ToolGator – Portsmouth based tool hire system:

**Discussion**

Background:

Located in Portsmouth on Herald Industrial Estate, ToolGator is a small scale hire warehouse that allows for flexible term loans of tools and construction equipment to tradesmen. During peak times, ToolGator averages 30 customers per hour (8am – 10am) and only a single customer off-peak.

The centre has to monitor all tools currently in their inventory – preventing an overflow of orders for an out of stock item. It also must keep track of the return date of each hired tool, its owner, as well as the pending days until its return. Each tool is inspected when returned and its condition recorded, consequently ensuring that each item conforms to any regulations and is safe to use for the next customer.

ToolGator presently utilises a logbook paper based system to record the status of tools and the log of past hires. Problems arise during busy hours in the warehouse in which if a rental is not written immediately the staff member may forget to record it; creating a discrepancy between the warehouse’s inventory and what is logged so the company may lose track of tools if they are not returned, ultimately costing money. The warehouse’s inventory, rentals, and customer details are all stored in separate logbooks as to keep data organised. With a paper based system the data is vulnerable to damage or theft because all data must be recorded to a physical copy, risking permanent loss if backups aren’t made. In addition, illegible handwriting is a concern since it can be difficult to read at times, potentially casing errors if data is misread.

ToolGator requires a complete modernisation of its tool hire structure, making it an ideal centre for which to base this new system. While it is important to retain prior functionality, the new management program should focus on the aforementioned weaknesses, therefore mitigating the major problems of pen and paper and ensuring that the data is correctly logged with a lower risk of human error.

An intuitive user interface will ensure that employees will know how to operate the system effectively so that the transition to the new software is simple.

Stakeholder Identification – people with a vested interest in the tool hire system:

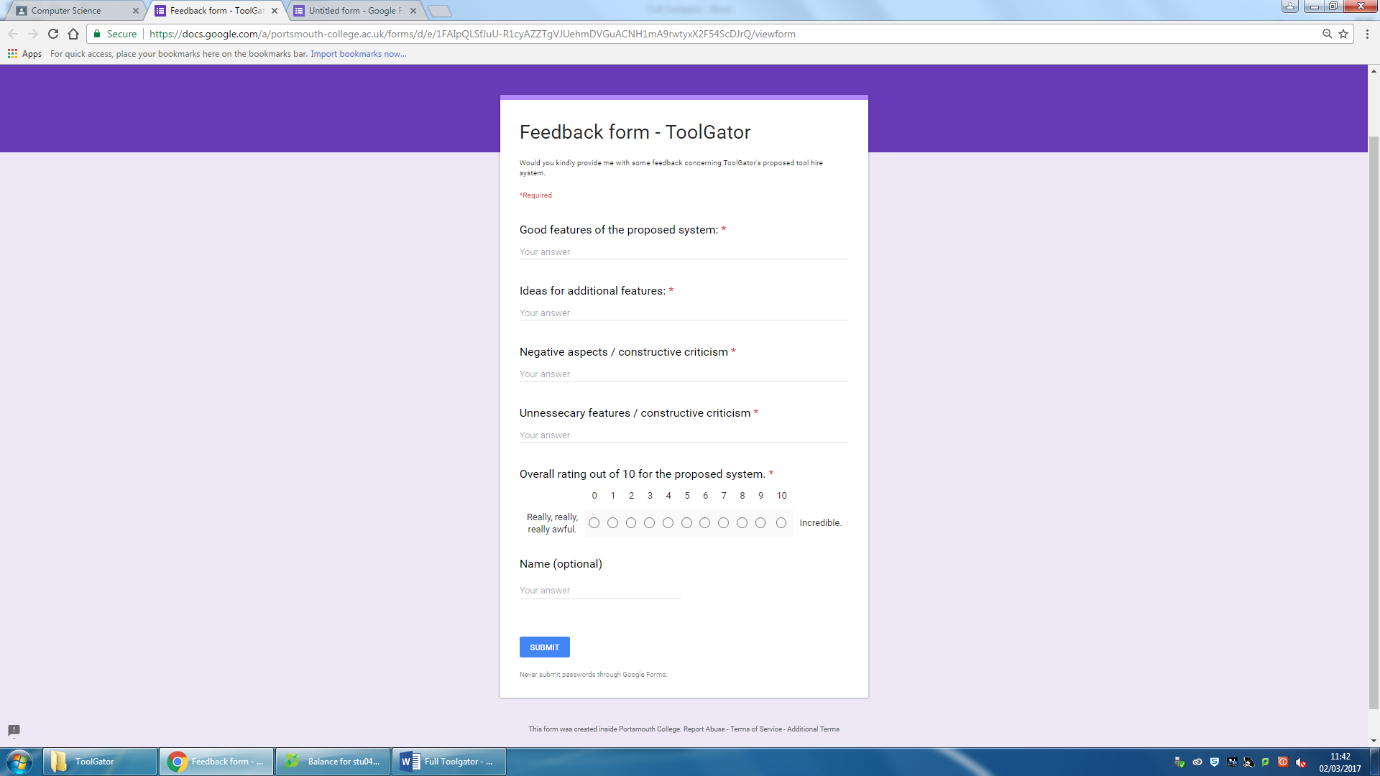
|  |  |
| --- | --- |
| Staff Member | Job Description |
| Nigel Bones – Manager | The manager is responsible for running ToolGator, taking care of staff work times, and adding new stock to the inventory.  If necessary, he can take the role of a staff member. Responsible for buying stock.  The manager would benefit from a renovated system due to fewer errors that may result in lost productivity / profit. |
| David Kirk – Counter Staff | Counter staff is responsible for facilitating customers at the desk. Records customer details in the logbook as well recording each hire in the rental logbook. Keeps the current system up to date.  The hire process should be more efficient with a computerised system, less time should be spent searching the logbooks. |
| Fred Scott – Equipment inspector/Counter Staff | The equipment inspector is responsible for ensuring that all equipment is in adequate condition, carrying out necessary repairs to equipment and noting down additional details in the inventory log. He is also responsible for cross checking the inventory with the logbook on a regular basis to ensure that recorded data is correct. |
| John Riker – Part Time Counter Staff | Counter staff is responsible for facilitating customers at the desk. Records customer details in the logbook as well recording each hire in the rental logbook. |
| Bernard Troi – Part Time Cleaning and Maintenance | Responsible for cleaning of the warehouse and general building maintenance. Has little to no vested interest in the functionality of the hire system - he does the cleaning. |

Presentation to students:

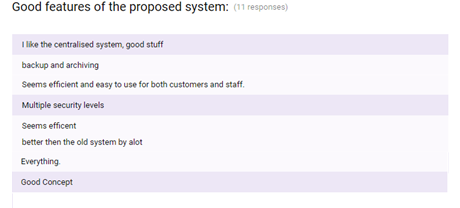
As a prerequisite to receiving feedback and the potential ideas of others, presentation was given outlining the basics of Toolgator’s current system in addition to some features that could only be accomplished by a computerised replacement.

