with different levels of computer literacy

Success Criteria:

In this section I will be describing the success criteria that I will use to evaluate the proposed system. The criteria that I intend to employ to evaluate the finished system include:

Performance, suitability, usability, reliability and robustness.

**Performance:**

I plan to test the system’s performance by entering test data into the system to check if it can handle even the most complicated data. This process will come under the testing and evaluation section of the project and will test the functionality of the system to determine whether it works well before being implemented on the client’s computers.

**Suitability:**

When the system has been successfully designed and created it will be tested to check that it’s definitely suitable for the client’s store and its customers. I will compare the system to the aims and objectives that Mongoose Games have asked me to meet. If all of the aims and objectives have been met by the proposed system then this will confirm that the system is fully suitable for the store’s needs and ensure that they will be satisfied with their new system.

**Usability:**

It’s important to ensure that the new system will be easy for members of staff as well as the customers to get used to. If the system is laid out well and easy to navigate then this will make it comfortable to use by all the people who need to do so. In order to test the usability of the system I will ask friends and family to test it and ask for their feedback on various aspects of how they felt when using the system. For example, “Were instructions on how to use the system clear?”, “Did you feel confused at all when using the system?”

It’s best to ask people who have never used the system before in order to get a clear understanding of how they feel as it would be biased if I tested the system due to the fact that I know how it will work. I will lay the questionnaire out by putting boxes with varying levels of satisfaction e.g. Strongly Disagree, Disagree, Agree, Strongly Agree. In order to make the test a success I will say that an average of 85% of reviews to Agree or strongly agree in order to make the test a success.

**Reliability:**

In order to measure how reliable the new system is that I will put it through some tests that repeat general tasks and measure to see if the results of the test are the same each time. This is an important test as the system cannot produce varying outputs at different times or days it needs to be consistent. The system should be able to work 24 hours a day 7 days a week if it was needed to. It is unlikely that this would be the case as the store is only open for 9 hours a day however the security systems need to be running constantly to keep customer information safe and secure.

For example, I will sell 20 games one after the other to test accounts in order to check if the details are correct each time and the system is updated with the new game quantities available for sale each time. This will be a routine task and if the data is updated as expected then this test will be a pass and the system will be confirmed as reliable.

**Robustness:**

Robustness is an important feature of the system as it needs to be able to cope with incorrect data being entered into it and ensure that the correct error messages are displayed and it does not crash. If a member of staff were to enter a piece of information into the system incorrectly then the system should produce an error message to let the member of staff know what they have done wrong. This is essential as if the error message did not appear then the system could be said to not be functioning correctly and this could cause disruption in the store and therefore bad customer service.

I will test the security features incorporated on the proposed system to ensure that there are no security flaws and to check that no unauthorised person can access the system. I will do this by entering incorrect username and password combinations and then checking what the system will output. If no access is granted and error messages appear explaining that either the username or password is wrong then this would be a success. I will also carry out similar tests on different parts of the system such as buying games or selling games to be able to check if the correct error messages appear and that no incorrect data is entered into the database. If both of these tests are successful and produce the correct outputs then this will be a successful test and ensure the proposed system will not crash during normal and exceptional circumstances.  
  
  
  
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

Login

Staff

Customer

Add/remove game

Search Game

View recently transferred games

Update customer details

Game price calculator

View games

Request quote for game

Buy game

Sell Game

Create account

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.

Mongoose Games Login

Please select one of the following:

Customer

Labels

Staff member

BtnStaff

BtnCust

Mongoose Games staff login

Username

Password

Labels

Login

txtUsername

txtPassword

BtnLogin

Clear

Attempts:

BtnClear

BtnAttempts

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. Once they create their account they are able to login using their respective username and password.

Customer login

Username:

Password:

Login

Create account

Labels

txtUsername

txtPassword

BtnLogin1

BtnCreaAcc

Clear

BtnClear

Attemptss:

txtAttempts

|  |  |
| --- | --- |
| 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.

Name:

Address:

County:

Telephone No.

Email:

Clear

Continue

Labels

txtName

txtAddress

txtCounty

txtTelephone

txtEmail

BtnCont

BtnClear

Customer Details

Create Account

Username:

Password:

Save and Login

Labels

BtnCredentials

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 allowed them to perform various tasks that had previously needed to be done manually on notepads.

Games:

Welcome to the Mongoose Games computer system. Please click the respective button to carry out your desired task.

Register sold game to customer

Mongoose Games system for staff

Register bought game from customer

View/edit all available games

Customer Details:

Check customer details

Update customer details

Other:

Selling price calculator

Buying price calculator

View games recently sold/bought to/from customers

Check game quantities

**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

Check customer details

Search by name:

Customer details:

Labels

txtNameSearch

LstBoxCustDetails

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

Check number of games available for sale

Check

Total available games:

Labels

BtnCheck

txtTotal

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

Check games that have been brought into/taken out the store

Please select to view the games that belong to either category

Bought from Customers:

Sold to Customers:

View

View

Labels

BtnView1

BtnView2

This screen is shown when “BtnView1” is pressed

Games bought from customers

Game details:

Labels

LstBoxGameDetails1

Display

BtnDisplay1

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

Games sold to customers

Game details:

Labels

LstBoxGameDetails2

Display

BtnDisplay2

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

View/edit all available games

Search by game:

Game details:

Labels

txtGameSearch

LstBoxCustDetails

Delete game

Add game

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.

Buying price calculator

Enter number of days since release date:

Enter condition of case (1-10):

Enter popularity: (1-10)

Enter RRP of game:

Labels

TxtDays

TxtCondition

TxtPop

TxtRRP

Calculate

Price:

£

BtnCalculate1

TxtTotalPrice

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

Resell price calculator

Enter price paid for game:

Calculate

£

Resell price:

£

Labels

TxtAmountGiven

BtnCalculate2

TxtPrice

**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.

View/edit all available games

Search by game:

Game details:

Labels

txtGameSearch

LstBoxCustDetails

Delete game

Add game

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

Register game sold to customer

Name:

Price:

Genre:

Developer:

CustomerID

Labels

txtGameName

txtPrice

txtGenre

txtDeveloper

txtCustomerID

Delete

BtnDel

GameID

TxtGameID

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

Register game bought from customer

Name:

Price:

Genre:

Developer:

CustomerID

Labels

txtGameName

txtPrice

txtGenre

txtDeveloper

txtCustomerID

Add

BtnAdd

GameID

TxtGameID

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

Name:

Address:

County:

Telephone No.

Email:

Clear

Update

Labels

txtName

txtAddress

txtCounty

txtTelephone

txtEmail

BtnUpdate

BtnClear

Update Customer Details

CustomerID:

txtID

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.

Mongoose Game system for customers

Greetings valued customer and welcome to the Mongoose Games system. Feel free to carry out your desired task by using the facilities below.

View all games currently available

Buy a game

Receive a quota for a game you wish to sell

Sell a game

BtnSell

BtnQuota

BtnViewAll

BtnBuy

**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.

Estimate quota receiver

Enter number of days since release date:

Enter condition of case (1-10):

Enter popularity: (1-10)

Enter RRP of game:

Labels

TxtDays

TxtCondition

TxtPop

TxtRRP

Calculate

Price:

£

BtnCalculate1

TxtTotalPrice

**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 |

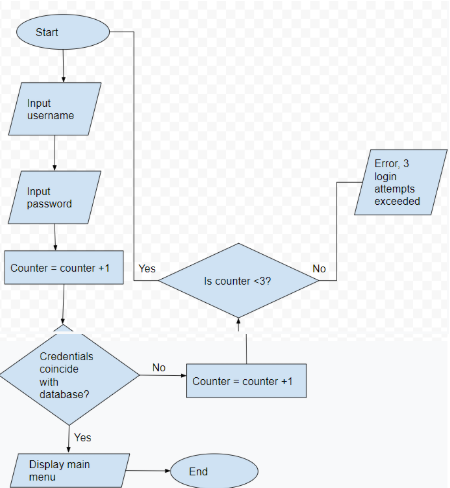
(0) 42

1. William
2. 100 Tudor Rose
3. Surrey
4. William21@gmail.com
5. 07827382771

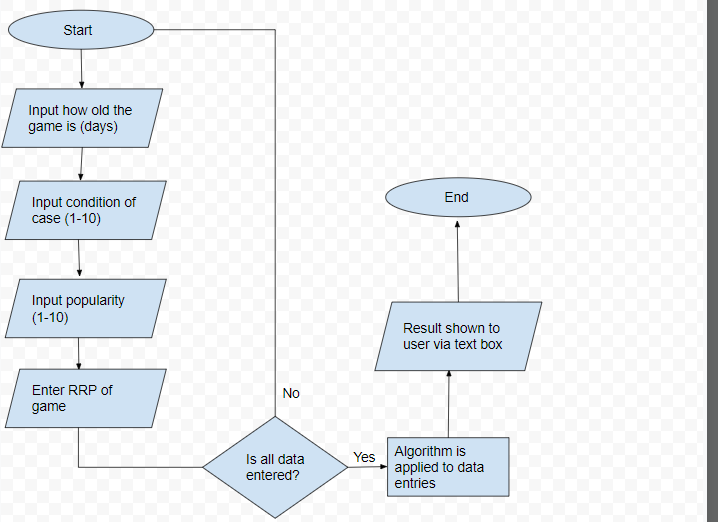
Entity Relationship Diagram of the proposed database

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.

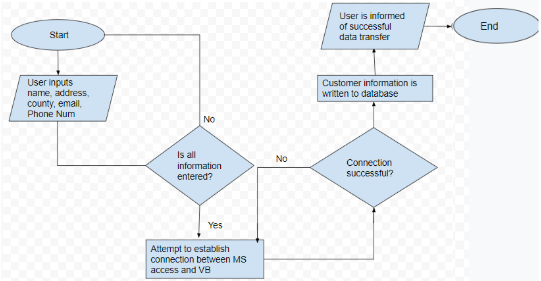
Login screen flow chart



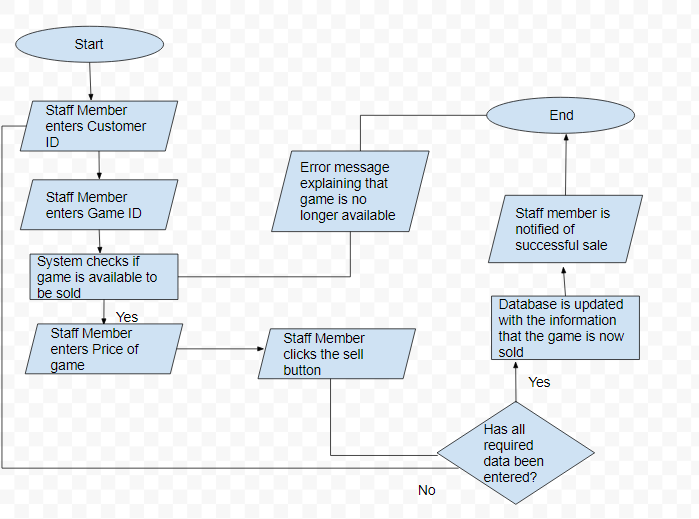
Flowchart for buying price calculator



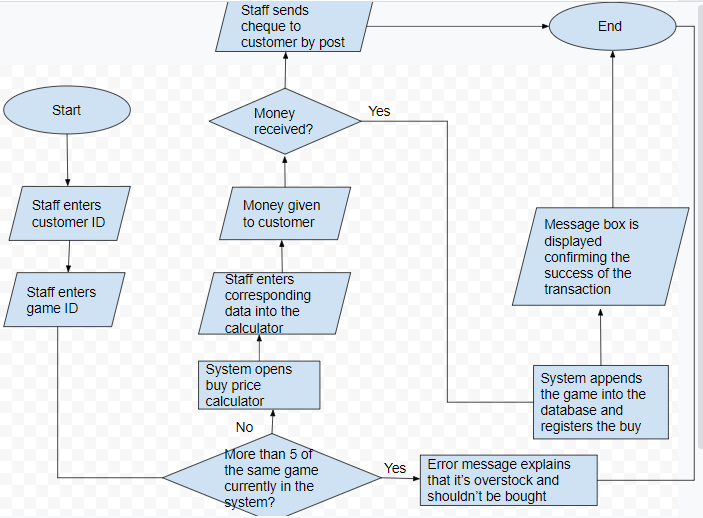
Flowchart for adding new customer to database



Flowchart for selling game to customer



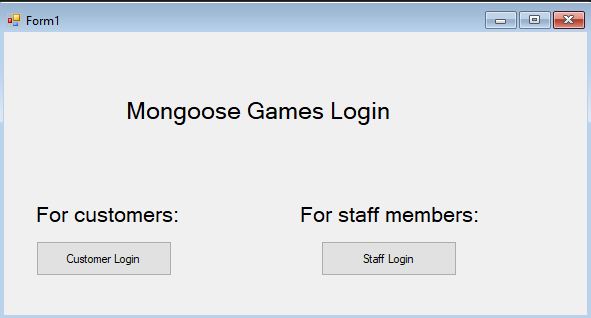
Flow chart for buying game from customer



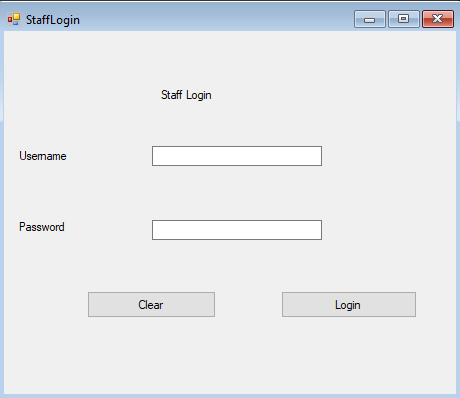
Prototype

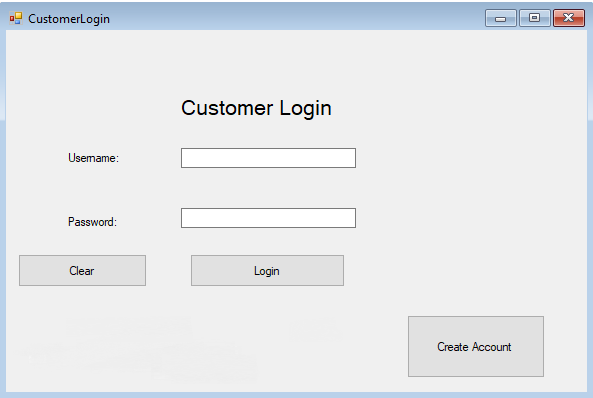
**Introduction:**

In this section of the project I will be explaining the parts of the proposed project that I will choose to prototype and the reasons for doing so. Furthermore, I will explain why I have chosen not to prototype certain areas of the project. This will then be followed by an evaluation and shortcomings and improvement section.



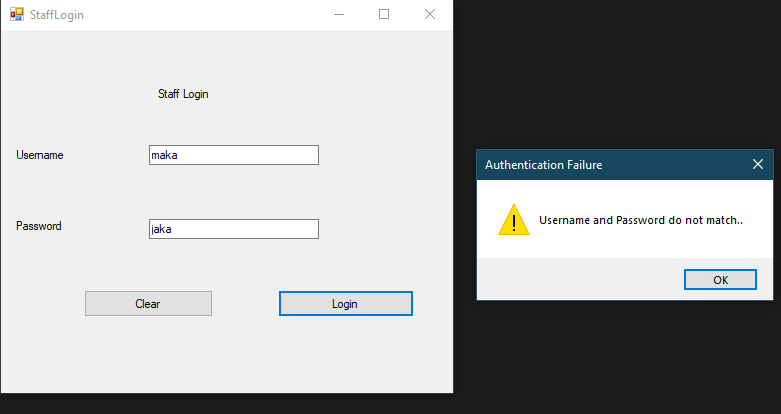
I have chosen to prototype the main login section which is the first thing any user will see when they log into the system. The login system that I have created works with a Microsoft access file where all of the user’s details are stored in. This ensures that only the people who have a prior created account stored in the database can be granted access to the system. During the creation of this prototype I was faced with various errors and issues as I had no experience with linking MS Access databases with vbnet and so I had to engulf in fairly deep research in order to achieve this, but eventually I was able to create the connection resulting in a fully functional login system for both customers and staff members.



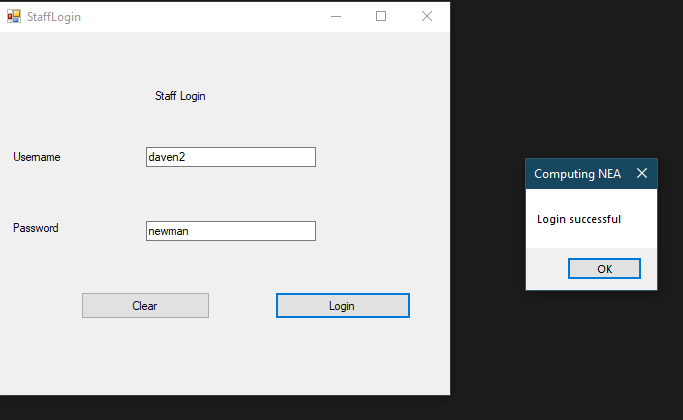


Output when invalid credentials are presented

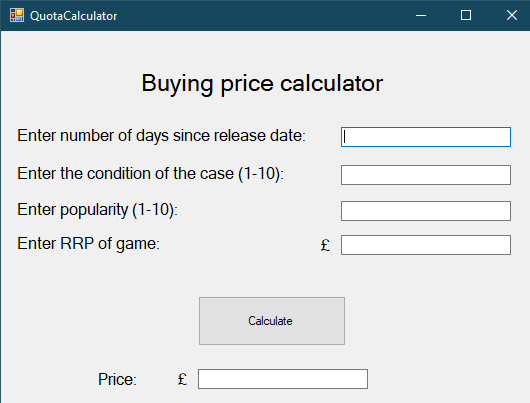
This image below demonstrates the functional login screen that I have prototyped for the user requirement specification. The prototype shows an error message dialogue box to confirm that the prototype successfully validated the user login.



Output when valid login is presented

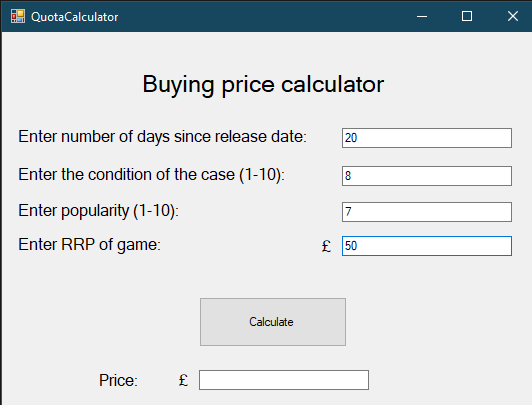


The next part of the system that I prototyped was the quote calculator. This is a facility that will be used regularly by the staff members in order to give them an indication for how much they should buy a game off a customer. This calculator makes use of a complex algorithm which accurately determines a price based on factors. The difficulties I faced while programming this part was that I found it hard to come up with an efficient and reliable algorithm that takes into account all of the factors of the game and casing that would ensure profitability and maintain the integrity of the business.



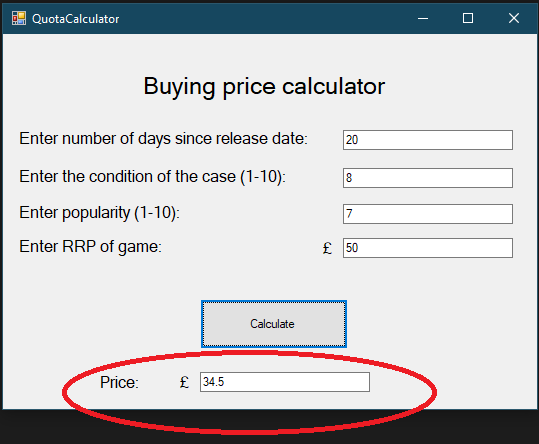
Test data entered into the quote calculator form

This shows normal, expected data that would be entered into the system about a game given in by a customer.

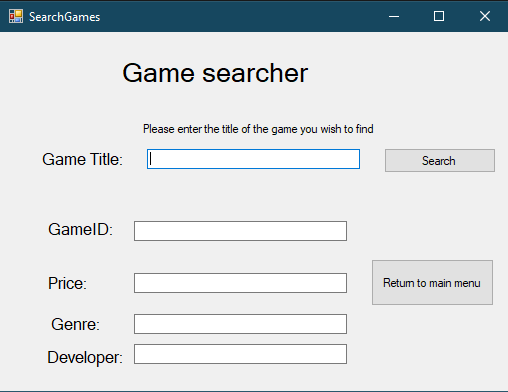


Output

This shows the result after all factors being used in the algorithm, to give the quote for the game.

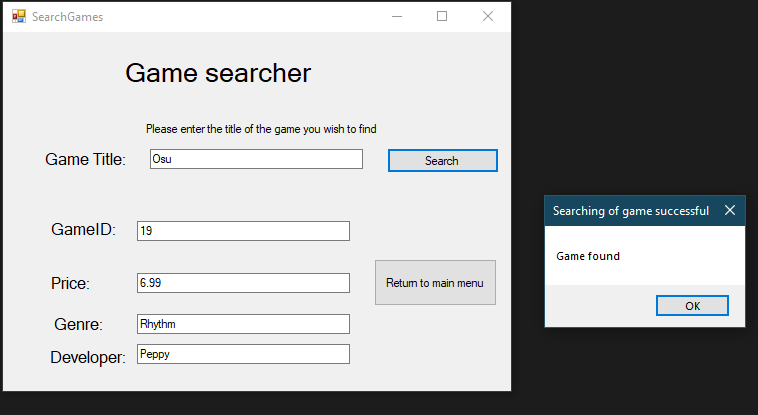


Furthermore, I have also prototyped the part of the system which allows customers to search for games by their title to see if their desired game is in stock. I was able to carry out this by again, utilising the pre-existing connection of my database with VB.NET, however, the Structured Query Language (SQL) statement works with the Table “Games” and not “Staff” or “Customers” which I used for the login section.