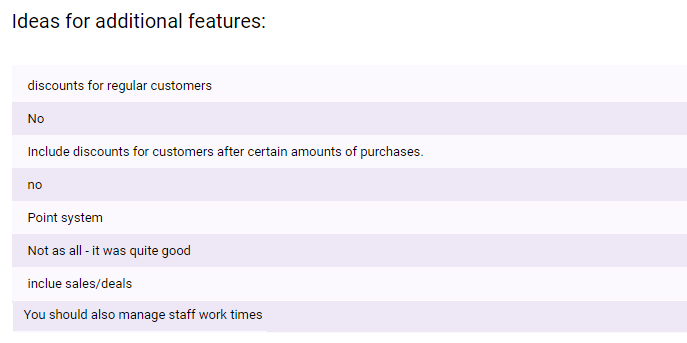


After conducting the ToolGator presentation to other students on the computer science course, the following questions are posed using a Google form.

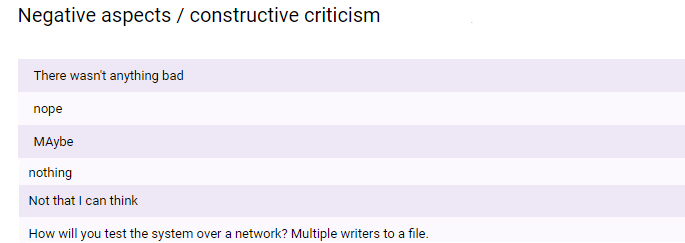


Received Feedback:









Response to Feedback:

Overall, feedback was positive for the new proposed system having received a median 8/10 and a mean 7.81/10.

The posed question “Good features of the proposed system” received encouraging feedback for the features to be debued in the electronic system:

A **centralised system** will certainly be persued with the master files being located on the staff/admin computer and various customer terminals being allowed only read priviliges across the store – providing that the terminal has the file path. This differs from the original strategy to have multiple staff/admin level computers reading and writing to files because I am not equipped with the tools to test a complex system over a local area network; resulting in a compromise to only have a single computer with write priviliges.

The plan for **file backup and archiving** is now highly likely since its commendation in the feedback, it will be included and utilised. Implementation would use the operating system’s data and time to automatically write file backups to another location per user specified time period.

Additional features were suggested, a loyalty points system in which discounts are awarded to regular customers should be included in the program. The **loyalty point** algorithm should be programmable using the GUI by any staff member with an **Admin privilige**. Points will be added to a customer’s details and can be redeemed for discounts. Such a system will enourage customer reliability and benefit ToolGator. As of this time, the approach to which such a system will be implemented is undecided, however, it should take advantage of being in place on an electronic system – adding points automatically on customer transaction.

Furthermore, a point was raised that the new system should manage staff work times – a feature that will, with most likelihood, not be included because it is out of the scope of the system’s objective; to create a system to manage tools and registered customers.

To conclude, the creation of the system will proceed as planned, but also including the suggested loyalty system for customers.

**Summary:**

Feedback highlighted the following points to attention to which I previously overlooked:

* **Inclusuion of automatic backup and archiving** – necessary to mitigate the effects of data loss. Should also include the ability create a backup at the user’s desire.
* **Potential implementation of loyalty points** – an additional feature that may encourage customers to continue using ToolGator. Should be an automated process.
* **Multiple views from customer terminals to central files**. Necessary for the potential for in-store customer computers that can browse the inventory.
* **Multiple user priviliges; admin, staff, customer terminal**. An admin account should be able to aceess all areas of the program including adjust pricing/loyalty point algorithms. Staff can take out and return tools for customers. The customer terminal can only browse the stock.
* **Encryption** of customer and staff details. As with any computer system connected to the internet, there is a risk of malicious attacks with the intent of stealing information. Although a firewall and antivirus should be in use, the encryption of customer details will provide an additional layer of protection if the files are accessed.
* **Customer details should be deleted from the system if they have not made a transaction within the span of a year.** In accordance with the Data Protection Act, customer data should not be retained for longer than necessary, however, no exact time period is specified.
* Limitation - System operates on a local level due to lack of equipment to create a networked system – only one computer can write to the central files.
* Limitation – No protection on file theft. Despite the fact that encryrption is planned to be in place, files may still be copied by an employee.

**Investigation**

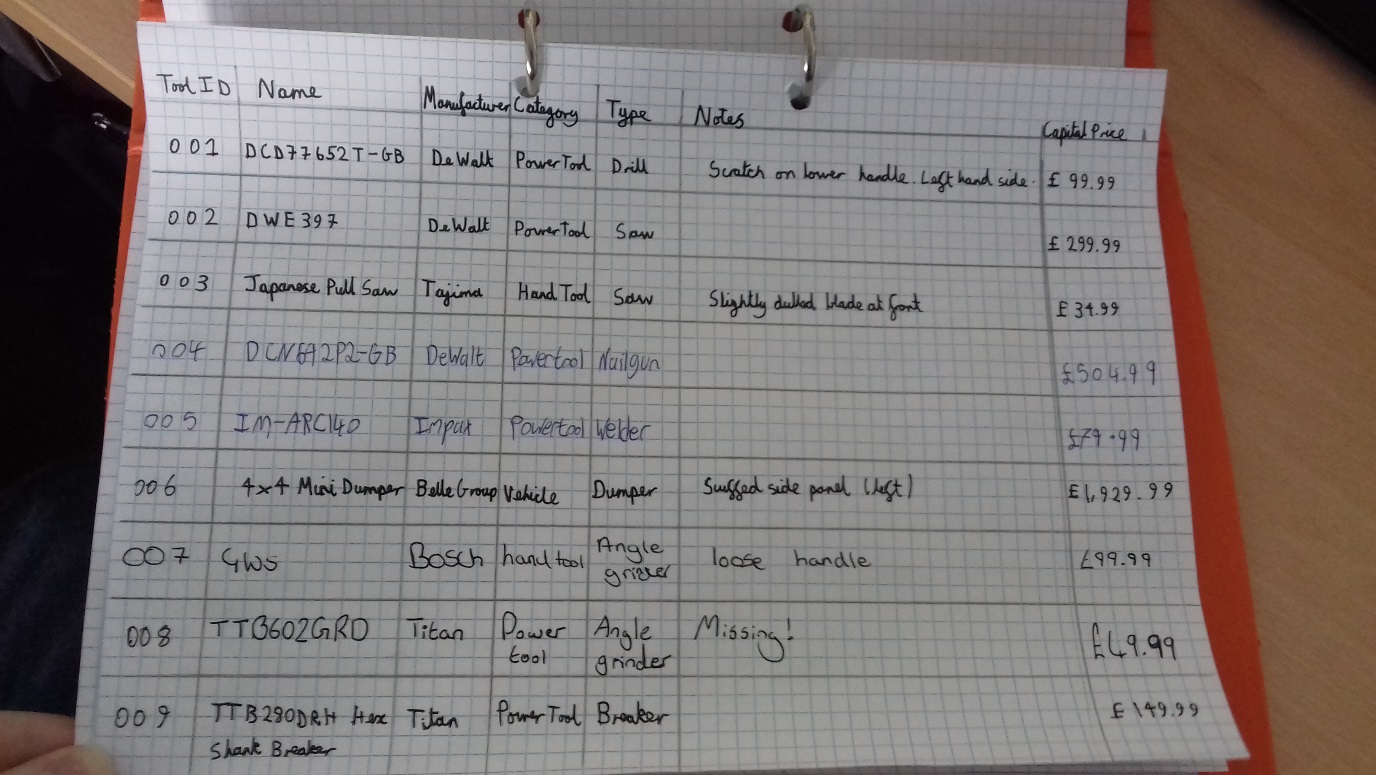
Analysis of current system:

In order to gain a greater understanding of the operation of the current system, a wide scale investigation was conducted. It includes a document analysis, staff observation, and staff a interview.

Document Analysis:

Presently, ToolGator utilises a paper based log to manage the hiring of tools. Each individual tool has a record and a key in the inventory log. Hiring is facilitated though a separate logbook, referencing the tool’s ID, customer ID, and rental dates. A third logbook is used for recording customer details. Using this process, data is stored in an organised manner with each logbook containing only relevent information – minimising data redundancy and speed at which staff can locate a record.

Regarding tool maintenance, any problems with an item are noted in the inventory log in the corresponding record. Amendments/erasure of the notes is carried out using correction fluid – over time leading to wear and tear of the paper.

*Figure 1: The Inventory log:*

The first page of the Inventory Log has the fields:

* Tool ID (primary key)
* Name
* Manufacturer
* Category
* Type
* Notes
* Capital Price (how much the tool was bought for).

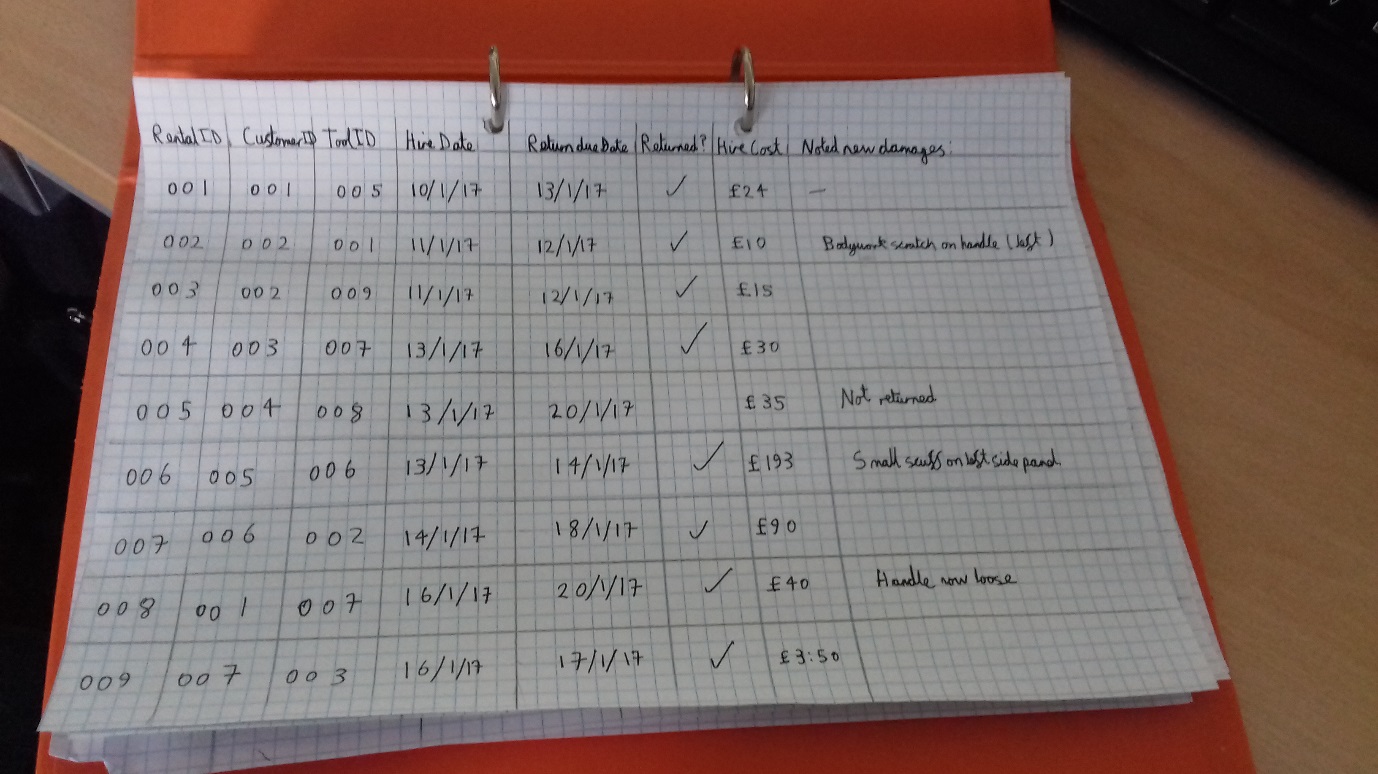
The Inventory Log stores all the data regarding each individual tool with the unique ToolID allowing for efficient referencing capability in other logbooks (see *figure 2*).