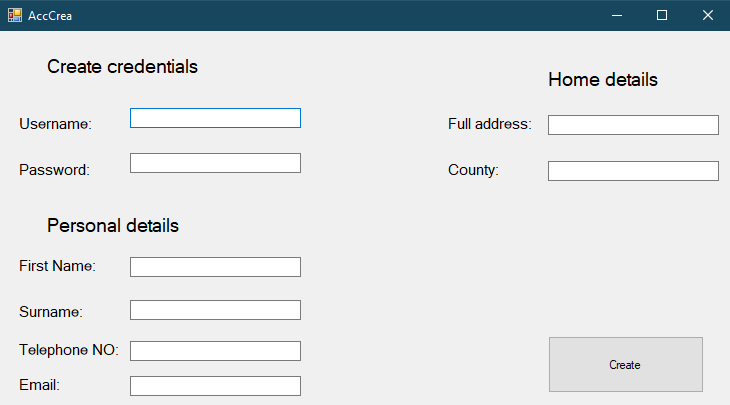


Test data entered into the form and output of the game’s relative details displayed via text boxes.

This shows test data being entered into the search bar for a game that exists in the database, upon clicking the search button, the database is searched and the data is instantly retrieved and populated into their corresponding text boxes.

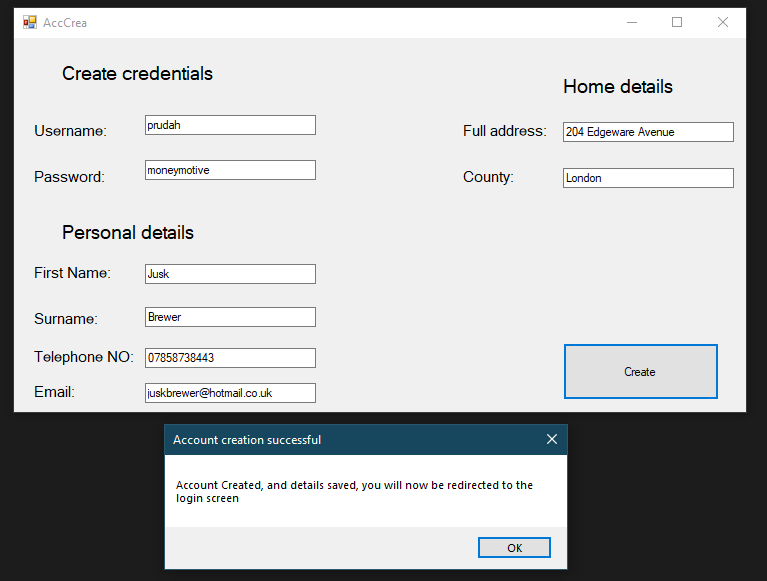


Moreover, I have gone on to prototype the “Create Account” part of the system which is used by customers who do not have an existing account on the system. This section consists of getting a full fact file of the customer including their address, personal details as well as their desired credentials that they will then use henceforth when logging in to the system.

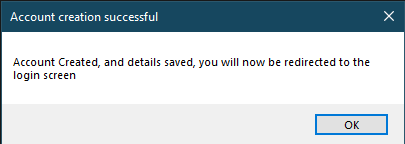


Test data entered into the form

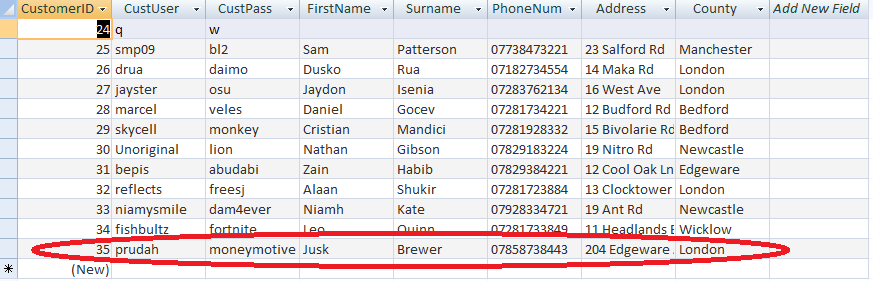
This shows a selection of data that has been entered into the system which would be done by a new customer who wishes to create an account in order to have access to the functionalities of the program. All of their inputted data is then sent to the access document in the corresponding tables, ready for future reference e.g. login



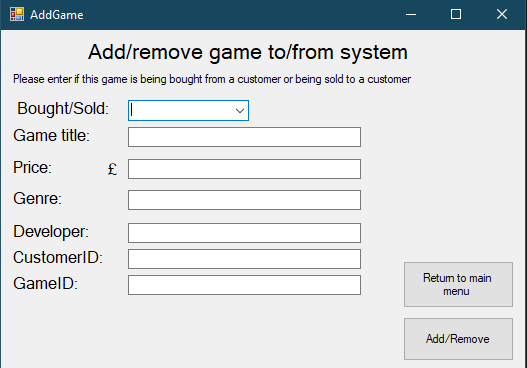
Output



Data saved in the MS Access Database

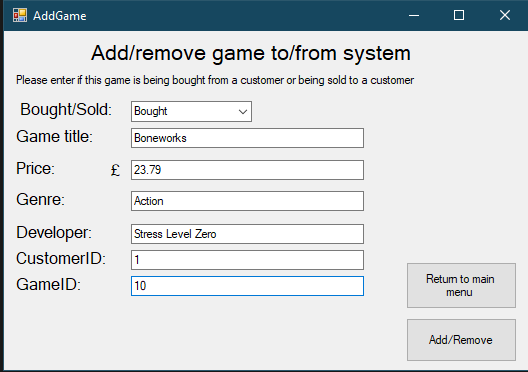


Finally, I have decided to prototype another integral part of the system which allows staff members to delete/add games from/to the database (see form below) depending on whether they have been bought from a customer (add) or sold to one (delete). The details of the game will then be subsequently added or removed from the Games table in the database and the transaction will be logged in the RecentGames table for when the staff member desires to see a report of the games that have been sold/bought recently.



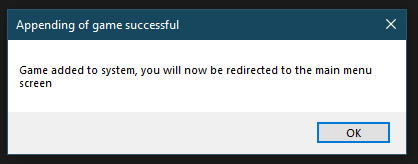
This form above will allow for a staff member to enter related data of the game, specify if the game is being sold or bought, and then which customer is involved in the transaction. These details will then be added to or removed from the database(see form below) and then the staff member gets a confirmation if they want to log the transaction into the system. It is possible that they won’t want to if they had to delete a game from a database from another reason than that of it being bought such as if it was stolen or lost. Moreover, if they want to add a game to the database that hasn’t been bought from a customer however has been brought brand new from a warehouse to then be sold in the store.

Test data entered into the form



Output

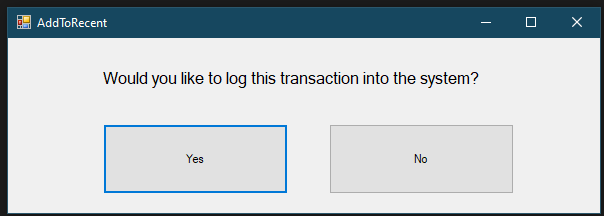
This informs the staff member in their success in adding a game to the system. They are subsequently redirected to the main menu.



Output (confirmation)

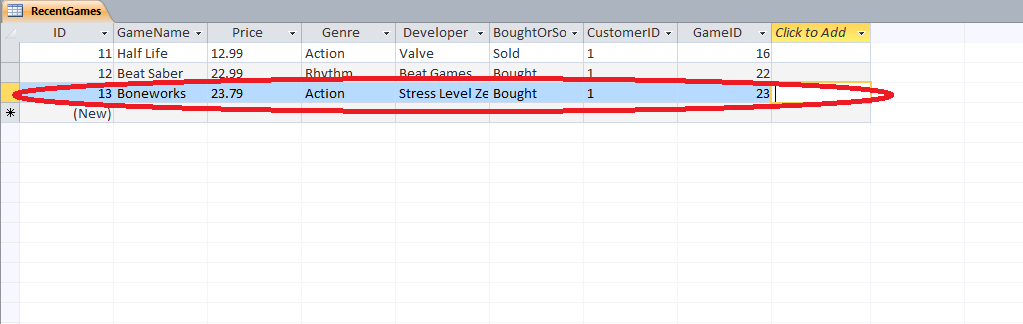
I added this confirmation for if the staff member wants to add a game to the system that has been bought brand new from a warehouse or remove a game that has been lost/stolen from the store without having to log the transaction to the “RecentGames” table which stores all of the games that have been bought/sold from/to a customer.

If the user selects the “Yes” button, then the transaction will be recorded in the “RecentGames” table (which records all of the games that have been appended/deleted from the system) as well as being added/removed from the “Games” table and if no is pressed the transaction won’t be recorded in the “RecentGames” table but it will still be added/removed from the “Games” table.



Transaction being recorded in the system

Below shows the “RecentGames” table in the database, confirming that the game has been successfully logged in the database as by the staff member’s wish, as a recently bought game.



Sections of the proposed solution I haven’t included

I haven’t developed the feature that allows staff members to view the recently bought and sold games from the store as it’s similar to the game search prototype as it reads from the database and produces a result, however in a listbox, which I am yet to find out how to do. Nevertheless, this will still be included in the development stage.

Moreover, I haven’t included the function that allows staff to update customer details in the program as that isn’t on high priority as at the moment only test customer data is in the database which won’t change. Yet, of course this will be implemented into the development stage for when this system goes live and real customers will be in the database which can have constantly changing information such as address, phone numbers and emails.

In addition, I haven’t coded the functionality which allows the customer to view all games that are currently available to be bought and also haven’t made it possible for customers to send buy request or sell requests of games, this is also due to how the system at the moment has no real customers but this will too be added in the development stage.

Evaluation:

Overall, the current prototype that I have established functions well and effectively carries out the tasks it was set out to do. It enables both customers and staff members to log into the system and carry out some of their respective tasks. For example, staff members are able to calculate the price of a game that a customer wants to sell which works hand in hand with the function that allows customers to request a quote for their game. Moreover, the Game Search feature of the project to be used by customers to allow them to search the store digitally for any desired games. This can also be carried out by staff members if they want to ensure games have been successfully added to the database or just for maintenance purposes. The account creation for customers has also been established which allows new customers to create accounts for future login and use of the system.

The program debugs with ease and works as expected. The current prototyped system has already solved some underlying issues that were experienced with the anterior paper-based system such as different handwriting of staff members reducing the accuracy of the data stored. However, there is still a long way to go before this application can be implemented on the newly installed computers at the store.

Shortcomings and Improvements:

I definitely need to add more validation to ensure that the data a user inputs meets certain requirements e.g. range check, length check, format check and presence check which will subsequently prevent erroneous data from entering the system . I should also consider adding verification to my system to check that data has been inputted correctly, such as using techniques such as double entry. These processes will in turn lessen the amount of issues caused when customers or staff desire to utilise the information in the database to either generate outputs or input into them.

Post Prototype

Introduction:

In this section of the project I will describe what feedback I have received from my client and then display the changes to my project from any ideas that I have put on board. Moreover, I will explain how these changes have affected the system as well as how they’ve impacted how the system is used by both members and staff.

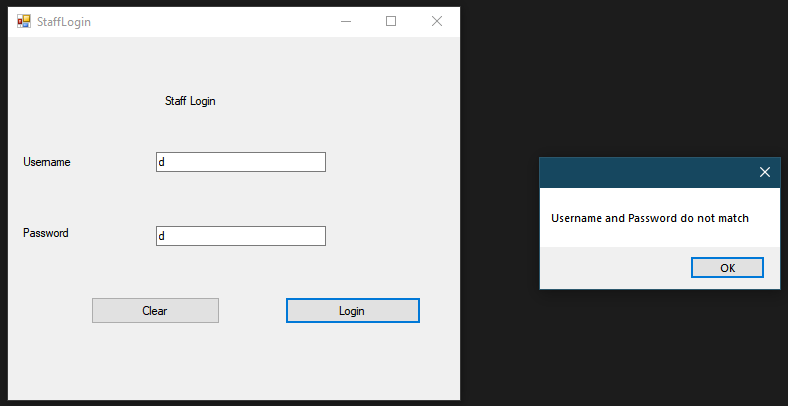
General and third party feedback:

I presented my prototype to the staff members and some volunteering customers from Mongoose Games. After having them test it, I was given some valued feedback about my system. The customers told me that they liked the account creation screen, and that it was very intuitive and asked appropriate personal details from them (which complies with GDPR rules), yet they demanded that a feature were implemented for them to be able to view/update their details to ensure accuracy of their details, they also wanted a feature to view their purchase history, as well as being able to add games to their wishlist and be notified when the game gets in stock. In addition, the staff members were ecstatic with the system and were impressed with the variety of important functions that I managed to prototype successfully for them. For example the Game Search function which they particularly enjoyed as in comparison to the paper based system, where they would have to flick through all of the pages in the notepads to find a certain game, they just need to know the title and all relevant details are shown to them. However, they made it clear that I needed to add a “return to main menu” button on each function for when they would want to carry out various tasks in one instance of the program without having to restart every time which would be inconvenient and time consuming. They also wanted to make the system more professional and time efficient, that during tasks such as adding a game or calculating a buying price for a game that they would be a “Clear” button which would clear all of the fields so they wouldn’t have to manually clear the fields to populate them again. The staff members also expressed their opinion on how the customers shouldn’t have access to the “quote calculator” for games as they might get inaccurate results, instead they should send a request to the staff members to calculate a quote for their game, and then be sent their quote back to them to let them know exactly how much they’ll get.

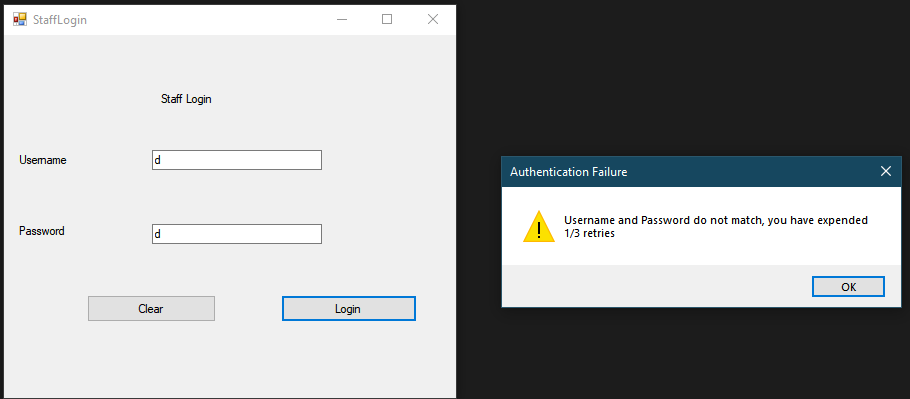
Feedback:

* I interviewed some colleagues and asked them questions about how they would feel about using the current system for a normal 9-5 working day which is what the staff members at Mongoose Games will end up doing. this made them think of ideas that would speed up processes or allow more detail to be obtained to facilitate transactions. After showing and discussing the prototype with them this is the feedback they gave me:
* My first colleague told me that both staff members and customers should have a limit as to how many times they can attempt to login, which will be viewed by them receiving a message after every login attempt, informing them of how many attempts they have and how many attempts they have made. After 3 attempts, the system will close itself.
* The second colleague then requested to me that more information should be stored on customers such as DOB, town and postcode to be able to differentiate customers with more ease and have more detail on where they reside.
* The first colleague them suggested to me, prior to reviewing the prototype, that in the Buying price calculator, the fields Popularity and Condition of the case should be populated using numeric up down boxes to reduce human error which would give a wrong, more expensive price which could cause the business to lose money.

Before changes

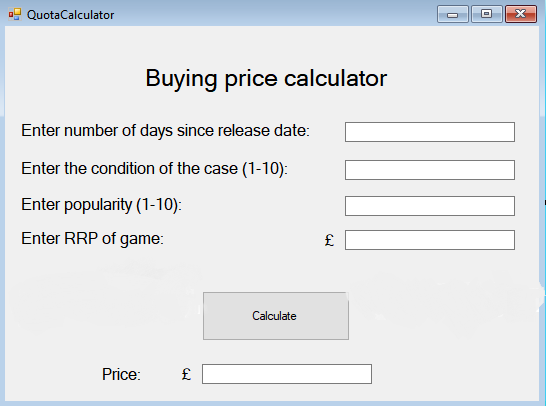


After changes

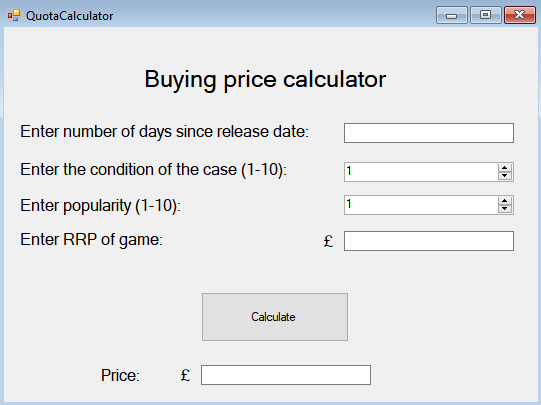


This is a screenshot of invalid credentials being inputted into the system before and after adding the 3 try limit. Every time they attempt to log in but fail, they are notified of this, as well as how many retries they have left. Once they surpass 3 attempts, the system will shut itself down to prevent further login attempts from being made. This is something that my client made very clear to me that he wanted during our interview as it prevents against brute force attacks from hackers in an attempt to gain access to the system which they could then access personal customer details and use them maliciously. I have chosen to include it into the messagebox instead of using a textbox as it gets the user’s attention more and they’re more likely of realising the presence of the 3 attempt limit.

Before changes

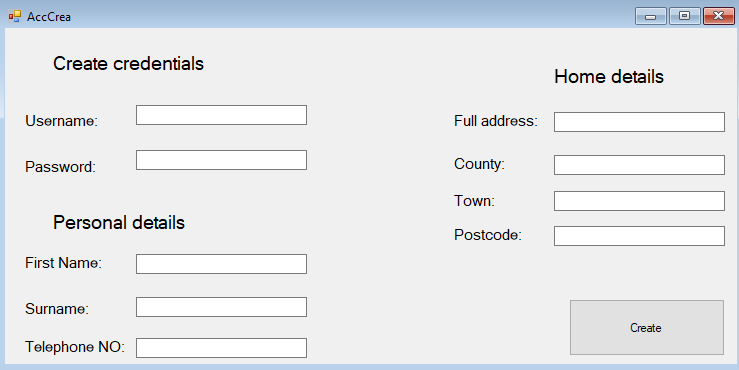


After changes

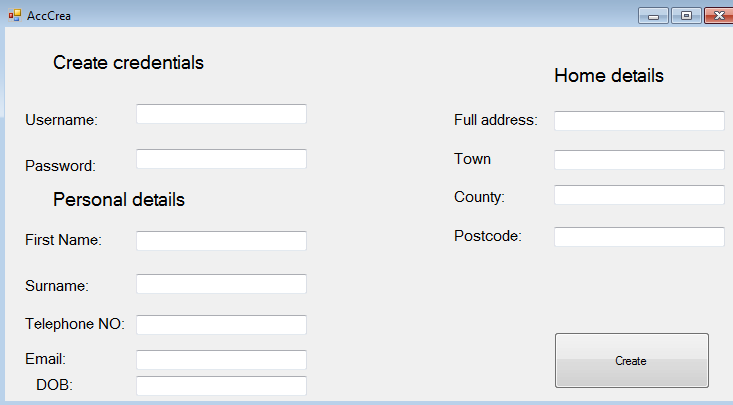


This is the screen of the Buying price calculator that staff use to calculate a quote they should give the customers for a game. As it can be seen, I have changed the text boxes that I used for the condition and popularity of the game into Numeric up and down boxes. I did this in order to implement a form of erroneous data entry prevention which could potentially affect the profitability of the business if it were to be entered. It also helps the staff member keep into the set boundaries (1-10) of the category and could be quicker to enter in some instances as the keyboard won’t have to be used.

Before



After



After the changes I have added fields to the customer file to store their date of birth, their town and postcode to be able to differentiate customers with more ease and have further information on where they reside for general purposes. It could also help us identify trends in our customer base by data such as age and their consequent spending habits, which could help us better tailor our stock.

Refinement of design interfaces

Original

Name:

Address:

County:

Telephone No.

Email:

Clear

Continue

Labels

txtName

txtAddress

txtCounty

txtTelephone

txtEmail

BtnCont

BtnClear

Customer Details

Refined

Forename:

Telephone No

Email:

Clear

Continue

BtnCont

BtnClear

Customer Details

**Personal details**

Surname:

DOB:

**Home details**

Address:

Town:

County:

Postcode:

txtForename

txtSurname

txtPhoneNum

txtEmail

txtDOB

txtAddress

txtTown

txtCounty

txtPostcode

Labels

I have refined the design interface for the customer details, adding all of the new requirements for the customer to enter and with a more intuitive structure.

Original

Buying price calculator

Enter number of days since release date:

Enter condition of case (1-10):

Enter popularity: (1-10)

Enter RRP of game:

Labels

TxtDays

TxtCondition

TxtPop

TxtRRP

Calculate

Price:

£

BtnCalculate1

TxtTotalPrice

Refined

Buying price calculator

Enter number of days since release date:

Enter condition of case (1-10):

Enter popularity: (1-10)

Enter RRP of game:

Labels

TxtDays

NumUpDownCondition

NumUpDownPop

TxtRRP

Calculate

Price:

£

BtnCalculate1

TxtTotalPrice



1

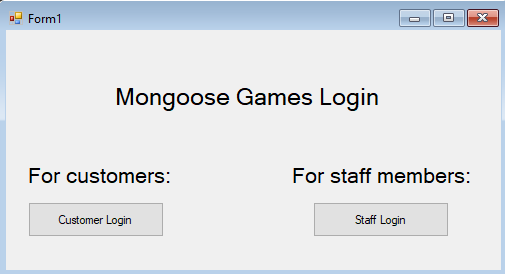
1

I have refined the design interface for the buying price calculator, taking into account the NumericUpDown boxes that I have chosen to implement for the condition and popularity in replacement for text boxes.

Software Development

Introduction: During this section of the project, I will provide screenshots of all of the forms in my project along with the corresponding bodies of code which relate to their functionality, proving the validity of my project. Furthermore, I will explain how data is entered into the system as well as showing how I have ensured my project to be user friendly and intuitive for both staff members and customers.