Records are created in the Inventory Log upon the addition of a new tool to the stock. Each entry must be hand-written, potentially taking a long time as well as increasing the potential for human error which is common for monotonous tasks. **In the computerised replacement, the creation of unique records for the same tool will be automated, the user should only be required to enter some fields (Name, Manufacturer, Type, and capital Cost), Tool ID will be generated based on the current last record.**

Short descriptions are added to a tool’s notes field if it is returned with any new damages. With time, problems with this system arise as old issues are fixed and new issues are found; leading to an illegible mess of amendments and correction fluid. Physical space in the field is also an issue, large handwriting or multiple notes would fill the box. The proposed computerised system has none of these problems, the length and amount of notes should be as long as nessecary with no imposed limit. **Staff should also be able to amend/erase any notes with ease unlike with the logbook.**

Evident in *figure 1* are long tool names, the majority consisting of a difficult to remember string of numbers and characters. If Toolgator were to increase the amount of a tool already listed in the inventory, there is the probability that the staff member re-writes the name differently than before, reducing overall data integrity. **To rectify this, the computerised system should have the ability to suggest a tool’s name based on prior additions to the inventory.**

There is the possibility that at least one of the tool’s descriptive fields (Manufacturer, Category, Type) is filled in incorrectly, as evident in *figure 1* ToolID 007, an angle grinder is not a hand tool. The previous idea would partially solve this problem because the tool name would not be suggested if a field input is incorrect (if adding more of a pre-existing tool to the inventory). **In the computerised system, tool records should be open to amendment.**

*Figure 2: The Rental Logbook*

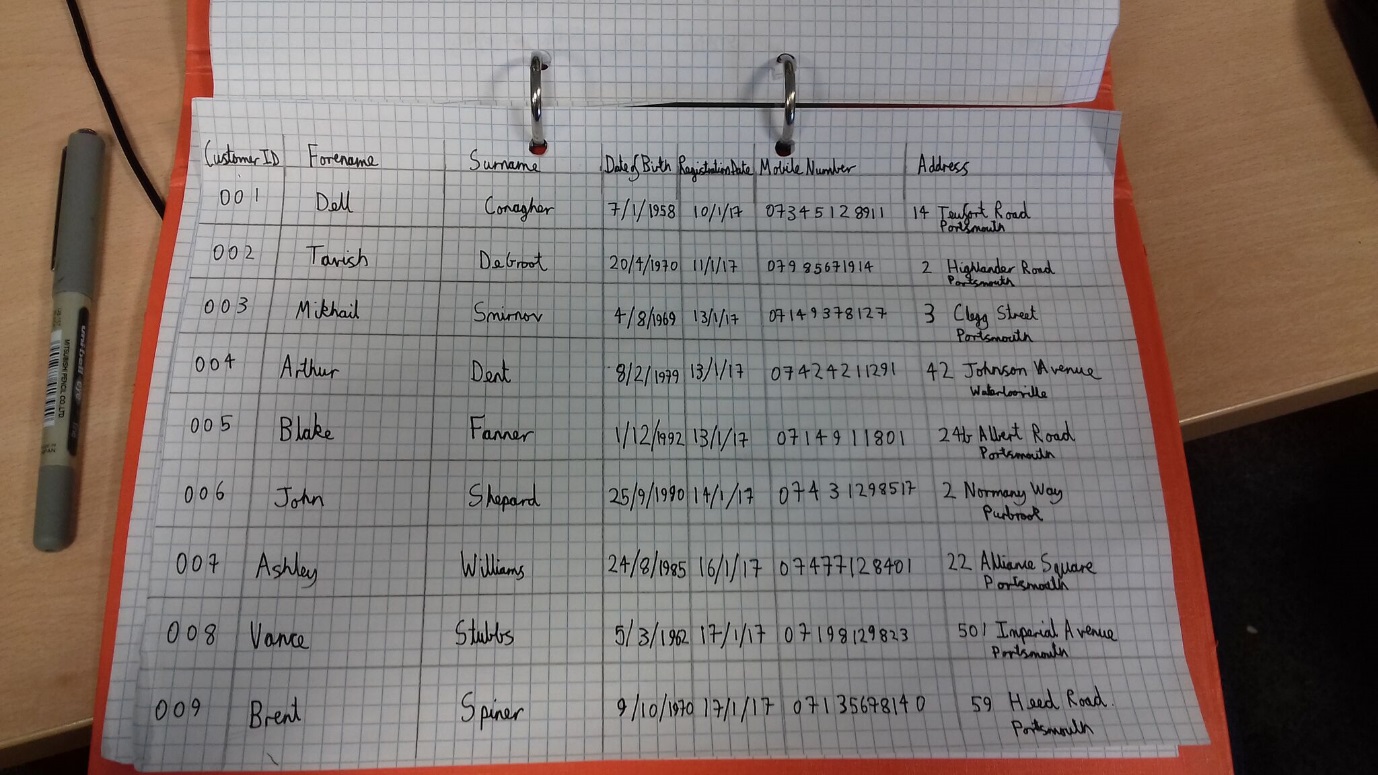
The Rental Logbook contains the fields:

* Rental ID (primary key)
* Customer ID (foreign key)
* Tool ID (foreign key)
* Hire date
* Return due date
* Returned
* Hire cost
* New Damages

Foreign keys Customer ID and Tool ID are used to reduce data redundancy and mitigate integrity loss in other logbooks. A unique rental ID is assigned to each hire transaction, noting the details in the record.