**Main Login form:**



This is the first thing any user sees when booting the system, the purpose of this part of the system is to allow the user to choose to login as a member of staff or as a customer, depending on which category they fall under.

They will then be directed to the login page of their choice, in order for them to access the system using their predefined credentials.

----------------------------------------------------------------------------------------------------------------

Imports System.Data.OleDb

Public Class Form1

Private Sub BtnStaffLogin\_Click(sender As Object, e As EventArgs) Handles BtnStaffLogin.Click

' Hides the MainLogin form

Hide()

' Shows the Member Login form when BtnMemberLogin is clicked

StaffLogin.Show()

End Sub

Private Sub BtnCustomerLogin\_Click(sender As Object, e As EventArgs) Handles BtnCustomerLogin.Click

' Hides the MainLogin form

Hide()

' Shows the Customer Login form when BtnCustomerLogin is pressed

CustomerLogin.Show()

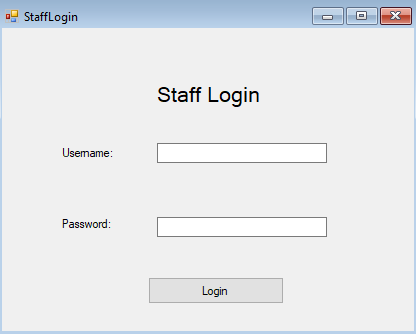
End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Staff Login:**

This form allows a staff member to access the system using their valid credentials (provided by), if the wrong credentials are inputted, the text boxes are cleared and they are then informed of their 2 remaining tries to access the system.



----------------------------------------------------------------------------------------------------------------

Imports System.Data.OleDb

Public Class StaffLogin

Dim connString As String

Public myConnection As OleDbConnection = New OleDbConnection

Public dr As OleDbDataReader

Public attempts As Integer

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles BtnStaffLogin.Click

Try

' Check if username or password is empty

If TxtStaffPassword.Text = "" Or TxtUsername.Text = "" Then

MessageBox.Show("Please complete the required fields..", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Both fields was supply

' Check if user exist in database

' Connect to DB

Dim conn As New System.Data.OleDb.OleDbConnection()

' Connect with database on the current system

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

Dim sql As String = "SELECT \* FROM Staff WHERE StaffUser='" & TxtUsername.Text & "' AND StaffPass = '" & TxtStaffPassword.Text & "'"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If they enter credentials that match to the database, they are then redirected to the customer menu

If sqlRead.Read() Then

MsgBox("Login successful")

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

' if they expend their 3 attempts, they are then notified of this and the application is closed

ElseIf attempts = 2 Then

MessageBox.Show("You have expended your 3 attempts, system shutting down", "Security procedure")

Close()

Else

' If user enter wrong username and password combination

' Throw an error message

' Every time they get the credentials wrong, the attempts value increments by 1 to register that an attempt has been made

attempts += 1

MessageBox.Show("Username and Password do not match, you have expended " & attempts & "/3 retries", "Authentication Failure", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

'Clear all fields

TxtStaffPassword.Text = ""

TxtUsername.Text = ""

'Focus on Username field (so that they can promptly retry without having to use their mouse)

TxtUsername.Focus()

' Closes the connection to the data source

conn.Close()

End If

End If

' Catch exception is caught here, so that if there is an issue connecting to the database they are aware of this, instead of the program crashing

Catch ex As Exception

MessageBox.Show("Failed to connect to Database..", "Database Connection Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Staff Main Menu:**

****

This is the base of the system for the staff members, and allows them to access any function of the system that they want by clicking on the relevant button. The structure is intuitive and the labels are self-explanatory, to ensure that staff members will be able to use the system effectively and efficiently with little training required.

A return to main menu function has been included with each subsequent form that can be accessed from this menu, to allow staff members to carry out multiple tasks in one instance.

A logout button has also been implemented into the main menu, which ensures the changes in shifts are fluent as a staff member can log out of their account to allow another to access the system without having to restart the program, which saves a lot of time and effort. It can also be used as a safety feature, if the staff member is taking a break, they can log out to ensure that when they’re gone no unauthorised personnel can access the system and cause malicious damage.

----------------------------------------------------------------------------------------------------------------

Public Class StaffMenu

Private Sub BtnQuotaCalc\_Click(sender As Object, e As EventArgs) Handles BtnQuotaCalc.Click

Me.Hide()

QuotaCalculator.Show()

End Sub

Private Sub BtnResell\_Click(sender As Object, e As EventArgs) Handles BtnResell.Click

Me.Hide()

ResellCalculator.Show()

End Sub

Private Sub BtnAddGame\_Click(sender As Object, e As EventArgs) Handles BtnAddGame.Click

Me.Hide()

AddGame.Show()

End Sub

Private Sub BtnRecentGames\_Click(sender As Object, e As EventArgs) Handles BtnRecentGames.Click

Me.Hide()

ViewRecentGames.Show()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles BtnSearchGame.Click

Me.Hide()

SearchGames.Show()

End Sub

Private Sub Button7\_Click(sender As Object, e As EventArgs) Handles BtnUpdateDetails.Click

Me.Hide()

UpdateCustDetails.Show()

End Sub

Private Sub BtnViewDetails\_Click(sender As Object, e As EventArgs) Handles BtnSearchCust.Click

Me.Hide()

ViewCustDetails.Show()

End Sub

Private Sub BtnLogout\_Click(sender As Object, e As EventArgs) Handles BtnLogout.Click

Me.Hide()

Form1.Show()

End Sub

Private Sub Button1\_Click\_1(sender As Object, e As EventArgs) Handles Button1.Click

Me.Hide()

ViewCustomerRequests.Show()

End Sub

Private Sub BtnInform\_Click(sender As Object, e As EventArgs) Handles BtnInform.Click

Me.Hide()

RequestReply.Show()

End Sub

Private Sub BtnSuitCheck\_Click(sender As Object, e As EventArgs)

Me.Hide()

SuitChecker.Show()

End Sub

Private Sub BtnViewCustomers\_Click(sender As Object, e As EventArgs) Handles BtnViewCustomers.Click

Me.Hide()

ViewAllCustomers.Show()

End Sub

Private Sub BtnViewGames\_Click(sender As Object, e As EventArgs) Handles BtnViewGames.Click

Me.Hide()

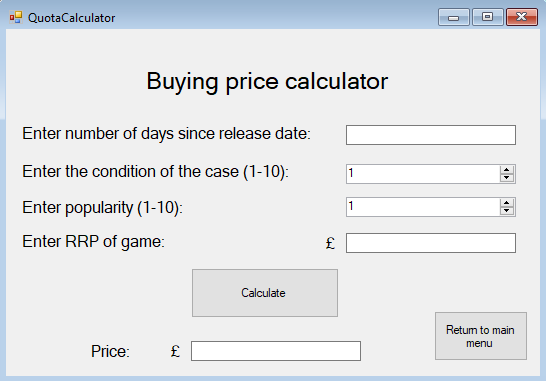
ViewAllGames.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Buying price calculator:**

****

The buying price calculator (quote calculator) allows for a staff member to calculate the amount that they will pay a customer for their game based upon multiple factors that they will input in the corresponding text boxes such as the age of a game, condition of case, popularity and RRP.

The age of the game will be calculated by subtracting the release date from the current date.

The condition of the case will be determined by the staff member after a visual inspection.

The popularity will be determined by the staff member, after considering factors such as sales, reviews, scores and press coverage.

The RRP will be found out by searching how much the game was when it first game out, in brand new condition.

After all of these factors are inputted into the system, the algorithm will take all of the factors and produce a result and populate it into the price textbox.

----------------------------------------------------------------------------------------------------------------

Public Class QuotaCalculator

Private Sub BtnCalc\_Click(sender As Object, e As EventArgs) Handles BtnCalc.Click

Try

' Factors inputted are converted into variables with the double datatype, to allow any sort of calculations under any circumstances to be performed

Dim GameAge As Integer = Convert.ToDouble(txtGameAge.Text)

Dim condition As Integer = Convert.ToDouble(NumUpDownCondition.Value)

Dim popularity As Integer = Convert.ToDouble(NumUpDownPopularity.Value)

Dim RRP As Integer = Convert.ToDouble(txtRRP.Text)

' rules for games under a year old

' for every 1 condition value below 10, the price is subtracted by £1.5

' the less popular a game is, the more they are worth less than the RRP

If popularity >= 7 And GameAge <= 365 Then

txtPrice.Text = (RRP \* 0.75) - (10 - condition) \* 1.5

ElseIf popularity >= 5 And GameAge <= 365 Then

txtPrice.Text = (RRP \* 0.65) - (10 - condition) \* 1.5

ElseIf popularity >= 3 And GameAge <= 365 Then

txtPrice.Text = (RRP \* 0.5) - (10 - condition) \* 1.5

ElseIf popularity < 2 And GameAge <= 365 Then

txtPrice.Text = (RRP \* 0.3) - (10 - condition) \* 1.5

' now rules for games over a year old

' for every 1 condition value below 10, the price is subtracted by £1.5

' the less popular a game is, the more they are worth less than the RRP

ElseIf popularity >= 7 And GameAge > 365 Then

txtPrice.Text = (RRP \* 0.7) - (10 - condition) \* 1.5

ElseIf popularity >= 5 And GameAge > 365 Then

txtPrice.Text = (RRP \* 0.6) - (10 - condition) \* 1.5

ElseIf popularity >= 3 And GameAge > 365 Then

txtPrice.Text = (RRP \* 0.45) - (10 - condition) \* 1.5

ElseIf popularity < 2 And GameAge > 365 Then

txtPrice.Text = (RRP \* 0.25) - (10 - condition) \* 1.5

' If there is an error while trying to fit the inputted values into a category for a respective quote, the staff member is notified of this.

Else

MessageBox.Show("The game's characteristics do not match any field ", "Error while calculating quote")

End If

' If there is some sort of error, the staff member is made aware of it, without the program crashing.

Catch ex As Exception

MessageBox.Show("Please enter the required fields", "Error While compiling values")

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Hide current form

Me.Hide()

' Show staff menu

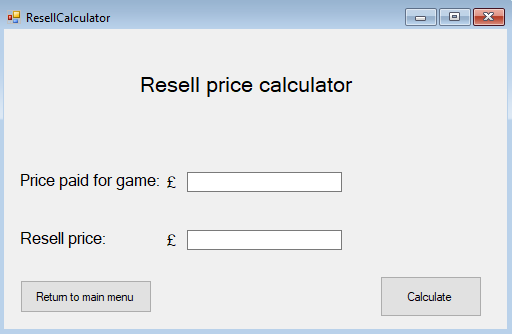
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Resell price calculator**

****

This part of the system allows a staff member to calculate the price that they will sell a game that has been bought from a customer for. This is important as it ensures the profitability of the business and eliminates human error and/or bias/subjective perspectives which will promote a healthy and thriving business. The staff member will enter the price they paid for the game and then the price that they should put the game back in stock for will be calculated for them. The algorithms increments the price paid by 25% for a worthy profit margin to be made.

----------------------------------------------------------------------------------------------------------------

Public Class ResellCalculator

Private Sub BtnCalculate\_Click(sender As Object, e As EventArgs) Handles BtnCalculate.Click

' Converts the string input to integer to allow the value to be manipulated mathematically

Dim PricePaid As Integer = Convert.ToDouble(txtPricePaid.Text)

' The price is incremented by 25%

txtResellPrice.Text = PricePaid \* 1.25

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Hides current form

Me.Hide()

' Shows staff menu

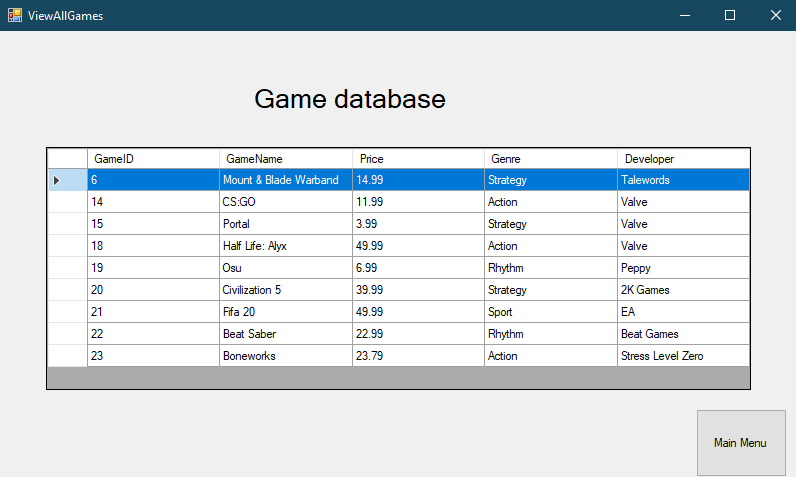
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all available games on system**



This is a vital part of the system which allows the staff member to view all of the games that are currently available in store to be bought. This is a convenient part of the system as the staff member can use this feature to be able to verify that a game they have added to the system is now in the database and if a game that they have deleted has now been removed from the database.

Moreover, this feature can be used to confirm to customers that games are still in stock, if they are in a hurry or simply don’t have time to log onto a computer and will be given a prompt response on their query. This helps boost customer satisfaction as well as providing a major improvement from the prior system, where the staff member would then have to physically check whether the game was in the store or not.

Furthermore, this part of the system enables staff members to see the quantity of each game on the system, by ordering the records by GameName. This could prove useful when a customer wants to sell a game, and then after using this form, the staff member realises there are already 3 copies of that same game, and so decides to inform the customer that they are unable to purchase their game as it’s overstocked. This prevents a large amount of the same game being bought, which could then affect the business as the game would evidently have a low demand.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewAllGames

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

Me.Hide()

StaffMenu.Show()

End Sub

Private Sub ViewAllGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

GamesDataGridView.DataSource = GetGameInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetGameInformation() As DataTable

' Defines the data table

Dim Games As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM Games", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Games.Load(reader)

End Using

End Using

' The function returns the data table

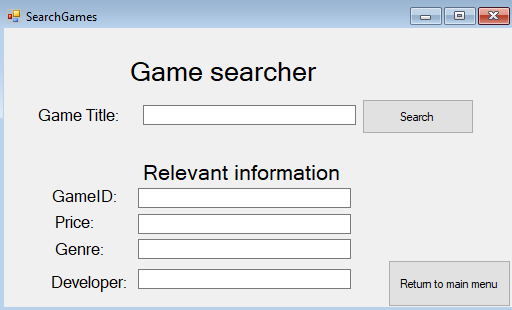
Return Games

End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Search for a game on the system**

****

This feature allows the staff member to search for a certain game on the system by its name, where its corresponding information will then be given to them. They will need to input the game’s title, then click the button “Search”, they would then be notified if the game is found or not, and if it is, its information will be inputted in its corresponding text box, improving readability. This is important as it can be used to see if a game on a customer’s wishlist is in stock, and they could then subsequently report back to them, instead of having to search through the whole list of available games, or even worse, as how it used to be done, search through all of the notepads and paperwork, looking for the game. Overall, it’s an essential feature and a major improvement.

----------------------------------------------------------------------------------------------------------------

Public Class SearchGames

Private Sub BtnSearch\_Click(sender As Object, e As EventArgs) Handles BtnSearch.Click

' Connection is created

Dim conn As New System.Data.OleDb.OleDbConnection()

' Database location defined

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement which returns all of the fields from the Games database where its name is equal to what the staff member searched

Dim sql As String = "SELECT GameID,GameName,Price,Genre,Developer FROM Games WHERE (GameName) = ('" & txtGameTitle.Text & "')"

' Allows the SQL command to communicate with the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Open SQL conncetion

sqlCom.Connection = conn

'Open Database Connection

conn.Open()

' Executes the SQL command to the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If the game being searched for is found then this

If sqlRead.HasRows Then

' Reads through each field until all returned and then populates the relevant text boxes with each field value

While sqlRead.Read()

txtGameID.Text = sqlRead.Item("GameID")

txtGameTitle.Text = sqlRead.Item("GameName")

txtPrice.Text = sqlRead.Item("Price")

txtGenre.Text = sqlRead.Item("Genre")

txtDeveloper.Text = sqlRead.Item("Developer")

' User is notified of game being found

MessageBox.Show("Game found", "Searching of game successful")

End While

' Closes the read connection

sqlRead.Close()

Else

MessageBox.Show("No game called """ & txtGameTitle.Text & """" & " was found. Please try again", "Error while searching for game")

' All fields are cleared

txtGameID.Clear()

txtGameTitle.Clear()

txtPrice.Clear()

txtGenre.Clear()

txtDeveloper.Clear()

End If

' Closes database connection

conn.Close()

End Sub

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu

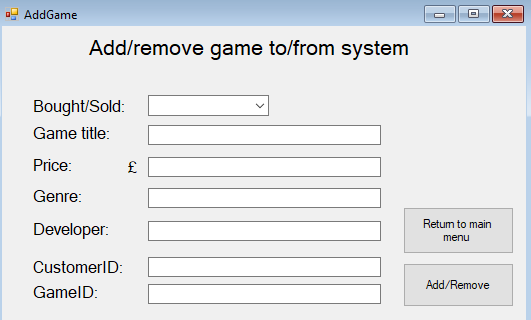
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Add/remove game from system**



This is one of the fundamental parts of the system, which enables staff members to add a game into the database after it’s bought from a customer, or remove a game from the database after it’s been sold to a customer. This will be used daily and is expected to be the most used function of the proposed system. The form is formatted intuitively, with the well-thought use of a bought/sold drop down box to avoid the implementation of two different functions on the system, which allows both of these tasks to be carried out through the same interface. Once the staff member enters all relative information, the database is subsequently manipulated by the system.

The “GameID” textbox will only have to be filled out when a game already exists in the database, and is being sold to a customer as it will have an assigned GameID, this would be explained to the staff member during training, or by hovering over the GameID label or related textbox, which through the use of a “ToolTip”, would inform them of this.

A staff member would be able to get all relevant information to be entered into the text box from the “View and search all available games” function for game information and “View customer details” function for the customerID, these will both be explained in detail later on in the project.

----------------------------------------------------------------------------------------------------------------

Imports System.Data.OleDb

Public Class AddGame

Private Sub BtnAdd\_Click(sender As Object, e As EventArgs) Handles BtnAdd.Click

' If the staff member wishes to add a game into the database as it has been bought by a customer, this action is carried out

If BoughtOrSold.Text = "Bought" Then

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which inserts what the staff member input into the database in the corresponding table

Dim sql As String = "INSERT INTO Games (GameName,Price,Genre,Developer) VALUES ('" & txtTitle.Text & "', '" & txtPrice.Text & "', '" & txtGenre.Text & "', '" & txtDeveloper.Text & "')"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Game added to system, you will now be redirected to the main menu screen", "Appending of game successful")

' Current form is hidden to allow the staff member to focus their attention on the AddToRecent form

Me.Hide()

' AddToRecent Form is shown, which asks the staff member if they wish to log the transaction into the system

AddToRecent.Show()

' Close database connection

'Open Database Connection again

sqlCom.Connection = conn

conn.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' If the staff member wishes to delete a game from the database as it has been sold to a customer, this action is carried out

ElseIf BoughtOrSold.Text = "Sold" Then

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which deletes the game that the staff member has referred to (using its ID) from the database

Dim sql As String = "DELETE \* FROM Games WHERE (GameID) = (" & txtGameID.Text & ")"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Game removed from system, you will now be redirected to the main menu screen", "Removal of game successful")

' Current form is hidden to allow the staff member to focus their attention on the AddToRecent form

Me.Hide()

' AddToRecent Form is shown, which asks the staff member if they wish to log the transaction into the system

AddToRecent.Show()

' Open connection with sql

sqlCom.Connection = conn

conn.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

End If

End Sub

' When main menu button is clicked

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Click

' Hides current menu

Me.Hide()

' Displays the staff main menu

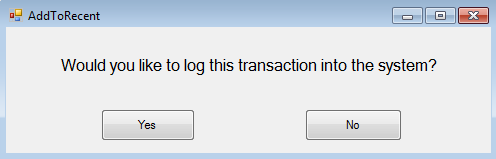
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Add to recent games**

****

After the staff member adds or removes a game, they will then be redirected to this form asking them if they wish to log this transaction into the system. If yes, it will be saved into the “RecentGames” table which stores all games that have been recently sold and bought from the system. If not, they will simply be redirected to the main menu.

----------------------------------------------------------------------------------------------------------------

Public Class AddToRecent

Private Sub btnYes\_Click(sender As Object, e As EventArgs) Handles btnYes.Click

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define location of database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=\\Ccw03usr02\2018$\WM180573\MongooseGames.accdb"

' Logs the transaction into the system

Dim sql As String = "INSERT INTO RecentGames (GameName,Price,Genre,Developer,BoughtOrSold,CustomerID) VALUES ('" & AddGame.txtTitle.Text & "', '" & AddGame.txtPrice.Text & "', '" & AddGame.txtGenre.Text & "', '" & AddGame.txtDeveloper.Text & "','" & AddGame.BoughtOrSold.Text & "', '" & AddGame.txtCustomerID.Text & "')"

' Represents the SQL statement to be executed against the data base

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member notified of successful logging of transaction

MessageBox.Show("Game added to system, you will now be redirected to the main menu screen", "Appending of game successful")

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

' Provides SQL command with the ability to read the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

End Sub

Private Sub BtnNo\_Click(sender As Object, e As EventArgs) Handles BtnNo.Click

' Hides current form

Me.Hide()

' Shows staff menu

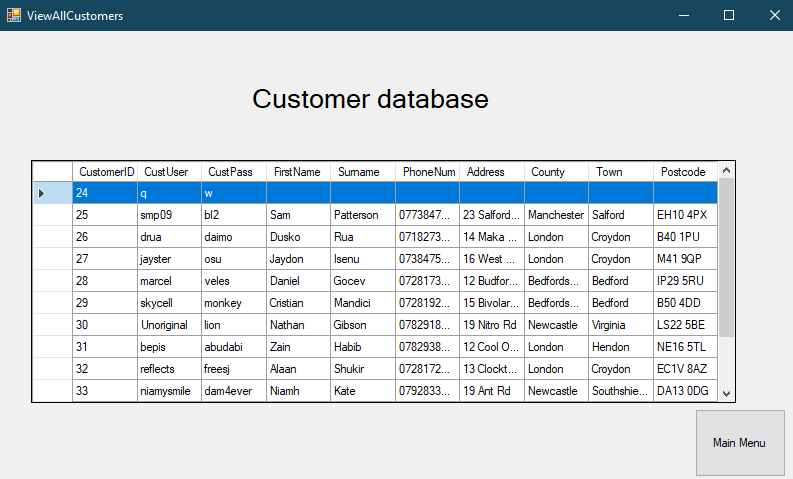
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Information Viewer**

****

As a staff member, being able to view all of the customers registered to the system is essential. The information from the database is imported into the DataGridView which allows the staff member to familiarise themselves with everyone on the system and to also verify that changes that they have made have been applied to the system.