In this page, the hire cost is calculated by 0.1 x tool capital cost x days rented, using the difference between the hire date and return date as well as the capital cost from the ToolID foreign key as the basis for the calculation. The manager can change the hire cost procedure at whim, **therefore the computerised system should have the ability to utilise multiple hire cost calculating algorithms.**

Upon the return of a tool, a tick is placed in the return field, ensuring staff can keep track of overdue equipment as well as the client in question. **The proposed system should automatically make a note of customers who do not return tools within their alloted hire period.**

*Figure 3: Customer Log*

The customer log contains the fields:

* Customer ID (Primary Key)
* Forename
* Surname
* Date of Birth
* Registration Date
* Mobile Number
* Address

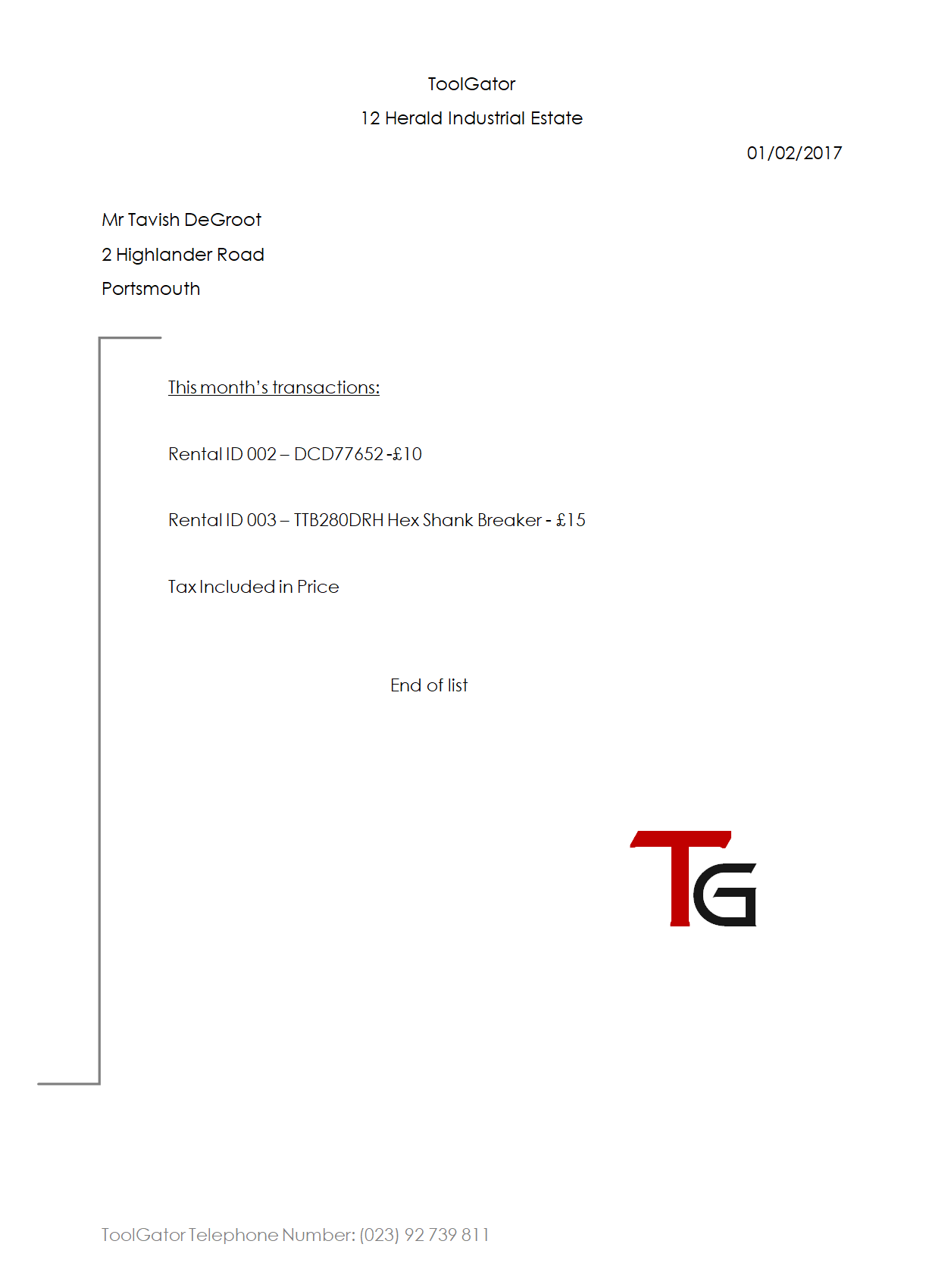
All customer details are recorded in the Customer Log. Customers are tied to each hire event, allowing staff members to easily see who is responsible for the tool and liable for any damages.

Data of Birth and the customer’s Registration Date is recorded for security purposes; customers must state their name and date of birth before any rental. In addition, all customers have a unique record in the Customer Log. Clients must have already registered before they hire any equipment. **The new system should have the ability to view customer details and require the customer to be registered before any rental transactions take place.**

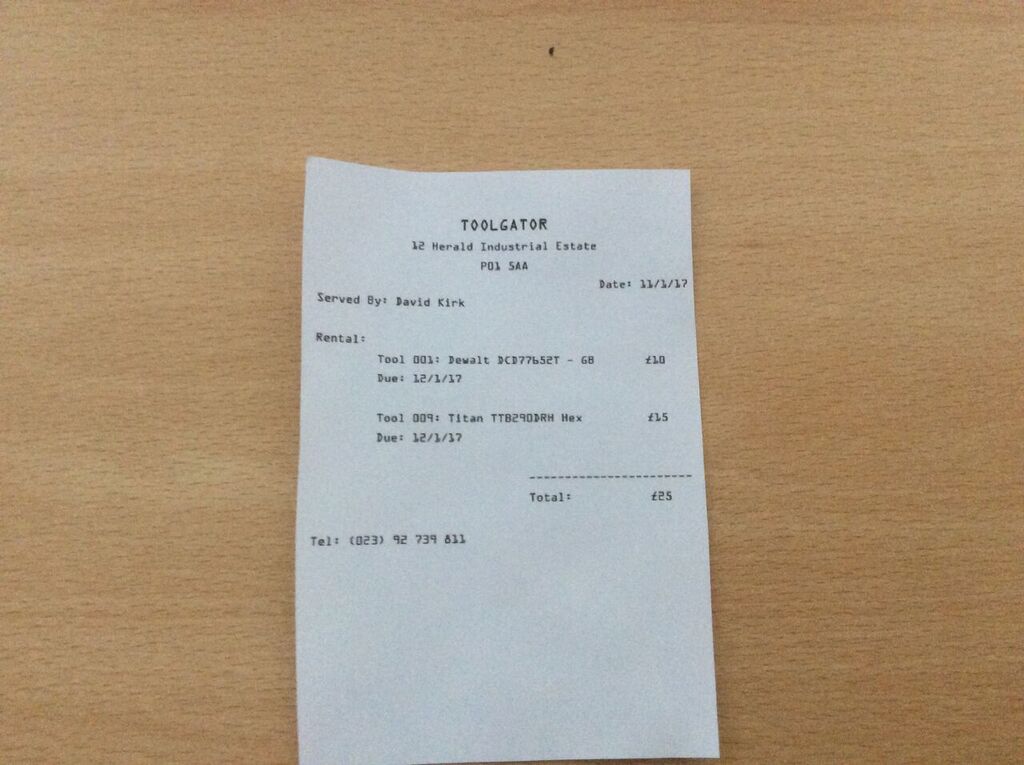
The mobile number and address is used for contact purposes, clients with overdue equipment are contacted by a member of staff to enquire about the state of the tool and its soonest possible return date.