Moreover, if a customer should have issues logging in to their account, they could ask a staff member if their details are stored into the system, and once they verify their identity, they can be given new credentials. This provides exceptional customer service and generally enables the staff member to be able to view what is stored in the database directly.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewAllCustomers

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu form

StaffMenu.Show()

End Sub

Private Sub ViewAllCustomers\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

CustomersDataGridView.DataSource = GetCustomerInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetCustomerInformation() As DataTable

' Defines the data table

Dim Customers As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM Customers", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Customers.Load(reader)

End Using

End Using

' The function returns the data table

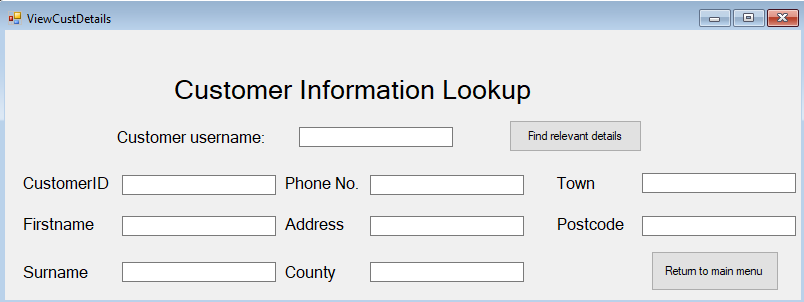
Return Customers

End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Information Lookup**



This form provides the staff members with the possibility of retrieving all of a certain customer’s details by providing their unique username. This will be used when a staff member buys a game from a customer, where they will need to know their customerID, they will simply ask the customer for their username and they’ll be able to retrieve their customerID from this form. Moreover, they will also be able to retrieve details such as their phone number and address, which will allow the staff member to contact the customer externally if need be.

----------------------------------------------------------------------------------------------------------------

Public Class ViewCustDetails

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

Private Sub BtnCustSearch\_Click(sender As Object, e As EventArgs) Handles BtnCustSearch.Click

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

' Retrieves the record of the customer with the username that the staff member inputs

Dim sql As String = "SELECT CustomerID,FirstName,Surname,PhoneNum,Address,County,Town,Postcode FROM Customers WHERE (CustUser) = ('" & TxtCustUser.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

TxtCustID.Text = sqlRead.Item("CustomerID")

TxtCustName.Text = sqlRead.Item("FirstName")

TxtCustSurname.Text = sqlRead.Item("Surname")

TxtCustPhone.Text = sqlRead.Item("PhoneNum")

TxtCustAddress.Text = sqlRead.Item("Address")

TxtCustCounty.Text = sqlRead.Item("County")

TxtCustTown.Text = sqlRead.Item("Town")

TxtCustPostcode.Text = sqlRead.Item("Postcode")

' Informs the staff member of their success of searching for a customer

MessageBox.Show("Customer found", "Searching of customer details successful")

End While

' Close connection between sql and database

sqlRead.Close()

Else

' If there are no customers with the username that the staff member inputted in the system they are informed of this and prompted to try again.

MessageBox.Show("No customer with the username of " & TxtCustUser.Text & " & "" was found. Please try again", "Error while searching for customer details")

' The customer username field is then cleared to faciliate the re-population of the textbox

TxtCustUser.Clear()

End If

' Connection of the database is closed

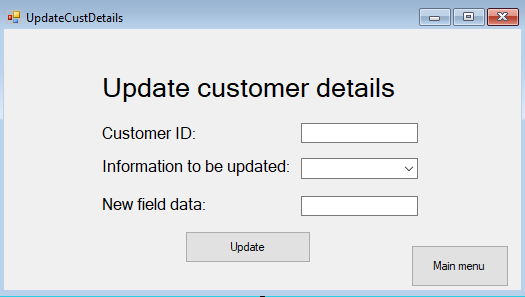
conn.Close()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Update customer Information:**



This form allows staff members to update a customer’s record by entering their CustomerID and then the data they would like to replace the certain field with. They are able to select, using the drop down menu what field they would like to update of the customer.

This form would be used when a customer would approach a staff member, informing them of a change in their information e.g. Address (moved house), phone number (bought a new phone) or surname (got married). This complies with GDPR rules, as it allows customers to request the data that is stored about them is to be changed in order to be accurate.

The UI uses very little text boxes and buttons i.e. avoids the use of using multiple text boxes that correspond to the different fields which can be edited separately, reducing confusion and clutter. This makes the interface a lot more digestible which can lead to more efficient and effective use of the program as more often than not, only a few fields will have to be updated rather than the entire customer record.

----------------------------------------------------------------------------------------------------------------

Public Class UpdateCustDetails

Private Sub BtnUpd\_Click(sender As Object, e As EventArgs) Handles BtnUpd.Click

' Converts the textbox value for the CustomerID into an integer to enable for it to be used in the SQL statement

Dim CustomerID As Integer = Convert.ToDouble(txtCustID.Text)

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which changes the value of the desired field to the desired value in the record of the desired customer

Dim sql As String = "UPDATE Customers SET " & FieldName.SelectedItem & " = """ & txtNewData.Text & """ WHERE CustomerID = " & CustomerID & ""

' Executes the statement against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects the sql statement to the database

sqlCom.Connection = conn

' Opens the connection

conn.Open()

' Staff member is notified of their success of their required task.

MessageBox.Show("Field value updated", "Database record updation successful")

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

End Sub

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides the current form

Me.Hide()

' Shows the staff menu

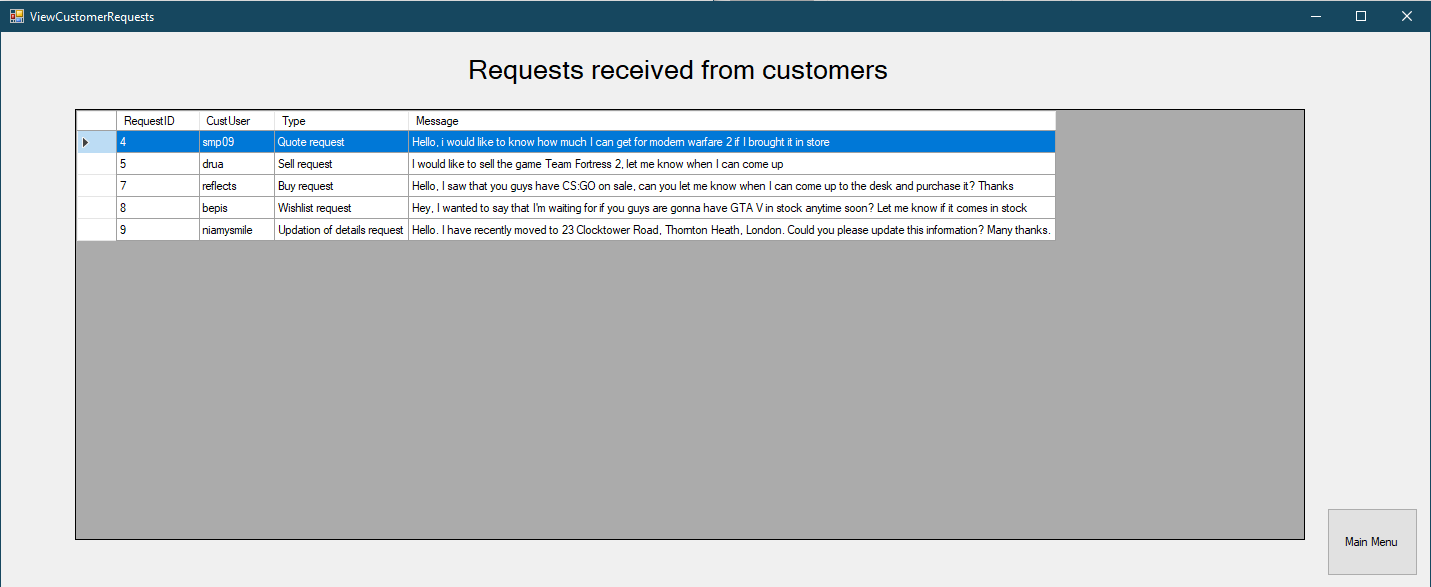
StaffMenu.Show()

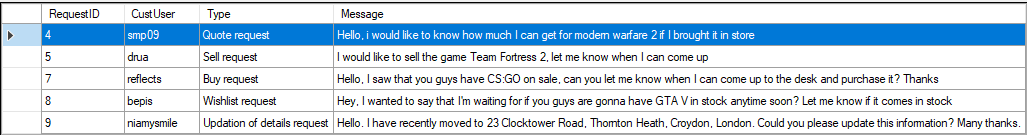
End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View customer requests:**

****

****

This is a vital part of the program, which enables a plethora of actions to be completed. It allows customers to sell and buy games, as well as requesting games to be brought in stock. Crucially, it allows customers to request that the details stored about them are updated, which is necessary in order for Mongoose Games to comply with GDPR rules, that the data should always be accurate - to the customer’s liking.

As soon as the form opens, the information from the table is loaded into the DataViewGrid. I have ensured that the CustUser and Type of request is present, to enable the staff member to know what they’re requesting and then subsequently communicate back to the customer, informing them of an advance on their request. I have also changed some formatting options to remove the scroll bar from the form, as I believe it will cause delays when reading customer requests. I have also made it so when a staff member selects a record, that the whole record is highlighted not just one field, to enable the staff member to read the message more easily. In addition, this allows the entire record to be copied at once, which will prove very useful for when the staff member wants to reply to the user’s request, and will need the user’s username as well as the RequestID which they’ll be able to paste in with ease. Moreover, it would help them keep track of their request type and relevant message while they are writing a response.

The requests can also be ordered by any field e.g. RequestID this could be useful if the staff member wants to prioritise handling the requests that were sent first, to reduce waiting times for customers. Sorting by the Type could also be used if the staff member wishes to prioritise selling/buying requests rather than wishlist or updating of information requests as although they’re important, the economic transactions are valued higher to ensure maximum business profitability.

----------------------------------------------------------------------------------------------------------------

Public Class RequestReply

Private Sub BtnSend\_Click(sender As Object, e As EventArgs) Handles BtnSend.Click

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=\\Ccw03usr02\2018$\WM180573\MongooseGames.accdb"

' Opens connection to the database

conn.Open()

' SQL (Structured Query Lanugage) statement which inserts what the staff member input into the database in the corresponding table

Dim sql As String = "INSERT INTO StaffActions(StaffUser,CustUser,Message) VALUES ('" & StaffLogin.TxtUsername.Text & "', '" & txtCustUser.Text & "', '" & txtMsg.Text & "')"

' Allows the sql command to be executed against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Message sent", "Sending of message successful")

'Open Database Connection again

sqlCom.Connection = conn

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' Hides current form

Me.Hide()

' Redirectes to form that asks them if they wish to now delete the original request from the system

DeleteFromRequests.Show()

End Sub

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu form

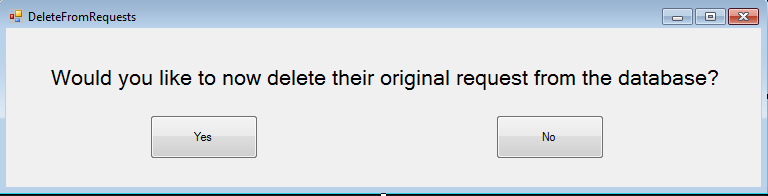
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Subsequent deletion of original customer request**



This is the form that will appear after the staff member has replied to a user request. If they choose to delete the original request (which will be done usually to avoid two staff members replying to one request and to reduce clutter) it will be removed from the database and then they will be redirected to the main menu. If not, they will simply be redirected to the main menu. This is useful as it could’ve been a wishlist request and the staff member would might want to keep it in the inbox so they don’t forget the customer has made that request, to then later on inform them of any updates relating to the availability of their game.

----------------------------------------------------------------------------------------------------------------

Public Class DeleteFromRequests

Private Sub BtnYes\_Click(sender As Object, e As EventArgs) Handles BtnYes.Click

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=\\Ccw03usr02\2018$\WM180573\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which deletes the request that the staff member has just dealt with from the system

Dim sql As String = "DELETE \* FROM CustRequests WHERE (RequestID) = (" & RequestReply.txtRequestID.Text & ")"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Request removed from system, you will now be redirected to the main menu screen", "Removal of request successful")

' Current form is hidden

Me.Hide()

' redirected to main menu

StaffMenu.Show()

' Open connection with sql

sqlCom.Connection = conn

conn.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

End Sub

Private Sub BtnNo\_Click(sender As Object, e As EventArgs) Handles BtnNo.Click

' Hides current form

Me.Hide()

' Shows menu

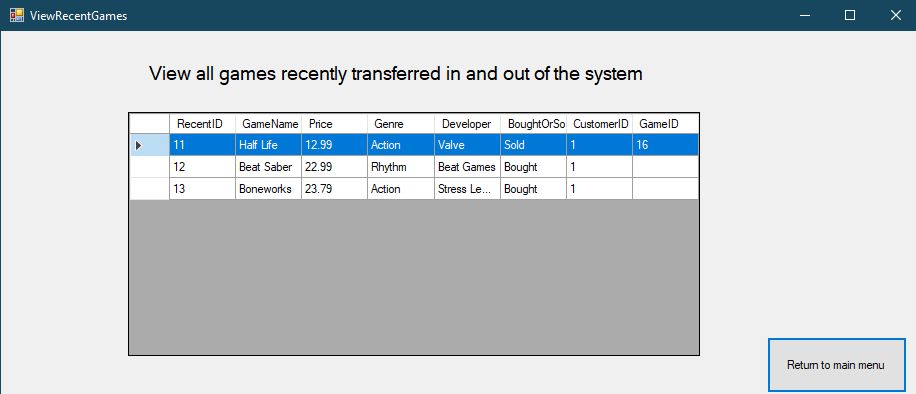
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all recent transactions from the system**

****

Viewing all of the recently sold/bought games is an essential feature as it allows Mongoose Games to keep track of what games are being entered and removed from the database which is important as it could be used to identify customer purchase patterns.

As with the ViewCustomerRequests form, the layout and design is clear and the staff members are able to easily identify the details of each record. This feature could also be useful as it could provide a means of verifying that a customer has bought/sold a game, should this proof be needed.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data

Imports System.Data.OleDb

Public Class ViewRecentGames

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

Private Sub ViewRecentGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

RecentGamesDataGridView.DataSource = GetRecentGames()

End Sub

' Creates function which returns the table to the staff member

Private Function GetRecentGames() As DataTable

' Defines the data table

Dim RecentGames As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM RecentGames", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

RecentGames.Load(reader)

End Using

End Using

' The function returns the data table

Return RecentGames

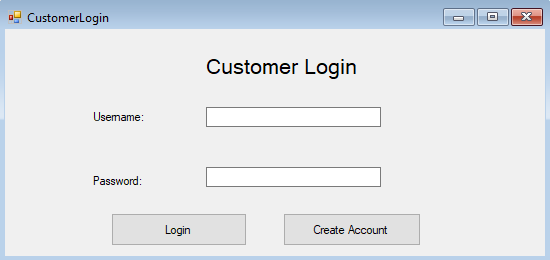
End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Customer system features**

**Customer Login:**

****

This form is accessed after clicking the “Customer Login” button on the main form and allows a customer to access the system using their valid credentials (created prior using Create Account function shown below), if the wrong credentials are inputted, the text boxes are cleared and they are then informed of their 2 remaining tries to access the system.

----------------------------------------------------------------------------------------------------------------

Public Class CustomerLogin

Dim attempts As Integer

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

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

' Check if username or password is empty

Try

If txtPassword.Text = "" Or txtUsername.Text = "" Then

MessageBox.Show("Please complete the required fields..", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Both fields was supply

' Check if user exist in database

' Connect to DB

Dim conn As New System.Data.OleDb.OleDbConnection()

' Connect with database on the current system

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\Downloads\MongooseGames.accdb"

Dim sql As String = "SELECT \* FROM Customers WHERE CustUser='" & txtUsername.Text & "' AND CustPass = '" & txtPassword.Text & "'"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If they enter the current combination of credentials, they are then redirected to the customer menu

If sqlRead.Read() Then

MsgBox("Login successful")

' Hides current form

Me.Hide()

' Shows menu for customers

CustomerMenu.Show()

' if they expend their 3 attempts, they are then notified of this and the application is closed

ElseIf attempts = 2 Then

MessageBox.Show("You have expended your 3 attempts, system shutting down", "Security procedure")

Close()

Else

' If user enter wrong username and password combination

' Throw an error message

' Every time they get the credentials wrong, the attempts value increments by 1 to register that an attempt has been made

attempts += 1

MessageBox.Show("Username and Password do not match, you have expended " & attempts & "/3 retries", "Authentication Failure", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

'Clear all fields

txtPassword.Text = ""

txtUsername.Text = ""

'Focus on Username field (so that they can promptly retry without having to use their mouse)

txtUsername.Focus()

' Closes the connection to the data source

conn.Close()

End If

End If

' Catch exception is caught here, so that if there is an issue connecting to the database they are aware of this, instead of the program crashing

Catch ex As Exception

MessageBox.Show("Failed to connect to Database..", "Database Connection Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' if account create button is pressed the form is hidden

Hide()

' Then the account creation screen is shown

AccCrea.Show()

End Sub

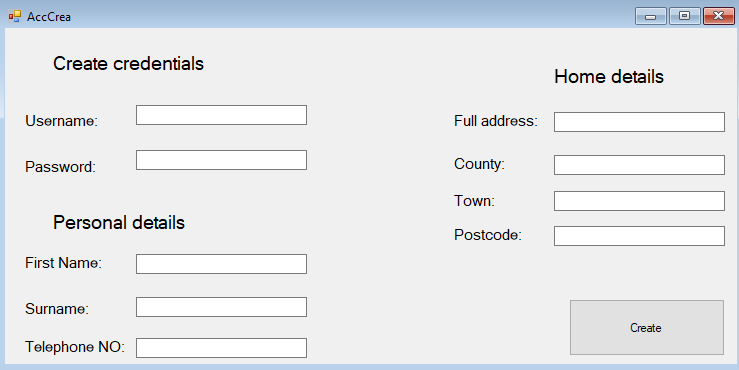
End Class

----------------------------------------------------------------------------------------------------------------

**Customer account creation**

If it’s the customer’s first time accessing the system, they will need to create an account first along with their corresponding information, which will then grant their future access to the system. This is then handled by the form below which is presented when a customer presses the “Create account” button on the customer login (form shown above)

Validation has also been added to ensure that two people with the same username can’t exist, this is done by checking, before the account is saved to the database, if the desired username already exists in the database, if so they are notified of this and have to change the username, if not, it will go through normally. This is done in order to prevent any clashes of data and to ensure that messages are sent to the correct recipients.



----------------------------------------------------------------------------------------------------------------

' Allows regex commands to be used to identify characters or numbers in inputs to aid validation

Imports System.Text.RegularExpressions

Public Class AccCrea

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Check if any field is empty and if any of them are empty

If txtPassword.Text = "" Or txtUsername.Text = "" Or txtFirstName.Text = "" Or txtSurname.Text = "" Or txtTelephone.Text = "" Or txtAddress.Text = "" Or txtCounty.Text = "" Or txtTown.Text = "" Or txtPostcode.Text = "" Then

' If so, then the user is notified of this and they are to then fill them in

MessageBox.Show("Please complete all required fields...", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if the password is the same as the username

ElseIf txtUsername.Text = txtPassword.Text Then

' If so, user is then to ensure that their username differs from their password

MessageBox.Show("Please ensure that your username is different from your password", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the length of the customer's username or password is less than 5

ElseIf txtUsername.Text.Length < 5 Or txtPassword.Text.Length < 5 Then

' If so, they are prompted to make their username/password longer than 5 characters so that their credentials are more sophisticated

MessageBox.Show("Please ensure that your username/password is longer than 5 characters", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the phone number entered is less/more than 11 digits long

ElseIf txtTelephone.Text.Length <> 11 Or IsNumeric(txtTelephone.Text) = False Then

' If so, they are informed that they need to ensure their phone number exactly 11 characters long

MessageBox.Show("Please ensure that your phone number is 11 digits long", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text box to faciliate re-entering of data

txtTelephone.Clear()

' Checks if the postcode entered is 7 or 8 characters long and if it contains a space

ElseIf txtPostcode.Text.Length < 7 Or txtPostcode.Text.Length > 8 Or txtPostcode.Text.Contains(" ") = False Then

' If not, they are informed that they need to enter a valid postcode

MessageBox.Show("Please ensure that your postcode is either 7 or 8 characters long, including a space in between the two parts ", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if any field which shouldn't be composed of digits e.g. firstname contains any digits

ElseIf Regex.Match(txtFirstName.Text, "\d").Success Or Regex.Match(txtSurname.Text, "\d").Success Or Regex.Match(txtCounty.Text, "\d").Success Or Regex.Match(txtTown.Text, "\d").Success Then

' If so, they are prompted to reevaluate this

MessageBox.Show("Please ensure that there are no numbers in fields which don't require them", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Define new connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Find records that have the same username as someone else on the system

Dim sqlValidation As String = "SELECT \* FROM Customers WHERE CustUser='" & txtUsername.Text & "'"

' Allows sql to communicate with database

Dim sqlComValidation As New System.Data.OleDb.OleDbCommand(sqlValidation)

' Connect SQL statement to database

sqlComValidation.Connection = conn

'Open Database Connection

conn.Open()

' Executes SQL statement against database

Dim sqlReadValidation As System.Data.OleDb.OleDbDataReader = sqlComValidation.ExecuteReader()

' If username is already in use

If sqlReadValidation.Read() Then

' User is notified that username is already in use and they must change it

MsgBox("Username is already in use, please try another one")

Else

' Inserts all of the information that the user inputs into the database

Dim sqlInsert As String = "INSERT INTO customers (CustUser,CustPass,FirstName,Surname,PhoneNum,Address,County,Town,Postcode) VALUES ('" & txtUsername.Text & "', '" & txtPassword.Text & "', '" & txtFirstName.Text & "', '" & txtSurname.Text & "', '" & txtTelephone.Text & "', '" & txtAddress.Text & "', '" & txtCounty.Text & "', '" & txtTown.Text & "', '" & txtPostcode.Text.ToUpper & "')"

' Allows SQL statement to communicate with databasde

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sqlInsert)

' They are then informed of their account creation

MessageBox.Show("Account Created, and details saved, you will now be redirected to the login screen", "Account creation successful")

' Current form hidden

Me.Hide()

' Directed to the customer login form to now use their newly created credentials to gain access to the system

CustomerLogin.Show()

'Connects SQL statement with database

sqlCom.Connection = conn

' Executes SQL statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Closes connection

conn.Close()

End If

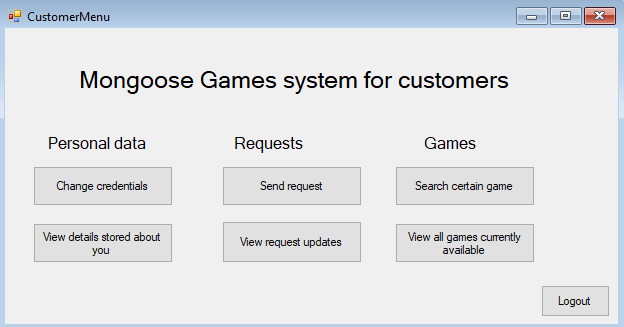
End If

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Main Menu:**



Once a customer has logged in successfully, they are redirected to the main menu, which is the hub of all the functionalities they will be able to carry out as a customer.

Depending on what they wish to do, they can click the corresponding button to be redirected to that feature.

The format and buttons are clear, with bold subheadings in order to help the customer locate their desired functionality

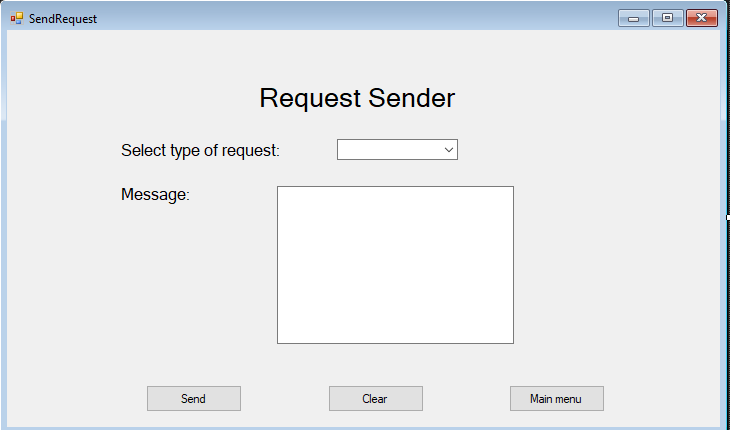
The most frequently used feature will be for customers to send requests to staff members and is subsequently placed in the middle of the form to reduce confusion and increase accessibility. Moreover, a ToolTip has been added for the button, as it’s fairly ambiguous, allowing the customer to understand what can be done with this function. This feature encompasses a plethora of requests including; buy game request, sell game request, quote request, wishlist request and updating of details request.

----------------------------------------------------------------------------------------------------------------

Enter code here when done

----------------------------------------------------------------------------------------------------------------

**Send request:**

****

This form allows the customers to make a range of requests for the staff member to then follow through on. They are required to select what type of request it will be (buy,sell,quote,wishlist,updating of details) along with a corresponding message to compliment their desired request. This will then be sent to the database, allowing a staff member to read and then reply to the request. This feature is essential as it aids the customers to be able to carry out fundamental tasks within just one form such as buying or selling a game. They will then receive a response from the staff member which they will then view under the “View Request Updates” form, shown below.

Overall, the ability to send messages regarding a large scope of topics, is essential in reducing customer waiting times, improving customer experience and improving the efficiency of the business.

----------------------------------------------------------------------------------------------------------------

Public Class SendRequest

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Redirects them to customermenu

CustomerMenu.Show()

End Sub

Private Sub BtnClear\_Click(sender As Object, e As EventArgs) Handles BtnClear.Click

' Clears all fields

txtMsg.Clear()

RequestType.Text = ""

End Sub

Private Sub BtnSend\_Click(sender As Object, e As EventArgs) Handles BtnSend.Click

' Defines new connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Opens connection to the database

conn.Open()

' SQL (Structured Query Lanugage) statement which inserts what the staff member input into the database in the corresponding table

Dim sql As String = "INSERT INTO CustRequests(CustUser,RequestType,Message) VALUES ('" & CustomerLogin.txtUsername.Text & "', '" & RequestType.Text & "', '" & txtMsg.Text & "')"

' Allows the sql command to be executed against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Customer is notififed of their success of their required task.

MessageBox.Show("Request sent", "Sending of request successful")

'Open Database Connection again

sqlCom.Connection = conn

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' Clears all fields

txtMsg.Clear()

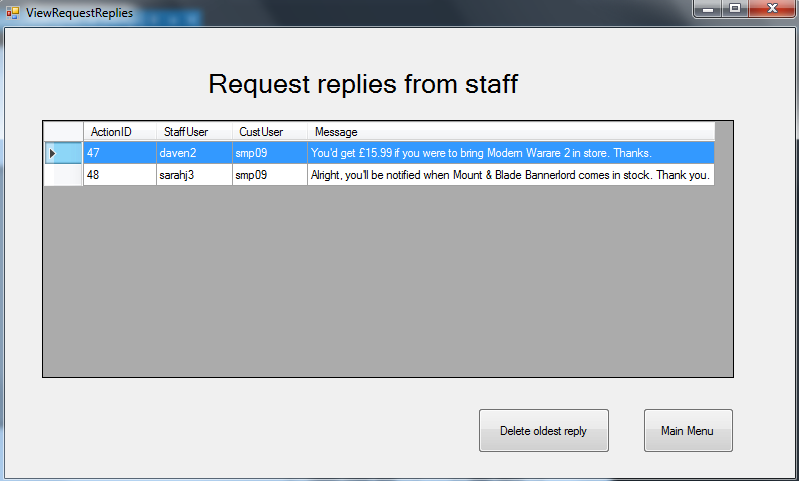
RequestType.Text = ""

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View request replies:**

****

This form will be used by customers who wish to see any advances on the former requests that they made. Once they click the button, all of their replies are shown to them. If they have none, the DataGridView will be empty.

Moreover, they will be able to delete replies from the DataGridView (and consequently the database) if they have acknowledged the response and now wish to focus on other replies or simply to reduce clutter. This is done using the “Delete oldest reply” button, which deletes the reply that they received first in regards to all of the replies as a whole.

This extra feature promotes customers adhering to the oldest replies in order to consequently act on what they have been wanting to do/know for the longest time. Moreover, if they have a further query to the reply they will be able to make a new message using the request sender shown previously, ensuring minimum waiting time for both parties.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewRequestReplies

' Fills data table with data from the database

Public connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Public conn As New OleDbConnection(connString)

' Defines the data table

Public StaffReplies As New DataTable

Private Sub ViewRequestReplies\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

StaffRepliesDataGridView.DataSource = GetCustomerRequests()

End Sub

' Creates function which returns the table to the staff member

Private Function GetCustomerRequests() As DataTable

Using conn

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM StaffActions WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

StaffReplies.Load(reader)

End Using

End Using

' The function returns the data table

Return StaffReplies

End Function

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Shows menu

CustomerMenu.Show()

End Sub

Private Sub BtnDelete\_Click(sender As Object, e As EventArgs) Handles BtnDelete.Click

' Defines variables that will be used to communicate with the database, allowing records to be deleted

Dim StaffRepliesDataSet As New DataSet

Dim StaffRepliesDataTable As New DataTable

' DataAdapter will be used to allow the SQL statement to communicate with the database

Dim StaffRepliesDataAdapter As New OleDbDataAdapter

' Loads all of the tables from the data set to the data table

StaffRepliesDataSet.Tables.Add(StaffRepliesDataTable)

' Define connection to the data source

Dim conn As New OleDbConnection(connString)

' Opens the connection

conn.Open()

' Fetches all records from the table

StaffRepliesDataAdapter = New OleDbDataAdapter("SELECT \* FROM [StaffActions]", conn)

' Refreshes the rows in the DataGridView to match the database

StaffRepliesDataAdapter.Fill(StaffRepliesDataTable)

' Deletes the datarow

StaffRepliesDataTable.Rows(0).BeginEdit()

StaffRepliesDataTable.Rows(0).Delete()

StaffRepliesDataTable.Rows(0).EndEdit()

' Reconciles the changes made to the database

Dim cb As New OleDbCommandBuilder(StaffRepliesDataAdapter)

' Updates the datatable with changes made

StaffRepliesDataAdapter.Update(StaffRepliesDataTable)

' Refreshes the DataGridView to show changes made

StaffRepliesDataGridView.DataSource = StaffRepliesDataTable.DefaultView

' Closes the connection

conn.Close()

' Informs the customer that the record was deleted successfully

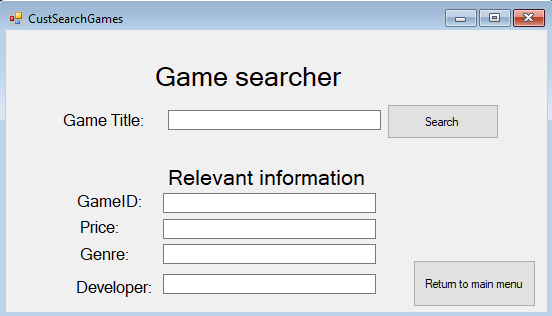
MessageBox.Show("Record deleted successfully", "Deletion of record successful")

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Search for a game on the system**

****

This feature, although already exists for staff members, will also be available for users to utilise. As for staff members, it allows the user member to search for a certain game on the system by its name, where its corresponding information will then be given to them. They will need to input the game’s title, then click the button “Search”, they would then be notified if the game is found or not, and if it is, its information will be inputted in its corresponding text box, improving readability. This is essential for customers specifically as can allow them to search for a game that they would wish to buy, giving them a quick and informative response. This improves customer service and satisfaction, along with better business results, and more importantly, less clutter around the desk asking for if games are available and/or erroneous wishlist requests, for games already in stock, but weren’t aware of their availability.

----------------------------------------------------------------------------------------------------------------

Public Class CustSearchGames

Private Sub BtnSearch\_Click(sender As Object, e As EventArgs) Handles BtnSearch.Click

' Connection is created

Dim conn As New System.Data.OleDb.OleDbConnection()

' Database location defined

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement which returns all of the fields from the Games database where its name is equal to what the staff member searched

Dim sql As String = "SELECT GameID,GameName,Price,Genre,Developer FROM Games WHERE (GameName) = ('" & txtGameTitle.Text & "')"

' Allows the SQL command to communicate with the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Open SQL connection

sqlCom.Connection = conn

'Open Database Connection

conn.Open()

' Executes the SQL command to the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If the game being searched for is found then this

If sqlRead.HasRows Then

' Reads through each field until all returned and then populates the relevant text boxes with each field value

While sqlRead.Read()

txtGameID.Text = sqlRead.Item("GameID")

txtGameTitle.Text = sqlRead.Item("GameName")

txtPrice.Text = sqlRead.Item("Price")

txtGenre.Text = sqlRead.Item("Genre")

txtDeveloper.Text = sqlRead.Item("Developer")

' User is notified of game being found

MessageBox.Show("Game found", "Searching of game successful")

End While

' Closes the read connection

sqlRead.Close()

Else

MessageBox.Show("No game called """ & txtGameTitle.Text & """" & " was found. Please try again", "Error while searching for game")

' All fields are cleared

txtGameID.Clear()

txtGameTitle.Clear()

txtPrice.Clear()

txtGenre.Clear()

txtDeveloper.Clear()

End If

' Closes database connection

conn.Close()

End Sub

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Click

' Hides current form

Me.Hide()

' Shows customer menu

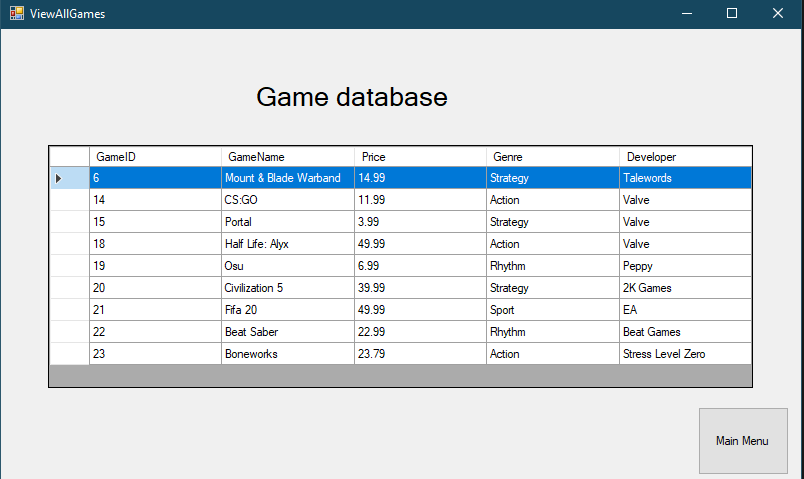
CustomerMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all games on the system**

****

This feature will allow a customer to view all of the games on the system. This is the same function that is given to staff members but customers will also need to utilise this function to be able to browse all of the available games that they can buy, they can then use the game information to make a buy request.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewAllGames

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

Private Sub ViewAllGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

GamesDataGridView.DataSource = GetGameInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetGameInformation() As DataTable

' Defines the data table

Dim Games As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM Games", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Games.Load(reader)

End Using

End Using

' The function returns the data table

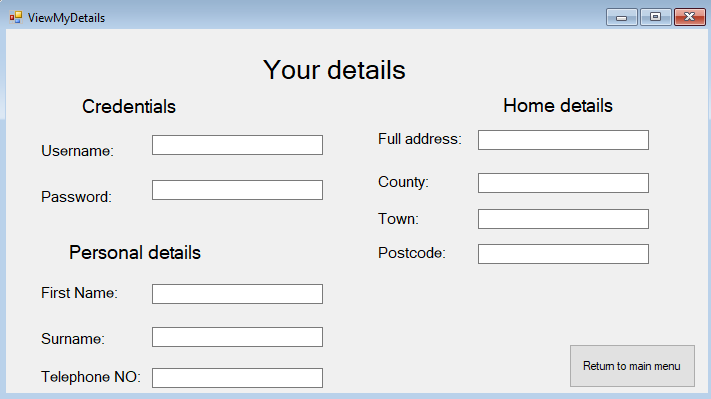
Return Games

End Function

End Class

----------------------------------------------------------------------------------------------------------------

**View current details**

****

This form allows a customer to be able to view all of the details that are stored about them. This works by recognising what account they are logged into and then displaying their information by populating the relevant text boxes. This is important as it complies with GDPR rules, which state that a customer has the right to view or request the information that is stored about them to ensure the information is up to date and not excessive.

Moreover, it will also allow a customer to detect any inaccuracies in the details that are stored about them which they can then report to a member of staff about utilising the request feature, to ensure that their details are accurate and up to date benefitting both them and the data integrity of the database as well as adhering to GDPR regulations.

----------------------------------------------------------------------------------------------------------------

Public Class ViewMyDetails

Private Sub ViewMyDetails\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Retrieves the record of the customer who is logged in

Dim sql As String = "SELECT CustUser,CustPass,FirstName,Surname,PhoneNum,Address,County,Town,Postcode FROM Customers WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

txtUsername.Text = sqlRead.Item("CustUser")

txtPassword.Text = sqlRead.Item("CustPass")

txtFirstName.Text = sqlRead.Item("FirstName")

txtSurname.Text = sqlRead.Item("Surname")

txtTelephone.Text = sqlRead.Item("PhoneNum")

txtAddress.Text = sqlRead.Item("Address")

txtCounty.Text = sqlRead.Item("County")

txtTown.Text = sqlRead.Item("Town")

txtPostcode.Text = sqlRead.Item("Postcode")

End While

' Close connection between sql and database

sqlRead.Close()

Else

' If they have missing details or no details stored about them on the system, they are notified about this and advised to report this to a member of staff

MessageBox.Show("No details could be found about you, please report this to a member of staff")

End If

' Connection of the database is closed

conn.Close()

End Sub

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Click

' hides current form

Me.Hide()

' shows customer menu

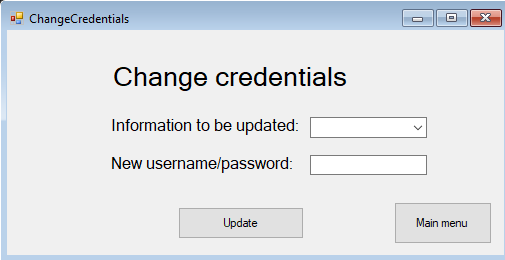
CustomerMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Change credentials**

****

This feature allows a customer to change their username/password by selecting their choice using the drop down menu, which has the values “CustUser” or “CustPass”, so it can be used directly within the SQL statement to communicate with the database. In case of confusion, a tool tip has been added to the textbox informing the user that “CustUser” is their Username and “CustPass” is their Password.

This feature is very similar to that of the “UpdateCustDetails” that the staff have at their disposal, however it’s based specifically on the username and password of the user. Since the username of a customer could be changed many times, they are given the function to change it themselves in order to decrease the influx of Updating of details requests to staff members. Since the password of a customer is a very personal piece of data, it would be inadequate and unprofessional for a customer to send a request to a staff member, stating they would like to change their password to x value and under the case of any malicious actions e.g. shoulder surfing, it could prove detrimental.

----------------------------------------------------------------------------------------------------------------

Public Class ChangeCredentials

' Allows the customerID to be stored here and used throughout the document

' The below procedure retrieves the customerID for the customer that is currently logged in, so this can be used for the SQL statement to change their username/password (as the user desires)

Sub RetrieveCustomerID()

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Retrieves the CustomerID of the customer who is logged in

Dim sql As String = "SELECT CustomerID FROM Customers WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

'Open Database Connection

sqlCom.Connection = conn

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

' Places the corresponding CustomerID in an invisible textbox so it can be used later

txtCustomerID.Text = sqlRead.Item("CustomerID")

End While

End If

' Close connection between sql and database

sqlRead.Close()

End Sub

Private Sub BtnUpd\_Click(sender As Object, e As EventArgs) Handles BtnUpd.Click

' Converts the invisible textbox value for the CustomerID into a real to enable for it to be used in the SQL statement

Dim CustomerID As Integer = Convert.ToDouble(txtCustomerID.Text)

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which changes the value of the desired field to the desired value in the record of the customer currently logged in

Dim sql As String = "UPDATE Customers SET " & FieldName.SelectedItem & " = """ & txtNewData.Text & """ WHERE CustomerID = " & CustomerID & ""

' Executes the statement against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects the sql statement to the database

sqlCom.Connection = conn

' Opens the connection

conn.Open()

' Staff member is notified of their success of their required task.

MessageBox.Show("Field value updated", "Database record updation successful")

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' Clears all text box values to enable them to change other information in rapid succession

FieldName.Text = ""

txtNewData.Clear()

End Sub

Private Sub BtnMenu\_Click(sender As Object, e As EventArgs) Handles BtnMenu.Click

' Hides current form

Me.Hide()

' Redirects to customer menu

CustomerMenu.Show()

End Sub

Private Sub ChangeCredentials\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' When the form loads, the CustomerID is retrieved

RetrieveCustomerID()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

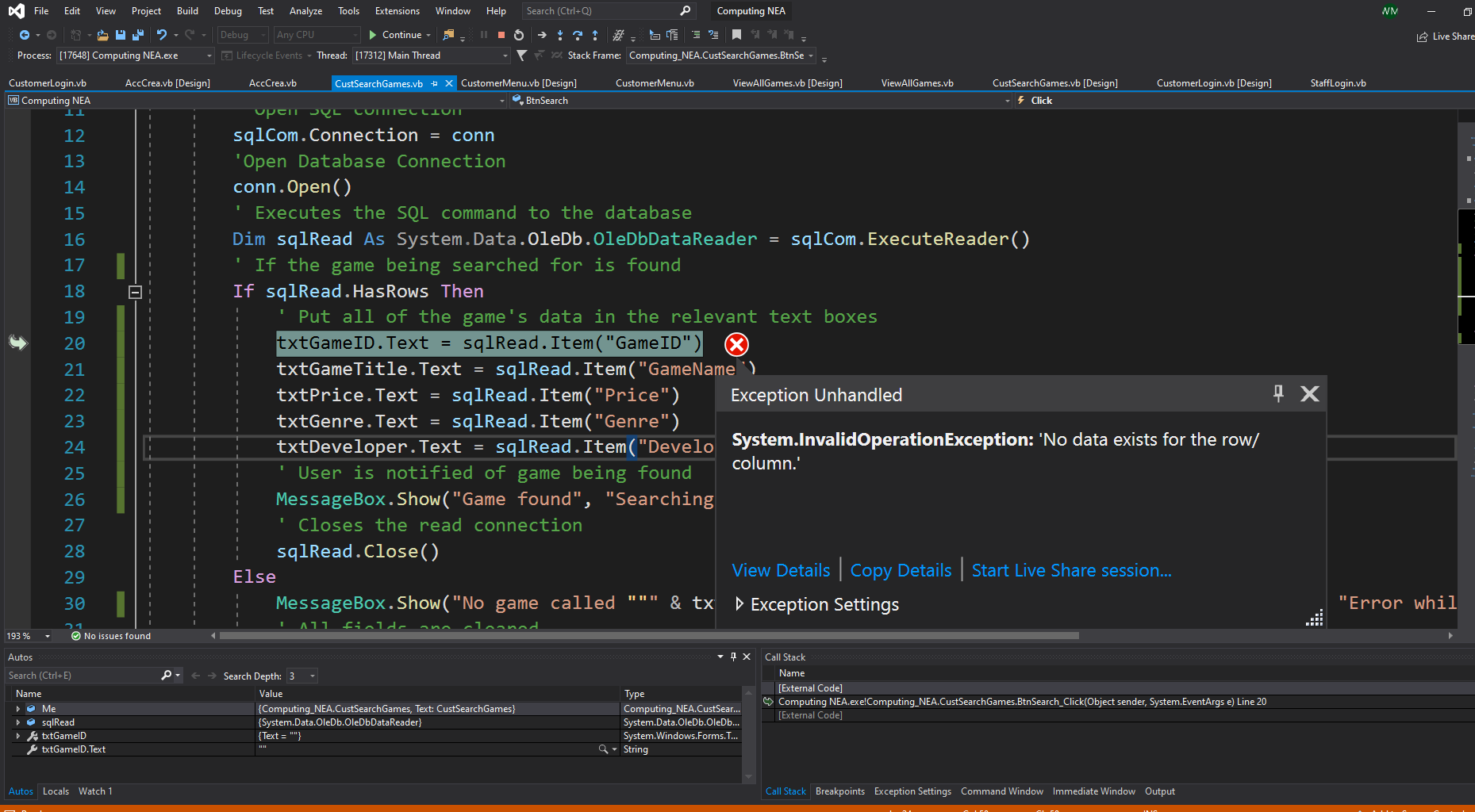
Developmental Testing

Introduction: In this part of the project I will be showing the testing that I have had to do while developing the program for Mongoose Games. I will include screenshots of the problems that I have encountered and the steps that I took to resolve them.