*Figure 4:*

Letters containing the information of each personal rental transaction are sent to registered customers on a monthly basis.



Presently, statements are hand typed by staff members at ToolGator (see Figure 4). It features a list of all tools taken out by a single customer over the course of a month; showing its name, rental cost to customer, and tool ID (useful for ToolGator staff). The manual creation of statements is time consuming and increases the likelihood of errors, therefore, **the new system should have the ability to generate statements** **for each customer.**

*Figure 5: Tavish DeGroot’s receipt.*

Presently, ToolGator uses its cash register to produce receipts for customers. This system functions adequately and, as the new system does not plan to include payment, will probably remain unchanged.

**The new system will make use of the old cash register and receipt machine, remaining unchanged.**

Interview:

Prior to the interview, a list of questions were created to warrant the relevancy of the interview – ensuring that the interview did not go off tangent with an excess of follow up questions. Despite this. Open questions allow expanded answers.

The following are questions asked in an interview with manager Nigel Bones, it was recorded and the transcript written:

**Me:** Firstly, talk me through the process when a customer wishes to rent or hire a tool.

**Nigel:**  Right, a customer will look through our in-store catalogue for a tool. Then he’ll come up to the counter and ask for the tool. One of our lads at the counter will then take a look in the rental log for that tool and see if it’s taken or not.

**Me:** Go on…

**Nigel:**  If it’s in stock, he’ll run the costs through a calculator for however long the customer wants to take out the tool. He’ll then find the tool in the back and make a new record in the rental log.

**Me:** Is that all?

**Nigel:** Yea… no. I forgot the first part. First thing is that we check if the customer’s come here before and look in the customer log before we let him take anything out. New customers have their contact details taken and put in the customer log, stops people running off with all our tools you see.

**Me:** Thanks. So, what happens when a tool is returned?

**Nigel:** Yeah, we give it a tick in the rental logbook and note down any damages the customer tells us about. We also write down the tool’s return date.

**Me:**  And what if the customer doesn’t tell you about some of the damages?

**Nigel:** Of course we give it an inspection and write down anything extra.

**Me:** About the return date, what if the tool is returned overdue?

**Nigel:** We usually just charge the normal rate, people tend to return things on time anyway. I think a tool wasn’t returned once, but the guy told us that the tool was stolen. Our insurance covered it.

**Me:** So you don’t have an overdue policy?

**Nigel:** No.

**Me:** You send invoices to customers, talk about the process.

**Nigel:** Every month we have to type up invoices andsend those to customers by post, of course, only if they’ve bought anything in the month. It takes quite a bit of time.

**Me:** So you’d benefit from an automatic invoice writer?

**Nigel:** If you could do it lad.

**Me:** When it comes to adjusting details, how would you go about doing that?

**Nigel:**  We usually just use some correction fluid, squirt over the wrong bit and make the amendment. Works fine at first, but it builds up over time, damaging the logbook. We usually note this down on a sticky note and get around to it as soon as possible.

**Me:** That’s another issue then, you’d like the ability to amend the data.

**Nigel:** Yeah, amending the data would be important.

**Me:** Is that all?

**Nigel:** Don’t bother asking for the customer’s date of birth to check if they’re registered, they don’t like it.

**Interview Summary:**

Based on the information gathered in the interview, some features are to be re-affirmed:

* **Invoice generation –** Nigel talked about how invoices are manually created.
* **Changeable rental price algorithm.**
* **Calculating an overdue fee.**
* **Ability to amend tools and customer details without system disruption.**

Observation:

In order to ascertain the practicality of the current system, an observation was carried out of various tasks that take place during the day.

**Task 1: Serving a pre-registered customer:**

* A returning customer (Dell Conagher) enters the shop and browses the catalogue for the tool he would like to hire, deciding to take out a GWS Bosch angle grinder.
* When he approaches the counter, staff member David Kirk asks him if he has been here before. Since he has, David asks for his full name and date of birth to verify if it is him.
* After verification, counter staff searches for the tool in the inventory log to find its ID, then enters a new record in the rental log, referencing both customer and tool ID.
* The customer pays upfront for the duration of the hire, either by cash or card, facilitated by the counter staff.
* The customer receives the angle grinder and is given a receipt with the due date printed on it.

**Task 2: Serving a new customer:**

* A new customer (Ashley Williams) enters the shop and browses the catalogue for the tool she would like to hire, deciding to take out a Tajima Japanese pull saw.
* When she approaches the counter, staff member John Riker asks her if she has been here before. Since she as not, she is asked for her name, date of birth, address, and mobile number to add to the customer log; allowing for her to be mailed monthly statements and to be called if she does not return tools on time.
* The customer pays upfront for the duration of the hire, either by cash or card, facilitated by the counter staff.
* The customer receives the angle grinder and is given a receipt with the due date printed on it.

**Task 3: Returning a tool:**

* The customer returns to the counter with the tool to be returned.
* The counter staff asks for the customer’s name, date of birth, and address. The tool name is also asked. With this, the appropriate tool ID can be found via the customer log using the customer ID, looking in the rental log, and then the inventory log.
* The returned field is ticked in the inventory log and the tool is returned to the warehouse. Any damages are noted in the inventory log.