No data exists for the row/column

Whilst developing the part of my system which allows the customer to search for a game by their title, and then if it’s found, its details should be used to populate the relevant text boxes. However, I came across an error shown in the screenshot below. I

I encountered this error when I was attempting to develop a part of the system which produces all of a game’s details when it’s searched by its title, and then these details are populated into the relevant text boxes. This error stated that no data existed for the record that I was trying to fetch.

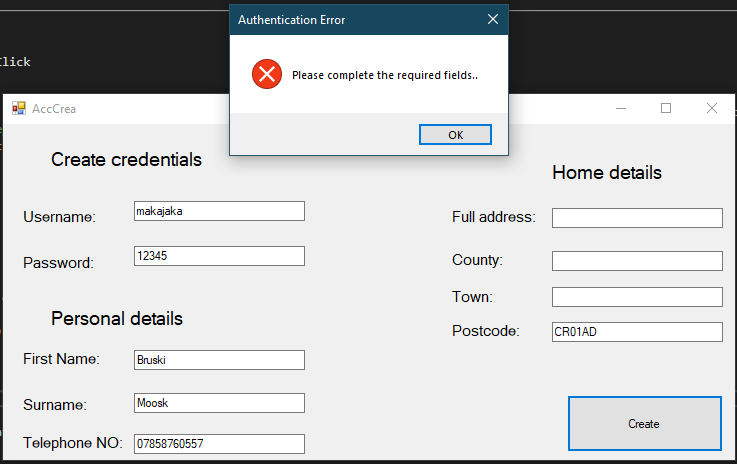


I then corrected this error by realising I needed to add a while loop after the IF clause utilising the sqlRead, to allow each field to be read through and copied into the text boxes until the entire table was searched.

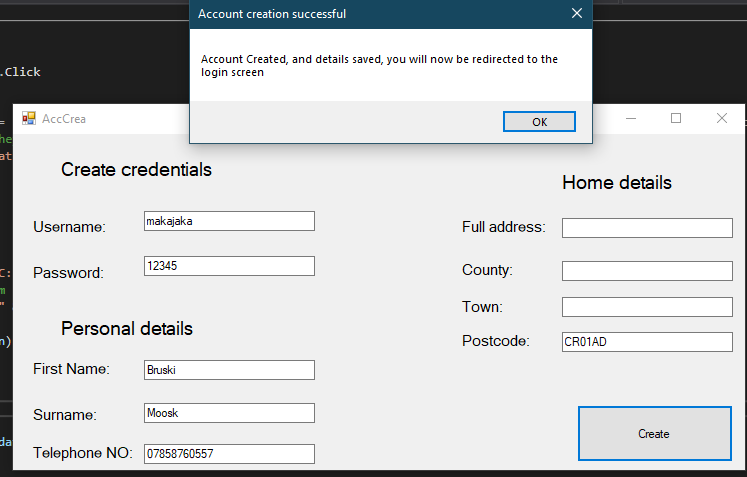
Customer account creation

While testing the customer account creation part of the system, I identified a logic error which was allowing customers to create accounts with missing parts of information.

If a customer would leave out one or more fields of information, an error is then produced which tells them that they are to complete all of the required fields



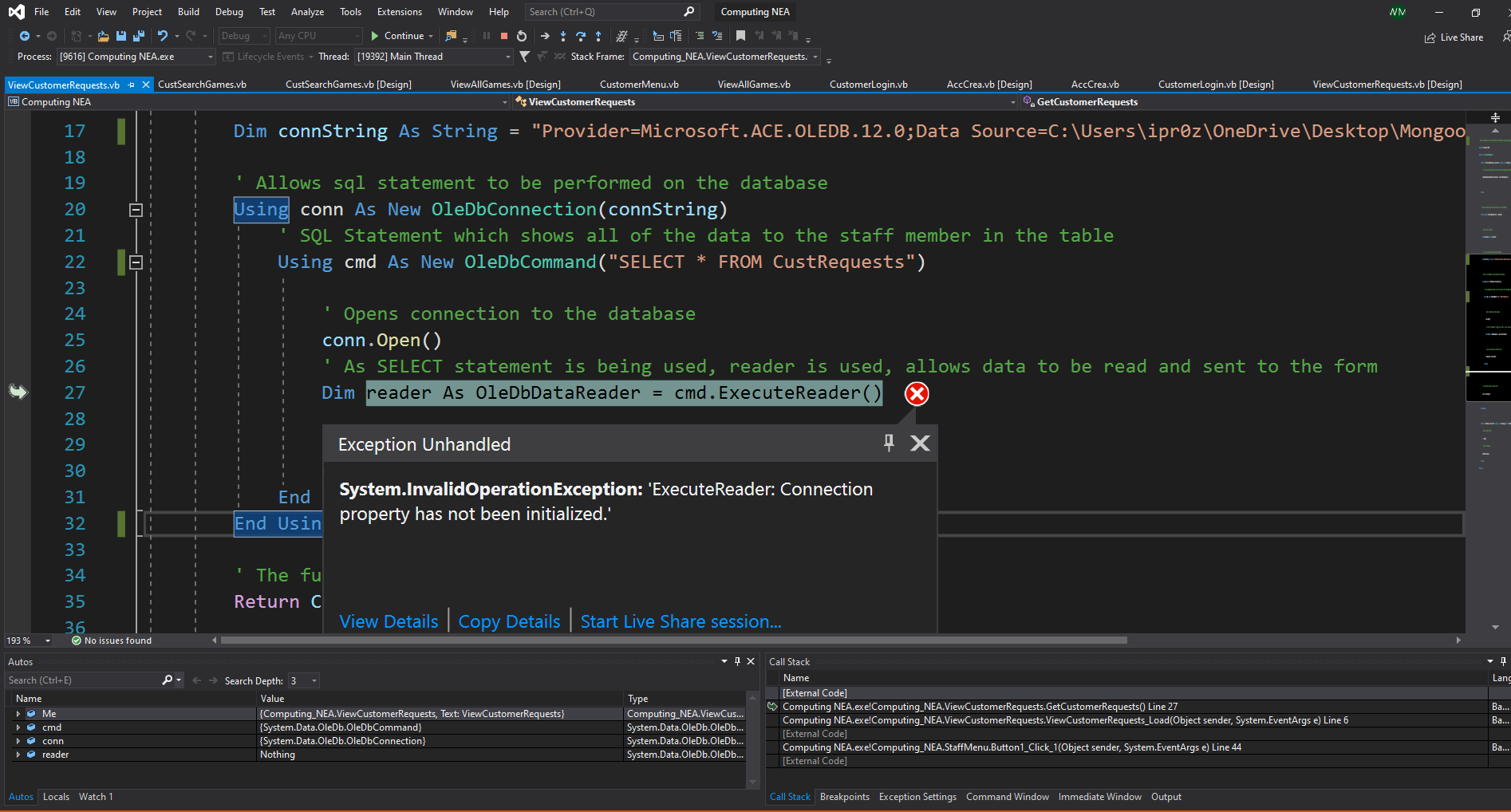
However, the information is still saved to the database.



In the light of this logic error, I then realised I had to use an ELSE statement, so that if they don’t receive an error, their details are saved, but if they do, they cannot create their account until no errors remain.

Connection property has not been initialized

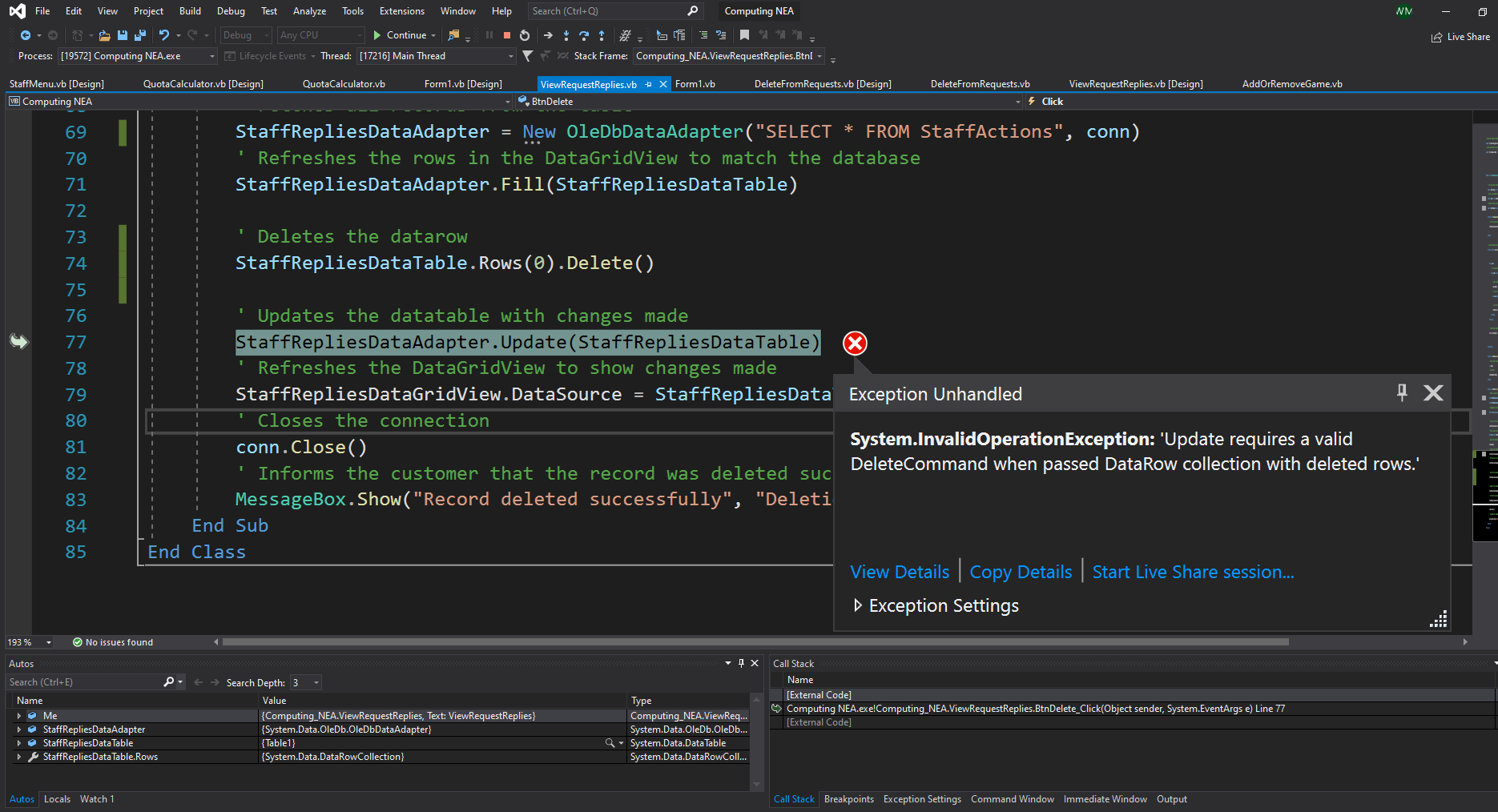
I encountered this error while trying to import a table into a DataGridView, allowing the staff member to view the customer requests from a form. This error stated that the connection property had not been initialized, even though it didn’t highlight the error to the connection string.



I then realised that I had not accompanied the SQL statement with the connection string, to enable the statement to be executed against the defined data source. After doing this, it stopped the error from happening and carried out the function as intended.

Requires a valid delete command

I encountered this error while trying to implement a feature for a customer to be able to delete the oldest reply from a staff member about a request of theirs, via a DataGridView on the form, which would subsequently delete that record from the database. This error stated that, while trying to update the DataGridView after a deletion, it required a valid DeleteCommand.



I corrected this error by realising that I needed to define a commandbuilder, and assign it to the data adapter, to be able to reconcile the changes that had been made to the database. After doing this, the error was no longer produced and allowed the intended function to be carried out.

Testing and Development for user inputs when creating a new account

Currently, there is no validation applied to the data entry of the customers when they create a new account. In order to ensure that only accounts with valid information can be created, their inputs will have to be validated so that no invalid data can enter the system, as well as checking if any fields are left blank.

---------------------------------------------------------------------------------------------------------------------

Imports System.Text.RegularExpressions

Public Class AccCrea

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Check if any field is empty and if any of them are empty

If txtPassword.Text = "" Or txtUsername.Text = "" Or txtFirstName.Text = "" Or txtSurname.Text = "" Or txtTelephone.Text = "" Or txtAddress.Text = "" Or txtCounty.Text = "" Or txtTown.Text = "" Or txtPostcode.Text = "" Then

' If so, then the user is notified of this and they are to then fill them in

MessageBox.Show("Please complete all required fields...", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if the password is the same as the username

ElseIf txtUsername.Text = txtPassword.Text Then

' If so, user is then to ensure that their username differs from their password

MessageBox.Show("Please ensure that your username is different from your password", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the length of the customer's username or password is less than 5

ElseIf txtUsername.Text.Length < 5 Or txtPassword.Text.Length < 5 Then

' If so, they are prompted to make their username/password longer than 5 characters so that their credentials are more sophisticated

MessageBox.Show("Please ensure that your username/password is longer than 5 characters", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the phone number entered is less/more than 11 digits long

ElseIf txtTelephone.Text.Length <> 11 Or IsNumeric(txtTelephone.Text) = False Then

' If so, they are informed that they need to ensure their phone number exactly 11 characters long

MessageBox.Show("Please ensure that your phone number is 11 digits long", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text box to faciliate re-entering of data

txtTelephone.Clear()

' Checks if the postcode entered is 7 or 8 characters long and if it contains a space

ElseIf txtPostcode.Text.Length < 7 Or txtPostcode.Text.Length > 8 Or txtPostcode.Text.Contains(" ") = False Then

' If not, they are informed that they need to enter a valid postcode

MessageBox.Show("Please ensure that your postcode is either 7 or 8 characters long, including a space in between the two parts ", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if any field which shouldn't be composed of digits e.g. firstname contains any digits

ElseIf Regex.Match(txtFirstName.Text, "\d").Success Or Regex.Match(txtSurname.Text, "\d").Success Or Regex.Match(txtCounty.Text, "\d").Success Or Regex.Match(txtTown.Text, "\d").Success Then

' If so, they are prompted to reevaluate this

MessageBox.Show("Please ensure that there are no numbers in fields which don't require them", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

---------------------------------------------------------------------------------------------------------------------

Shown above is all of the code that I have created for the validation for all of the text boxes in the Create account form. I used a large if statement to validate all of the different text boxes in the form. The first condition of the code consists of checking if any of the text boxes have been left blank, and if so, they are made aware of this and prompted to fill out all of the fields.

The next condition checks if the username is the same as the password, and if so, the user is prompted to change this. I did this in order to ensure that the credentials created by the user are more sophisticated, making them harder to guess, improving the security of their account.

The following condition then checks if the username or password is longer than 5 characters, and if not, the user is prompted to change this. This is to ensure that the credentials created are advanced enough to withstand simple brute force attacks.

The following condition checks if the phone number entered by the user is 11 characters long. It also checks if the phone number entered consists of only digits, as it should. If any of these conditions are met, the user is made aware of this and informed to enter a valid phone number. This ensures data integrity.

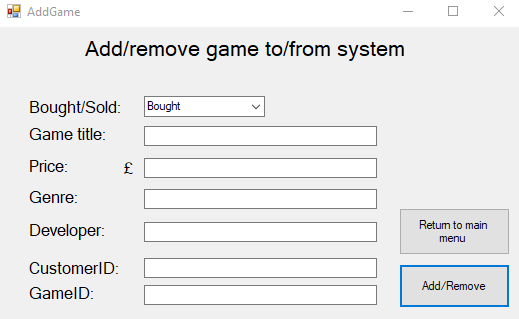
The next condition then checks if the postcode is not 7 or 8 characters long, or if it doesn’t contain a space. If the entered postcode meets any of these conditions, they are notified that they are required to enter a valid postcode, one that has a space in the middle, and comprised of 7 or 8 characters.

Finally, all of the text boxes which should only receive string inputs (Firstname, surname, county, town) are checked to see if the user has inputted any digits in the text boxes. If so, the user is informed of their error and that they need to ensure that text boxes which only warrant a string input, are only comprised of letters.

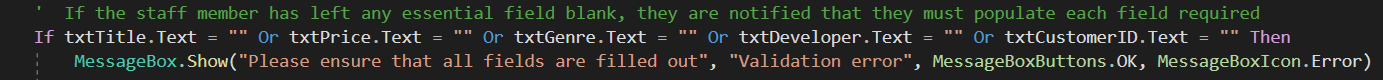
Testing and Development for adding games into the system

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a game is selected as being bought from a customer, but with all other fields left blank, the form just remains blank.



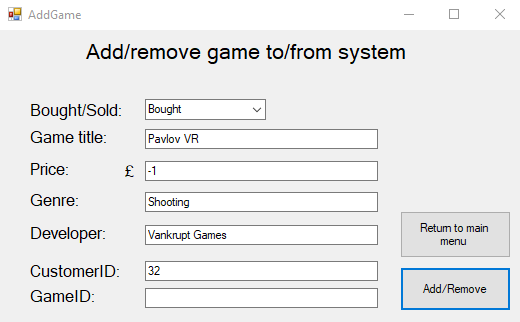
To fix this, I then had to add this code below which ensures that if any essential field is left blank, the game won’t be saved to the database

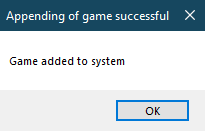


This detects if any fields have been left blank and if so they receive an error message informing them to populate all required text boxes.

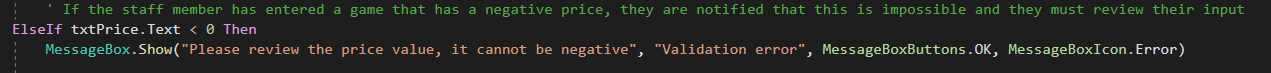
This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a game is selected as being bought from a customer, and the price field is entered erroneously (negative number) it still allows the game to be added to the database with a negative price.





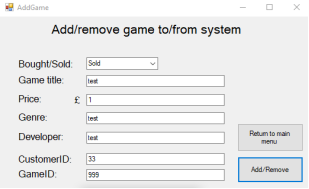
To fix this error, I then had to add this code below which ensures that any price value less than 0 cannot be accepted

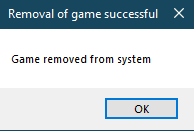


Testing and Development for removing games from the system

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a user chooses to sell a game to a customer which doesn’t exist in the system, an output is received saying that the game was deleted successfully from the system even though it wasn’t nor did it even exist in the first place.





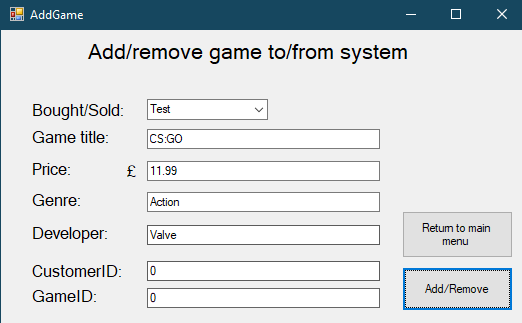
To fix this, I then had to add a condition to the current IF statement that ensured that the game that the user was entering, exists in the database.

...

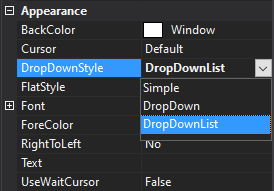
This checks if all of the games information inputted corresponds to an actual game in the system. If so, the process goes ahead and deletes the game from the system.

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a user edits the Bought/Sold dropdown menu, and clicks the Add/Remove button, no error message is displayed, the form just remains blank.



To fix this, I had to ensure that the drop down box values can only be what I have allowed, Bought or Sold. I did this by changing the properties of the ComboBox and changing the DropDownStyle to DropDownList

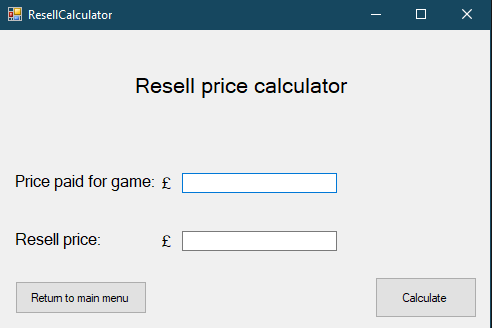


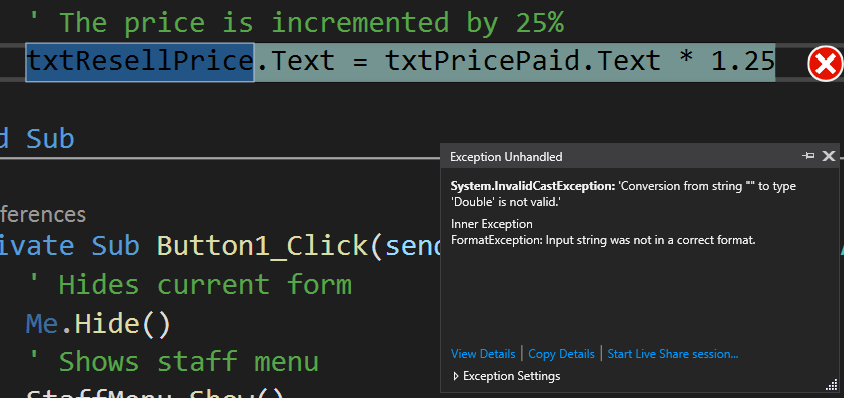
This now ensures that the dropdown box cannot be edited, and so this issue will not reoccur.