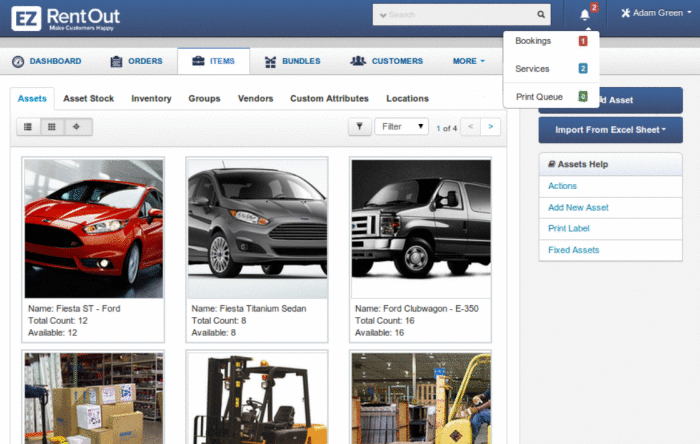
When observing the staff serve customers, it was evident that the current system is too cumbersome for efficient use. Both counter staff members that were observed were constantly flicking through all three logbooks in order to get the right IDs, wasting time and increasing the chance for error. Despite this, the relatively low rate of customers throughout the day stops this from being too much of a problem.

System Comparison:

In addition to the research into the current paper based system employed by ToolGator, the analysis of proprietary software was conducted to check for any possible additional features that could be included in the planned new system as well as the necessary input data.

For the purpose of this investigation, only program features concerning the management of tools will be included – as many solutions also manage staff work times and payment details.

*Figure 6: EZRentOut item browser:*

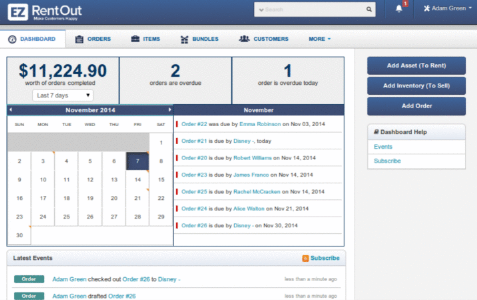
**Proprietary software – EZRentOut:**

EZRentOut is a proprietary software solution designed to manage a rental system. The software is available in four packages of varying features and price points.

Across all packages, rental related features include:

* Revenue tracking – The ability to see the incoming revenue for the month at a glance.
* The ability to see overdue items.
* Excel Import and export- Data can be imported and exported via spreadsheet format.

*Figure 7: EZRentOut finances page:*

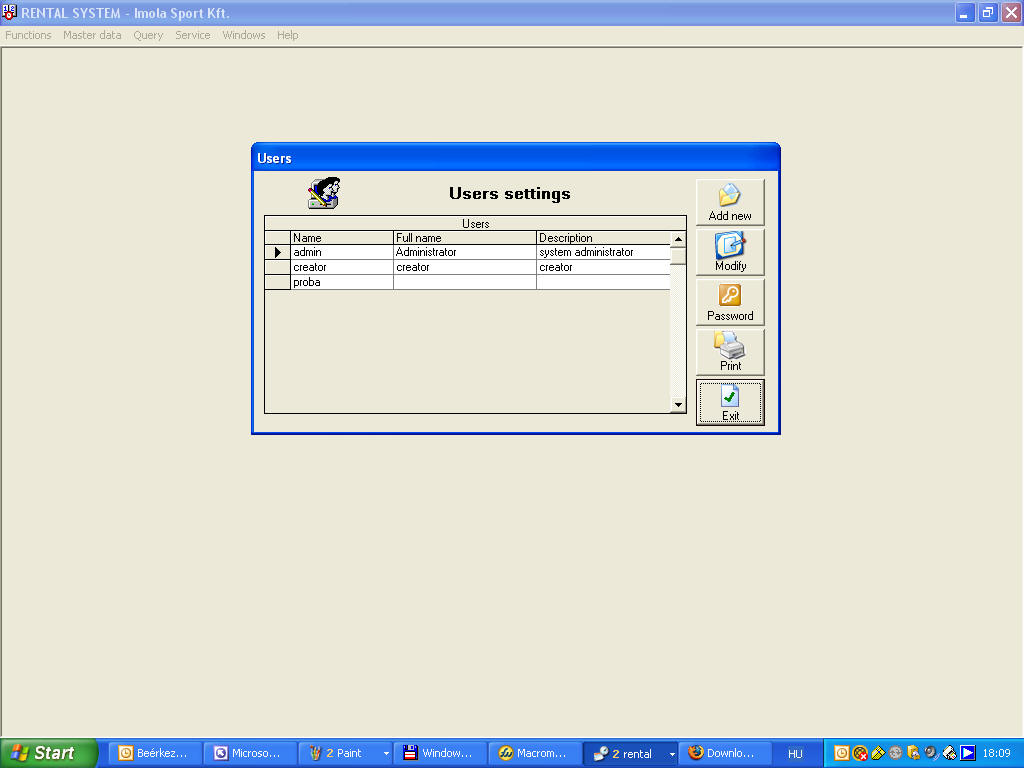
* Invoice management.
* Promotional coupons.
* Image library – Images can be stored and related to entities.
* GPS Location tracking – updated upon scan of a tool’s barcode.

Although preciously undiscussed, the revenue tracking feature appears to be useful. At a glance, the manager could see the income for the month. However, it would require an additional degree of security clearance above admin.

Ease of use would further be improved with overdue items being displayed clearly, otherwise unreturned items may go unnoticed similarly to the paper based solution.

Spreadsheet support will be excluded from the proposed system, it is planned to utilise files instead of spreadsheets and databases to store the data. Despite this, spreadsheets can potentially be a more effective method of storing data in an organised structure, but it should not concern the end user if the program handles all of the data.

GPS Location tracking is out of the scale of the project. Not only is the feature designed for multiple branches of a store (so that tools may be returned to any branch – of which ToolGator has only one), but I have no access to scanners.