Testing and Development for resell price calculator

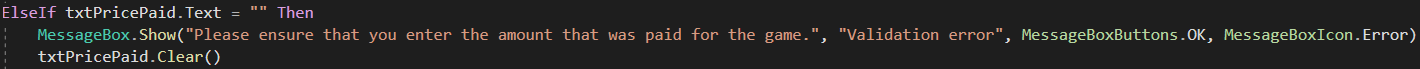
This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a user leaves the “Price paid for game” field blank, and clicks the “Calculate” button, the system crashes as it tries to manipulate the string “” mathematically





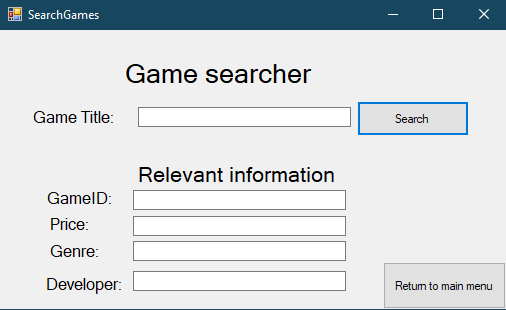
To fix this error, I had to introduce some validation which ensured that the text box holding the price paid value is populated before trying to manipulate the input mathematically.

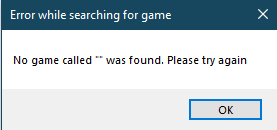


Testing and Development for game search

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently when a user attempts to search for a game, but leaves the Game title field blank, the output is an error message that says no game called “” was found instead of informing the user that the Game Title field cannot be left blank.





To fix this error, I had to introduce some validation which ensured that the text box that receives the game title is populated before trying to search for it.

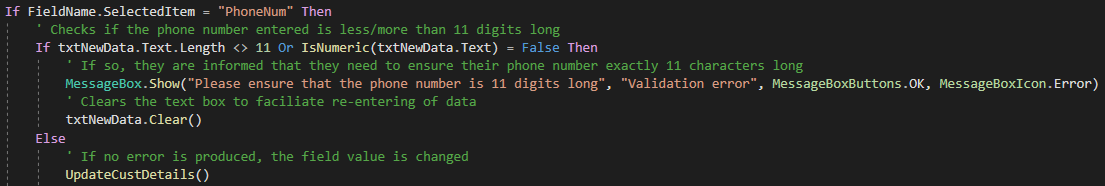


Testing and Development for customer data updater

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

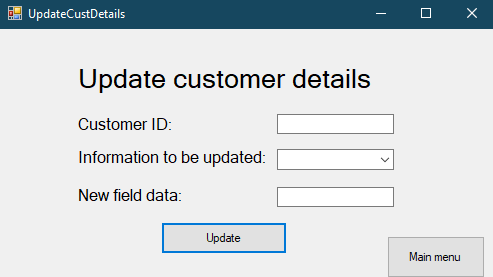
Currently, when a user wants to change a customer’s phone number to an invalid one, they are allowed to.

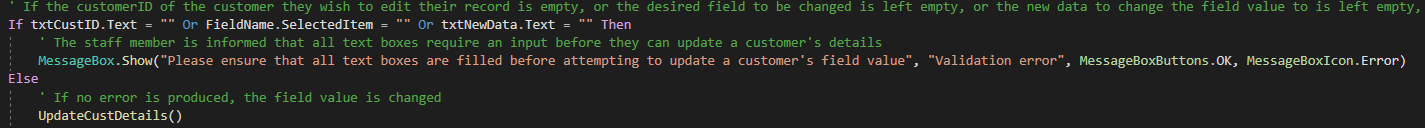
To fix this, I had to include the same validation from the user account creation section, ensuring the phone number is composed of digits and 11 digits long.



This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently, when a user leaves all fields blank, and then presses the update button, nothing happens and the form remains blank.

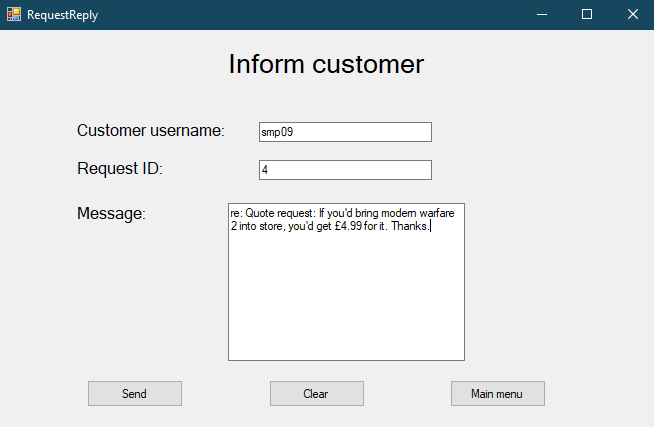


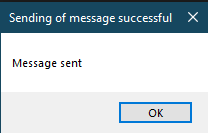
To fix this, I had to include validation that would inform the user if they had left any field blank. Moreover, I made the drop-down box read only, to allow the user to understand they have to choose a field to update.

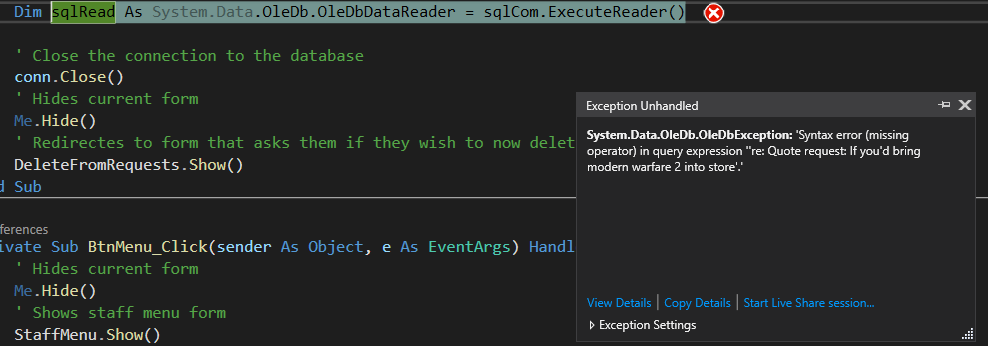
Testing and Development for sending a customer request

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently, when a user wants to reply to a customer request, and the “send” button is pressed, it will tell the user the message has been sent, and then promptly crash. I later found out this has to do with the fact that the message contains an apostrophe.







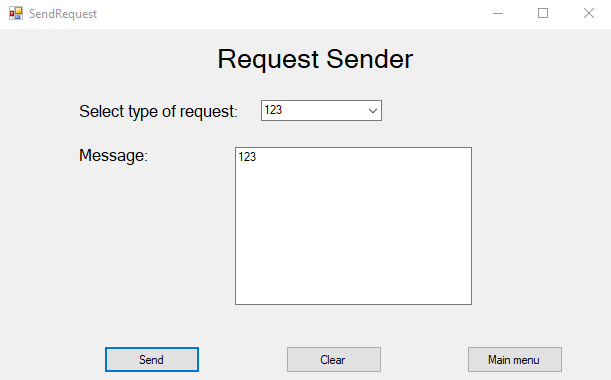
To fix this error, I had to change the SQL statement to be using double quotes “” instead of single quotes ‘’ surrounding the area referring to the data entered into the Message textbox

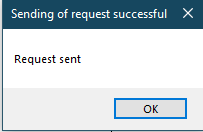


Testing and Development for sending a request to a staff member

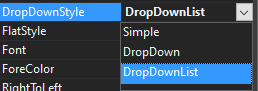
This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

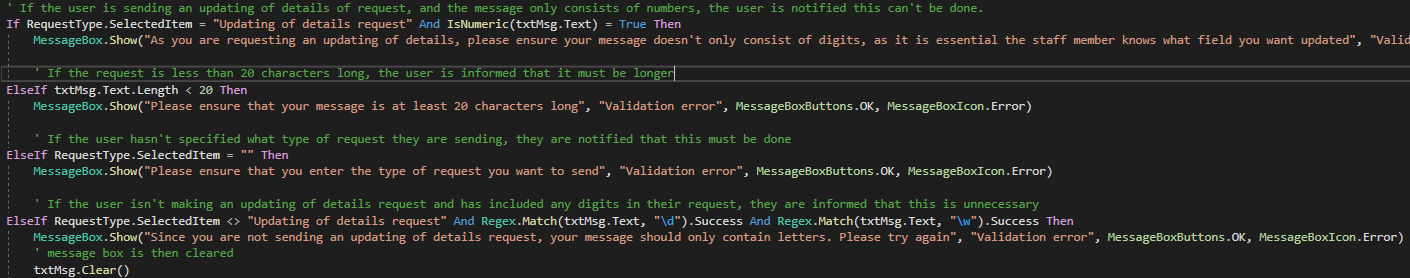
Currently, when a user makes an invalid request to a staff member, the request is still sent.





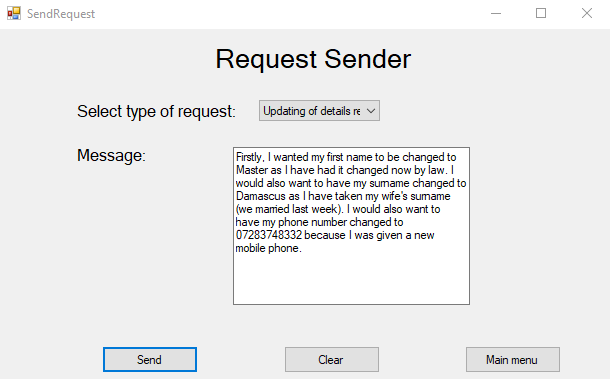
To fix this, I first had to ensure that the type of request dropdown box wouldn’t be able to be edited. I did this by changing the type of the dropdown to dropdownlist. I then had to add validation to ensure that the message entered is more than 20 characters long and that the Type of request was specified using the dropdown menu. Moreover, I added validation to check if the request isn’t an updating of details request, that the message does not need to contain digits, and if it does, the message won’t be sent an if the request is an updating of details of request, that the message must not only be composed of digits.

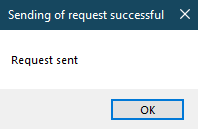


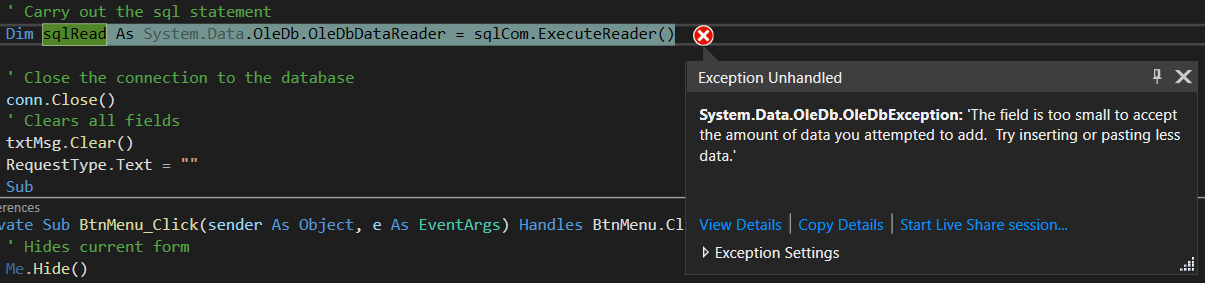


This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

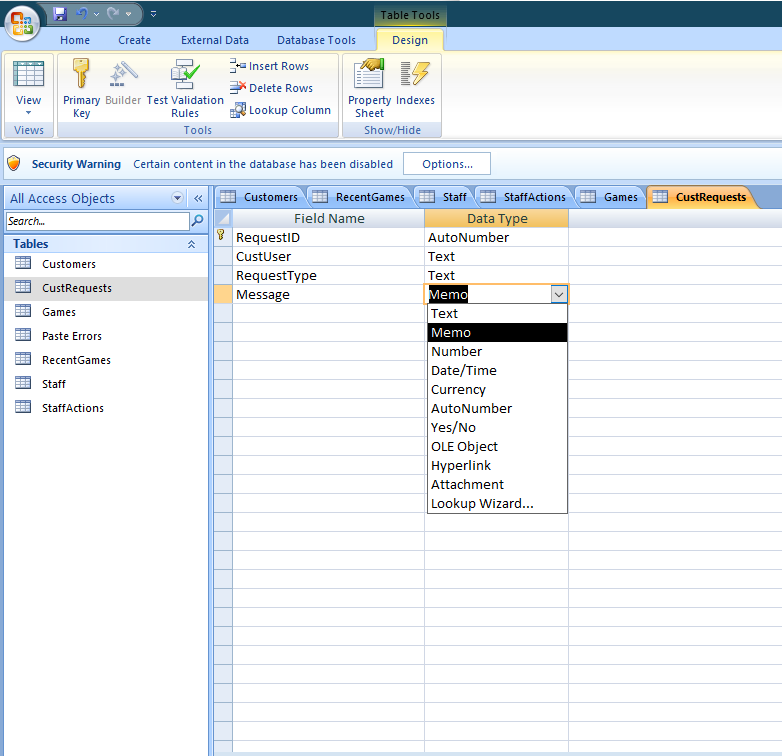
Currently, when a user writes an abnormally long request, the program produces an output wrongly saying that the request has been sent, the program then crashes.







To fix this, I had to change the data type of the field “Message” in the Microsoft Access database. This is the field that holds the requests that are sent by customers. I changed its data type from Text to Memo. I did this because as a string, the maximum length is 255 characters, and as a Memo, the maximum length is 65,536 which sufficiently covers any request that would be sent from a customer.



Testing and Development for change credentials form

This test shown below has been documented in my Test Plan Section of the project. I have identified that the program should have created a different output to the one that has occurred.

Currently, when a user tries to change their password to a value of 2 characters, no error is produced and their password is subsequently changed on the database.

To fix this, I had to add validation to the form to ensure that their new password/username needs to be at least 5 characters before it can be changed. This complies with the validation rules regarding usernames/passwords throughout the entire system.

Testing

Test plan

In this stage of the project, I will be conducting a test plan for all areas of the system. I will use a wide range of testing methods, along with normal (data that the system would normally work with), erroneous (data that is invalid and would produce an error message) and extreme data (data at the boundary between normal and erroneous data). Moreover, I will complement the results of each test with a screenshot along with commentaries, including any specific suggestions to refine the system.

Staff side testing

Login screen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the login screen works by seeing what happens when valid credentials are presented | Entered username: willb4  Entered password: brúsku | Normal | The system should grant the user access into the system | The system logs in the user to the system |  | It worked as intended and no changes were needed |
| 2 | Testing if the login screen works by seeing what happens when invalid credentials are presented | Entered username: waskb4  Entered password:  blyat | Erroneous | The system should display a message informing the user of incorrect credentials | The system produced the expected error message |  | It worked as intended and no changes were needed |
| 3 | Testing if the login screen works by seeing what happens when extreme credentials are presented | Entered username: willb4  Entered password: brúsk | Extreme | The system should distinguish that the the password is not correct and deny access | The system detects that they don’t match |  | It worked as intended and no changes were needed |

Buying price calculator

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the buying price calculator works by seeing what happens when valid data is entered | I will enter days since release date 40, condition as 10, popularity as 10 and RRP as £50 | Normal | The system should return a price value of £37.5 | The output was £37.5 |  | It worked as intended and no changes were needed |
| 2 | Testing if the buying price calculator works by seeing what happens when no data is entered | I will leave all of the fields back | Erroneous | The system should display a message informing the user that they need to enter all required fields | The system produced the expected error message |  | It worked as intended and no changes were needed |
| 3 | Testing if the buying price calculator works by seeing what happens when extreme data is entered | I will enter the RRP of said game to be £-1 and leave other fields as default value | Extreme | The system should still use the input against the algorithm to produce the output of £-13.8 | The expected price was calculated, although an unrealistic RRP was given |  | It worked as intended and no changes were needed |

Resell price calculator

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the resell price calculator works by seeing what happens when valid data is entered | I will enter the Price Paid for the game as £24 | Normal | The system should return a resell price value of £30 | The output was £30 |  | It worked as intended and no changes were needed |
| 2 | Testing if the buying price calculator works by seeing what happens when no data is entered | I will leave the Price paid for game field blank | Erroneous | The system should display a message informing the user that they need to enter a value into the Price Paid field | The system crashed, stating that the input string was given in the incorrect format |  | The test failed as the system crashed instead of producing an error message.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 3 | Testing if the buying price calculator works by seeing what happens when extreme data is entered | I will enter the Price paid for game as £-1 | Extreme | The system should still use the input against the algorithm to produce the output of £-1.25 | The expected price was calculated, although an unrealistic Price Paid was given |  | It worked as intended and no changes were needed |

Adding games into the system

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if adding a game in the system works by entering valid data | Bought from a customer, title rocket league, price £14.99, genre sport, developer Psyonix and CustomerID 30 | Normal | The system should first let the user know that they don’t need to enter the GameID as it didn’t currently exist in the system. It should inform the user that the game has been added to the system and then ask if they wish to log the transaction in the database. | When hovering over the GameID, the user is made aware that input in said textbox is unnecessary when buying a game from a customer as it wouldn’t have been currently existing in the system. When Add/Remove was pressed the system then added the game to the system, informed this to the user, then asked if they would like to log the transaction to the system. After yes was pressed, the user was then told that the game transaction was logged into the system. |  | No changes needed to be made as outputs were as expected. |
| 2 | Testing if adding a game in the system works by entering erroneous data | I will leave all of the fields blank (apart from bought/sold) | Erroneous | The system should display a message explaining that all data fields need to be entered before proceeding | The actual output did not match expected. The form remained blank. |  | The test failed as no error message was displayed to the user.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 3 | Testing if adding a game in the system works by entering extreme data | I will enter the price of the game being £-15 | Extreme | The system should display a message explaining that invalid data has been entered for the price value | The system did not respond as expected. The “game” was added to the system with the invalid information provided. |  | The test failed as the “game” was still added to the system despite the extreme data entered for the price.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |

Removing games from the system

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if removing a game from the system works by entering valid data | Selling to a customer, title rocket league, price £14.99, genre sport, developer Psyonix, GameID 35 and CustomerID 30 | Normal | It should inform the user that the game has been removed from the system and then ask if they wish to log the transaction in the database. | The program did what as expected, and the game was removed from the system successfully and the transaction was recorded to the RecentGames table. |  | No changes needed to be made as outputs were as expected. |
| 2 | Testing if removing a game from the system works by entering erroneous data | I will remove a game from the system that doesn’t exist. Selling to a customer, Game title = test, price = 1, genre = test, developer = test, CustomerID = 33 and GameID = 999 | Erroneous | The system should display a message explaining that the game entered does not exist and thus cannot be sold to a customer | The actual output did not match expected. The game was still sold |  | The test failed as no error the non-existent game was “removed” from the database.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 3 | Testing if removing a game from the system works by entering extreme data | I will edit the “BoughtOrSold” dropdown box to the value “Test” | Extreme | The system should display a message explaining that I must be selling a game to a customer or buying a game from a customer. | The actual output did not match expected. The form remained blank. |  | The test failed as no error message was displayed to the user.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |

View all available games

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a staff member is able to view all available games on the system | I will click the view all available games button | Normal | I expect the form to display all of the games that are available on the system | The system did as I expected |  | No changes needed to be made as outputs were as expected. |

View games recently sold to/bought from customers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a staff member is able to view all games that have been sold and bought | I will click the view games recently sold to/bought from customers | Normal | I expect the form to display all of the games that have recently been sold to or bought by customers | The system did as I expected |  | No changes needed to be made as outputs were as expected. |

Game search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if searching for a game in the system works by entering valid data | I will input the game im looking for (Game Title) as Half Life: Alyx | Normal | It should find the game (as it exists on the system) and then return its relevant details and then populate them in the corresponding text boxes | The program did what as expected, the game was found and its relevant details were displayed to the user |  | No changes needed to be made as outputs were as expected. |
| 2 | Testing if searching for a game in the system works by entering invalid data | I will search for a game that doesn’t exist on the system: Game Title: Half Life 2 | Erroneous | The system should display a message explaining that the game entered does not exist on the system. | The actual output met the expected. The user was informed that the game didn’t exist on the system |  | No changes needed to be made as outputs were as expected. |
| 3 | Testing if searching for a game in the system works by entering extreme data | I will attempt to search for a game with the Game Title left blank | Extreme | The system should display a message explaining that the game name field must be entered before a game can be searched | The actual output did not match expected. The system explained that there was no game called “” |  | The test failed as the incorrect error message was displayed to the user.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |

Customer search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the customer search form works by entering valid data | I will enter the customer username as: smp09, which is a valid customer. I should then be presented with all of their relevant information | Normal | The system should output the details for CustomerID 25 | The expected output was presented |  | It worked as intended and no changes were needed |
| 2 | Testing if the customer search form works by entering invalid data | I will leave the customer username field blank | Erroneous | The system should display a message informing the user that they have left the customer username field blank | The actual output was not as expected. It said “No customer with the name “” found” |  | I have now updated the code to produce the error message shown on the left. |
| 3 | Testing if the customer search form works by entering extreme data | In one instance of the form, I will search for 3 customer’s details: Somnaint, bepis and fishbultz | Extreme | The system should replace the text boxes with the new information each time | The output was as expected |  | It worked as intended and no changes were needed |

View all customers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a staff member is able to view all games that have been sold and bought | I will click the view all customers | Normal | I expect the form to display all of the customers and their information that exist on the system | The system did as I expected |  | No changes needed to be made as outputs were as expected. |

Update customer details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the update customer details form works with valid data | I will enter the CustomerID as 28 (which is a valid customer), the information to be updated as surname and the new field data to be Brúsku | Normal | The system should update the customer’s record with the information provided | The expected output was presented |  | It worked as intended and no changes were needed |
| 2 | Testing if the update customer details form works with invalid data | I will enter the CustomerID as 28 (real customer), information to be updated as PhoneNum and new field data as 12345 (invalid phone number) | Erroneous | The system should display an error message informing the user that the phone number they entered is invalid | The actual output was not as expected. It updated the customer’s record with the invalid data |  | The test failed as the system allowed the user to edit the customer’s details using invalid data  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 3 | Testing if the customer search form works by entering extreme data | I will leave all of the required fields blank | Extreme | The system should produce an error message informing the user that before any process can be made, each field needs to be populated | The output was not as expected. The form remained blank. |  | The test failed as the form just remained blank, without providing any error message.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |

View customer requests

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a staff member is able to view all requests that have been sent to the staff team by customers | I will click the view customer requests button from the staffmenu form | Normal | I expect the form to display all of the customer requests and relevant information | The system did as I expected |  | No changes needed to be made as outputs were as expected. |

Send customer request

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if the send customer request form works with valid data | I will enter the user as smp09, requestID as 4 (real customer request) and then the Message: re: Quote request: If you'd bring modern warfare 2 into store, you'd get £4.99 for it. Thanks. | Normal | The system should send the message to the customer, and then ask the staff member if they wish to delete the original request that corresponded to that reply from the database | The expected output did not match the actual output. Instead, |  | The test failed as the system crashed once the message was sent.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 2 | Testing if the send customer request form works with invalid data | I will enter the following data:  Customer username: neem  RequestID: 10  Message: You will get £15 if you bring GTA V in store | Erroneous | The system should display an error message informing the user that the reply cannot be sent as the original request doesn’t exist | The actual output was as expected. |  | No changes needed to be made as outputs were as expected. |
| 3 | Testing if the send customer request form works with extreme data | I will leave all of the required fields blank | Extreme | The system should produce an error message informing the user that before any the message can be sent, all fields need to be filled | The actual output was as expected. |  | The actual output was as expected. |

Customer side testing

Login screen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will test the system to see if existing customers will be able to login to the system without an issue | Entered username: fishbultz  Entered password: fortnite | Normal | The system should grant the user access into the system as the credentials match. | The system logs in the user to the system |  | It worked as intended and so no changes were needed |
| 2 | Testing if the login screen works by seeing what happens when invalid credentials are presented | Entered username: fishtank  Entered password:  basketball | Erroneous | The system should display a message informing the user that they have presented incorrect credentials and therefore they will not be given access to the system. The text boxes shall then be cleared to facilitate the re-entry of data. They should then be informed of how many attempts they have left. | The system produced the expected error message. |  | It worked as intended and so no changes were needed |
| 3 | Testing if the login screen works by seeing what happens when extreme credentials are presented (password correct but username wrong) | Entered username: fishbultz  Entered password: fortntie | Extreme | The system should distinguish that the the password is not correct and deny access and display the relevant message including their attempts left. The system should also not inform the user that they got one out of the two correct, as this will facilitate brute force entry attacks. | The system detects that they don’t match and denies access as well as informing of the attempts remaining. |  | It worked as intended and no changes were needed |

Account creation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will test the system to ensure that when user enters valid customer details and attempts to create an account, they are able to. | I will enter the following data:  Username: niki2001  Password:  Dutedracu  First name: Nicolae  Surname: Popescu  Telephone No:  07848574554  Full address:  12 Tree Road  County: London  Town: Barking  Postcode: SW12 9PL | Normal | The system should allow the user to create an account and inform them of their success as well as appending this information to the database for reference when logging in. | The system allowed the user to create the account and added their information o the database. |  | It worked as intended and so no changes were needed |
| 2 | I will test the system to see how it responds when a user tries to create an account with invalid data. | I will enter the following data:  Username: niki  Password:  sugi  First name: Nicolae  Surname: Popescu  Telephone No:  12345  Full address:  12 Tree Road  County: 1337  Town: 999  Postcode: SW19P | Erroneous | The system should not allow the user to create the account. Moreover, they should be informed that their username and password need to be over 5 characters, that their telephone number needs to consist of 11 digits, that their county and town must only be composed of letters and that the postcode should be 7 or 8 characters long (according to UK format) including a space in the middle | The system produced the expected error message. |  | The expected outputs were presented to the user and so no changes were needed.  Reference to this validation has been included in the Developmental Testing section of the project. |
| 3 | I will test how the create account form responds when all fields are left blank | I will not enter any data. | Extreme | The system should detect that not all fields have been filled and a subsequent error message should then be produced by the system informing the user that all fields are required. | The system produced the expected error message. |  | It worked as intended and no changes were needed |

Game search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | Testing if searching for a game in the system works by entering valid data | I will input the game I’m looking for (Game Title) as buhnroP | Normal | It should find the game (as it exists on the system) and then return its relevant details and then populate them in the corresponding text boxes. | The program did what as expected, the game was found and its relevant details were displayed to the user as well as notifying that their game was found. |  | No changes needed to be made as outputs were as expected. |
| 2 | Testing if searching for a game in the system works by entering invalid data | I will search for a game that doesn’t exist on the system: Game Title: brúsku | Erroneous | The system should display a message explaining that the game entered does not exist on the system. | The actual output met the expected. The user was informed that the game didn’t exist on the system |  | No changes needed to be made as outputs were as expected. |
| 3 | Testing if searching for a game in the system works by entering extreme data | I will attempt to search for a game with the Game Title left blank | Extreme | The system should display a message explaining that the game name field must be entered before a game can be searched. | The actual output matched the expected |  | No changes needed to be made as outputs were as expected. |

Request search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will test if a customer is able to send a valid request to the staff members. | I will input the following data:  Request type: Buy Request  Message: Hello, I saw you guys had buhnroP in store, I was wondering when I’d be able to come up and make the purchase. Thanks. | Normal | It should allow the customer to send the request, inform them of their success and then add their request to the database. | The program did what as expected, request was sent and saved in the database to allow the staff members to read it. |  | No changes needed to be made as outputs were as expected. |
| 2 | I will test how the request sender form copes when erroneous data is entered | I will edit the request type drop down box to the value 123 and the Message as 123 | Erroneous | The system should display a message explaining that the drop down box value must be on of the determines ones, and the message should be valid | The actual output was not as expected. The request was sent to the database. |  | The test failed as the invalid request was sent.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |
| 3 | I will test how the request sender form copes with extreme data | I will enter a message: Hello, I want some of my details to be updated. Firstly, I wanted my first name to be changed to Master as I have had it changed now by law. I would also want to have my surname changed to Damascus as I have taken my wife's surname (we married last week). I would also want to have my phone number changed to 07283748332 because I was given a new mobile phone. | Extreme | The system should display a message explaining that the message is too long and that it must be shorter. | The actual output did not match the expected. The system produced a message box saying that the message was sent, before crashing. |  | The test failed as the system crashed once the message was sent.  I will include the corrections that I have made to the code in the Developmental Testing section of the project. |

Change credentials

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will test if a customer is able to view all of the replies of their requests | I will click the View request replies button on the Customer menu | Normal | It should show the current logged in user (smp09) all of the replies they’ve received | The program did what as expected, the currently logged in user (smp09) was displayed all of their replies from the database. |  | No changes needed to be made as outputs were as expected. |
| 2 | I will test if a customer is able to delete the oldest reply that they have received from a staff member. | I will click the Delete oldest reply button. | Normal | The oldest reply on the list should be deleted from the customer’s view as well as from the database. They should also be notified of the successful deletion. | The actual output was not as expected. The oldest reply on the Data Grid View was deleted, as well as being deleted from the database. The user was also notified of the success of this process. |  | No changes needed to be made as outputs were as expected. |

View customer requests

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a customer is able to view all games currently available to be bought from the store | I will click the view all games currently available button from the customer menu | Normal | I expect the form to display all of the games that are available to be bought to the customer. | The system did as I expected. All of the available games were displayed to the customer via the Data Grid View |  | No changes needed to be made as outputs were as expected. |

View details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a customer is able to view all of the data that the system has stored about them | I will click the view my details button from the customer menu form | Normal | I expect the form to display all of the information that the system has stored about the user logged in (fishbultz) | The system did as I expected. The user was able to see all of the data stored about them by each piece of data being used to populate the text box labelled as that data. |  | No changes needed to be made as outputs were as expected. |

View details

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No | Test description | Input test data | Test type | Expected output | Actual output | Screenshot | Comments |
| 1 | I will be testing the form to ensure a customer is able to view all of the data that the system has stored about them | I will click the view my details button from the customer menu form | Normal | I expect the form to display all of the information that the system has stored about the user logged in (fishbultz) | The system did as I expected. The user was able to see all of the data stored about them by each piece of data being used to populate the text box labelled as that data. |  | No changes needed to be made as outputs were as expected. |

Acceptance Testing

Introduction: This test section has been created to illustrate client satisfaction regarding how the tests have met all of the requirement criteria that was specified by him. The client has checked all of the tests that I have made created and has subsequently given me feedback in the form of Yes/No answers, which determine if he thinks that each criteria has been met. The table below shows each test as well as whether or not the client believes the requirement has been met (shown by the Yes/No column). Finally, I have also included a comment column which describes in detail how/how not the requirement has been met.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No | Requirement specification | Test Category | Test No | Test result as expected:  Y/N | Requirement met:  Y/N | Comments |
| 1 | The system will be able to calculate selling prices for games using a set algorithm | Resell price calculator | 1 | Yes | Yes | The resell price calculator worked as it should and provides a business suitable value for games that have been bought from customers. |
| 2 | The system will be able to calculate buying prices for games using a set algorithm | Buying price calculator | 1 | Yes | Yes | The buying price calculator worked as it should and has also proved a large improvement from the prototype version where the member of staff had to manually enter the condition and popularity of the game. Now drop down boxes are used for those fields instead which facilitates the process as well as making it more efficient and accurate since it promotes faster, more restricted data entry. Overall, this part of the system functions as it should and calculates business suitable values for the games that are being bought |
| 3 | The system will allow staff members to login using their pre-set logins | Staff login | 1 | Yes | Yes | This requirement for staff members to be able to login their pre-set account has been met as all 3 staff members are currently able to login to the system using their credentials. |
| 4 | Allow customers to login to their existing account or create a new one in order to allow them to access the features that a customer account offers | Customer Login  Create account | 1  2 | Yes | Yes | This requirement for customers to be able to login to their existing account or to be able to create a new one have both been met. If a new customer wishes to create a new account they are required to fill in all specified information and accurately (checked by validation techniques) before they are allowed to carry out the task. This helps to ensure data integrity. |
| 5 | Allow staff members to buy and sell games to/from customers by entering relevant game and customer details | Adding games to the system  Removing games from the system | 1  2 | Yes | Yes | The system was able to add games to the system successfully using the data entry of the Game title, Price, Genre, Developer and CustomerID of who sold it to the store. Moreover, the system was also able to delete games from the system using the data entry of all of the fields specified above, along with the GameID. Both buying/selling transactions of games are logged in the system if the staff member chooses to, or if not in the case of a game being lost or stolen. |
| 6 | Allow the viewing of games that have been previously sold to/bought from customers | View recent games | 1 | Yes | Yes | The ability to view all games that have been previously bought by / sold to customers is very important as it allows staff members to keep track of what games have been sold/bought and to/from who. |
| 7 | Allow staff members to check the quantity of each game in store | N/A | N/A | N/A | Not entirely | The ability to specifically find out the quantity of a certain game on the system has not been implemented. Although, while viewing all available games, if sorted by Name, it can be manually counted how many of one game exists on the database. |
| 8 | Allow staff members to view and update customer information | View all customer details  Update customer details | 1  2 | Yes | Yes | This requirement of the system has been successfully met, when a staff member wishes to view all customers currently registered on the system, they are able to. Moreover, if they wish to update data on a customer record (as per customer request) they are able to, by entering the CustomerID, the field to be updated and valid new data (checked by validation) to replace the old data. |
| 9 | Allow anyone to view all of the games currently available | Staff: View all available games  Customer: View all available | 1  2 | Yes | Yes | Both staff members and customers are able to view all games on the system. This is an essential feature for both parties, as much as for staff members who want to keep track of what games are in stock as for customers who want to look for games to buy. Previously, when Mongoose Games used the paper based system, staff members had to contain a record of all games on the system, hand written. Moreover, customers had to ask staff members constantly what games were available to avoid having to look around the store. Now, both staff members and customers can carry out this task more efficiently and effectively, as well as reducing the amount of customer requests, saving time for staff. |
| 10 | Allow anyone to search through the games currently available | Staff: Game search  Customer: Game search | 1  2 | Yes | Yes | Both staff members are also currently able to search through the games in stock by their title. This is a very important feature for both parties because as the store will become more popular and profitable, more games will be stocked, meaning searching for one game in a database with thousands would be like finding a needle in a haystack. Therefore, this feature facilities the action for both staff members and customers to search for a game. Staff members could do this if they were verifying that a game they added to the system had been appended, or if asked by a rushed customer. Customers could use this if a staff member was busy, to be able to by themselves look for a certain game that they wish to purchase. This improves customer service as waiting times are reduced and customers could feel more independent and self-sufficient to be able to answer their query without asking a staff member. Again, compared to the old paper-based system, it’s a drastic improvement as finding a certain game from the records in the notebooks was heavily time consuming. |
| 11 | Develop the program so that customers can access the system from home via an application to pre-order a game or set up a sale for a game they have | Send request | 1 | Yes | Not fully | This requirement has not been fully met as the ability for users to access the system is not yet an option. I have discussed this further in my evaluation under the Successful features and suggested improvements section. However the parts of the criteria that have been met are the ability for customers to send pre-order requests as well as setting up a sale for their a game they have although they can only do this while being logged into a computer at the store. |
| 12 | Allow customers to receive a quote for their game | Send request  View request replies | 1  2 | Yes | Yes | This requirement has been met and fits under the send request form, where a customer can send a Quote request, specifying what game they wish to sell. They will then get a reply from a staff member telling them how much they would be given, and this would be viewed from the view request replies form. |
| 13 | Allow customers to be able to change their credentials | Change credentials | 1 | Yes | Yes | This requirement has been successfully met as customers are able to change their username/password whenever they wish as long as they know they current username and password, to be able to log into the system. |
| 14 | Allow customers to view all of the information that the system has stored about them | View your details | 1 | Yes | Yes | The ability for customers to have access to their customer record is essential as it allows customers to verify data integrity which complies with GDPR rules. With the paper-based system, it was a much more troublesome task for a staff member to show the customer what details were stored about them, as they had to find their record first on the notepads and had to physically show it to them which was very unprofessional and provides possibilities of a data breach due to physical records being kept of sensitive customer data. |
| 15 | Allow games that are bought from customers to be put up as available to be bought in the system instantly but for a higher price (determined by a set algorithm) | Adding games to the system  Resell price calculator | 1  2 | Yes | Not fully | The ability for games to be instantly put up for sale after being bought from a customer has not been implemented. However, a more manual and methodical way of carrying out this is possible. This involves the following: after a game has been purchased from a customer, you input the price paid for it into the resell price calculator to get a value of how much you should sell it for, you then go to the adding game to the system function and input the required details, and for the price value you enter what was given by the resell price calculator. |

Evaluation

**Introduction:** In this part of the project, I will identify good features and shortcomings of the my solution and I will also describe potential improvements that could be made to improve its effectiveness as well as specific changes of approach that I would adopt in the future to avoid the problems experienced during development. I will also be comparing the system that I have made with a commercially available system. Moreover, I will also describe how I found using visual basic as my programming language and the various different tools and techniques used whilst creating the program, provided by the Visual Studio IDE. Finally, I will evaluate my own strengths and weaknesses that have became prominent during the design and development of the system.

**Evaluation of Visual Basic as a programming language:**

Overall, whilst developing the program in Visual Basic, the intuitivity and simplicity of the language aided me significantly during development. Moreover, the IDE used (Visual Studio 2019) had a wide range of tools available that facilitated the creation of user interfaces and the implementation of event driven programming. For example, the use of forms, buttons, text boxes, grid views, labels, combo boxes and list boxes allowed detailed and user-friendly interfaces to be created.

The solution explorer allowed me to maintain all of the forms and to be able to switch between them with ease, as well as enabling the debugging of the forms individually instead of the program as a whole, to save time and ensure the intended functionality of each form.

The properties tab allowed me to assign meaningful names to the text boxes, buttons and other tools used in the form. This made developing each form easier as it was easier to keep track which textbox held which value e.g. a textbox called txtFirstName. This made the maintenance of the code a lot easier. Moreover, using the properties tab I was able to add ToolTips to certain text boxes and buttons, which would show when hovered over by a mouse. These tool tips would inform the user of what the current tool stores/does to enable a better understanding of the code.

The code for the manipulation of tools was very understandable, and consisted mainly of clicking on a button, and Visual Studio 2019 would instantly create a private sub-program which I would then enter the code for what I would like to happen when said button is pressed. Generally, the communication of the database with the program was simple and the tool DataGridView enabled tables from the database to be read with ease by customers and staff members alike.

Error messages produced by program where highlighted in detail by the IDE, usually giving a clear explanation to what was wrong with the code. This sped up the process of development and quality of life improvements. Due to the Visual Studio IDE series being popular, when an error was not too clear in relation to the course of action needed to fix it, a simple google search of the error usually yielded a plethora of results with subsequent fixes. In addition, in the case of the program having multiple errors, they were all displayed in the Error List and clearly specified which line of code of each form they were in, and if clicked, it would take me to that line. This also enabled an overall faster process of error correction.

Furthermore, I utilised the ability to write comments in Visual Studio 2019 to accompany code which made it more clear and understandable as it defined what each line of code was achieving. These comments have been added throughout the solution, one line before the line of code that is being described and are highlighted in green to promote readability. This was possible by utilising the syntax of the Visual Basic language which involves writing a comment by starting the line with an apostrophe e.g. (‘This code does this).

Debugging tools provided by the Visual Studio 2019 IDE were both useful and in abundance. For example, the use of breakpoints was available which allowed me to pause the code at certain points and examine the state of each variable and to then compare this with the intended value. This assisted me in being able to verify the correct functionality and execution of the program.

In addition, the component of the IDE IntelliSense supported the development of the program greatly. This code-completion tool delivers a great deal of functions such as list members, parameter info, quick info and complete word. These features helped me to further understand the code I utilised, keep track of the parameters used and to add calls to properties and methods with ease.

**Comparison of the system with commercially available systems: (x)**

**Successful features and suggested improvements:**

Successful features of the program include the ability of Mongoose Games customers to set up purchases and sales of games as well as staff members to view these and subsequently attend to their wishes in the store. This will ensure that the interaction between staff members and customers is minimal unless an arrangement has been made which reduces clutter in the store as well as wait times and allows customers to carry out tasks more independently and efficiently.

Another successful feature of the system is the overall improved productivity of the business, highlighted from the change from a paper based system to an electronic one. This is a significant improvement as tasks don’t have to be carried on paper, which negatively affects the structure and professionalism of the business. Moreover, the everyday tasks are completed expeditiously and makes the processes more secure and coherent.

However, in order to make the program more efficient, adding features that are compatible with hardware such as bar-code scanners would diminish the need for inputs of a game’s details that would be identified instantly through the scanning of a bar code e.g. Price. This would in turn enable the program to operate more productively as more customers would be attended to per day regarding buy/sell requests. This would also increase the profitability of the business.

In my test plan I identified that the format of the data should be more checked when a new user is creating an account. However, more thorough measures should have been used to ensure complete validation. For example, it would ensure that their postcode entered is in the format LLN NLL or LLNN NLL as per UK format. This would guarantee data integrity of the database as stricter rules would be placed on data input as well as it would help guide the user to the format required which would also speed up the process for them.

Another improvement to the system that could be made would be to add text-to-speech capabilities. This would enable Mongoose Games customers with visual impairments with the ability to have text narrated to them e.g. to listen to what a staff member has said in their reply to their request or to listen to their original request to ensure it’s comprehensible. Furthermore, a speech recognition feature could be added which would allow disabled users to navigate the system and dictate their inputs. Overall, these features would make the system more inclusive and engaging for a wider range of users.

Moreover, notifications could be added to the system both for customers and staff members which would benefit both parties tremendously. For example, if customers would receive notifications after a game in their wish-list would come in stock, it would be beneficial to both them and the business, as the customers would be more likely to be aware of the game availability meaning it’d be more likely they would buy the game. An option to then customise notifications for certain things would also allow customers to prioritise their interests, if they would be more interested in knowing when their wishlisted game would come in stock than if a staff member would’ve replied to them saying they have updated their details. With this use of notifications, it would save time constantly checking their messages which would save time, improving customer satisfaction. Staff members could also benefit from this if they were to receive a notification once a customer sends a request in, to be able to attend to their request as soon as possible which would allow them to get more work done and improve the overall efficiency of the program.

In addition, a feature for staff members to specifically add new games to the system that would’ve been bought directly from a warehouse. However, although unprofessional, this can be done by using the add/remove game from system form, and choosing to add a game as being bought from a customer, entering a random customerID and then choosing not to record the transaction, likewise this can be done when removing games from the system that have been lost/stolen. Nevertheless, these incidents were not given as much importance as they do not happen on a regular basis and due to time constraints, the general process was prioritised.

Another feature that wasn’t included due to time constraints was the original user requirement of allowing the system to be accessed via an application so they could make use of the program on the move or at home. This wasn’t capable of being implemented into the system as it’s not internet enabled and the data the system works with can only be manipulated on one computer at a time. To enable this possibility in the future there would have to be an online database that would be connected to, and a subsequent compatibility of the program on different devices e.g. mobile, tablet for users to use on the move or if they don’t have a computer at home. However, with this, extra security features would then need to be implemented to ensure the protection of customer data from unauthorised personnel. This is because with the system being internet enabled, it would be open to hackers that could carry out malicious activities e.g. SQL injection.