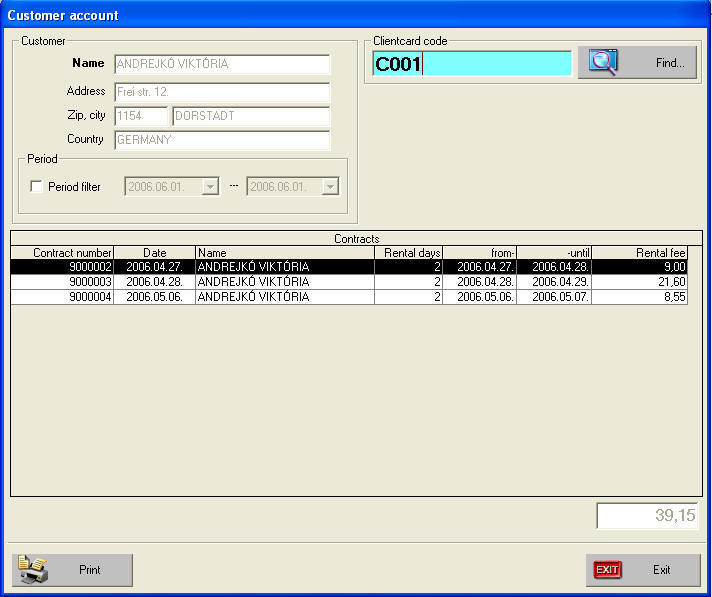
**Proprietary software – ESC-Rental:**

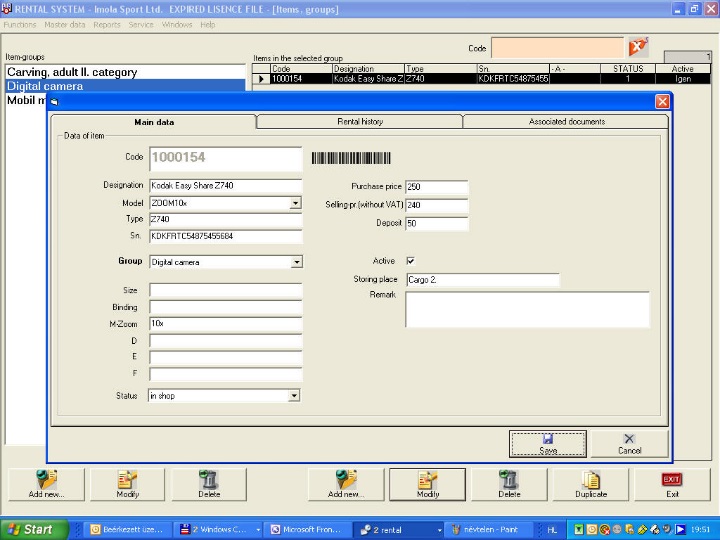
*Figure 8: ESC-Rental user settings:*

Comparatively, ESC-Rental is a low price point solution developed in the last decade. It boasts fewer features than EZRentOut.

* Data amendment – Data can be amended once it has been entered.
* Customer rental cards – Customers are issued with unique cards to speed up the rental procedure.

*Figure 9: ESC-Rental Customer details page:*

* Centralised information – Information about all items placed in the inventory is stored in a record, therefore enabling a simpler search process.
* Promotions and discounts.
* Item reservation.
* Statistics – Program displays top customer, most popular item, and monthly income.

This system is akin to what has been planned for ToolGator’s new system. However, unlike ESC-Rental, unique customer rental cards are not planned to be issued because of the limitation of not having access to a barcode scanner. Instead, **customers will simply be asked for their name, address, and date of birth.**

*Figure 10: ESC-Rental Add to inventory page:*

Data amendment is a point that was brought up before, highlighted in both the interview with manager Nigel Bones and the document analysis. **An administrator should have the ability to make correctional changes to data.**

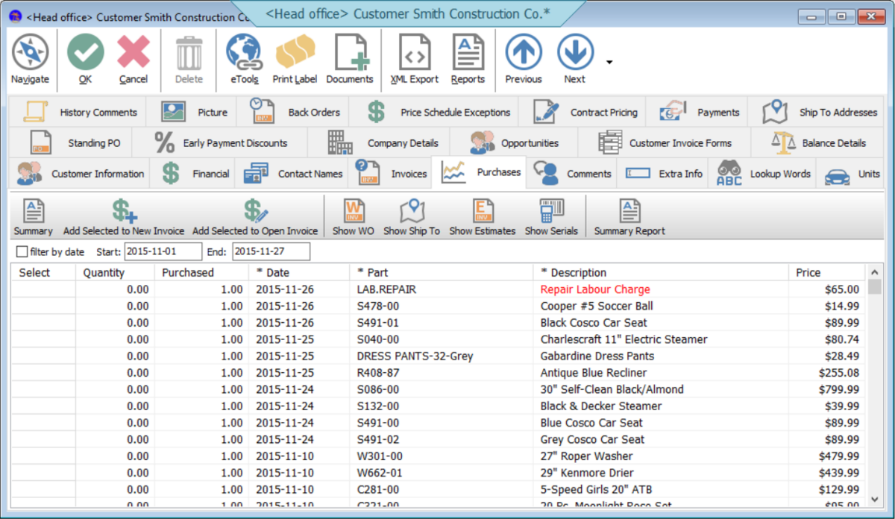
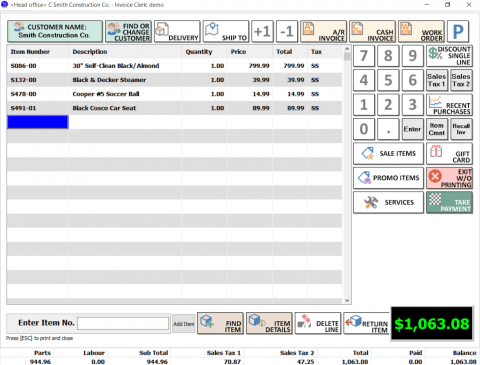
Item reservation has not been discussed in the investigation, implying that there is no demand for such a feature. Customers have had no issue in picking up tools on the day, especially if multiple tools of the same type are in the inventory.

The development of the new solution would be simplified by adopting the idea of centralised information about each tool stored in the inventory. This would reduce the number of files maintained by the system. In addition, in the event that a backup needs to be used to restore the system, fewer files leaves a lower overall chance of error in that the probability of a lost file would be lower.

**Proprietary software – Windward software – Rental Equipment Business:**

Windward software’s solution can either operate on local or cloud based installation. It is a point of sale software, allowing for both rental and sale transactions.

*Figure 10: Windward software customer transaction history:*

* Ability to switch tools between rental and sale.
* Administrator security levels – employees only access relevant information without compromising security or confidential data.
* Local or cloud installation.
* Email marketing – the ability to connect with customers to advertise deals.
* Customer management – Add new clients quickly and efficiently, amend customer information, search for customers by lookup words.

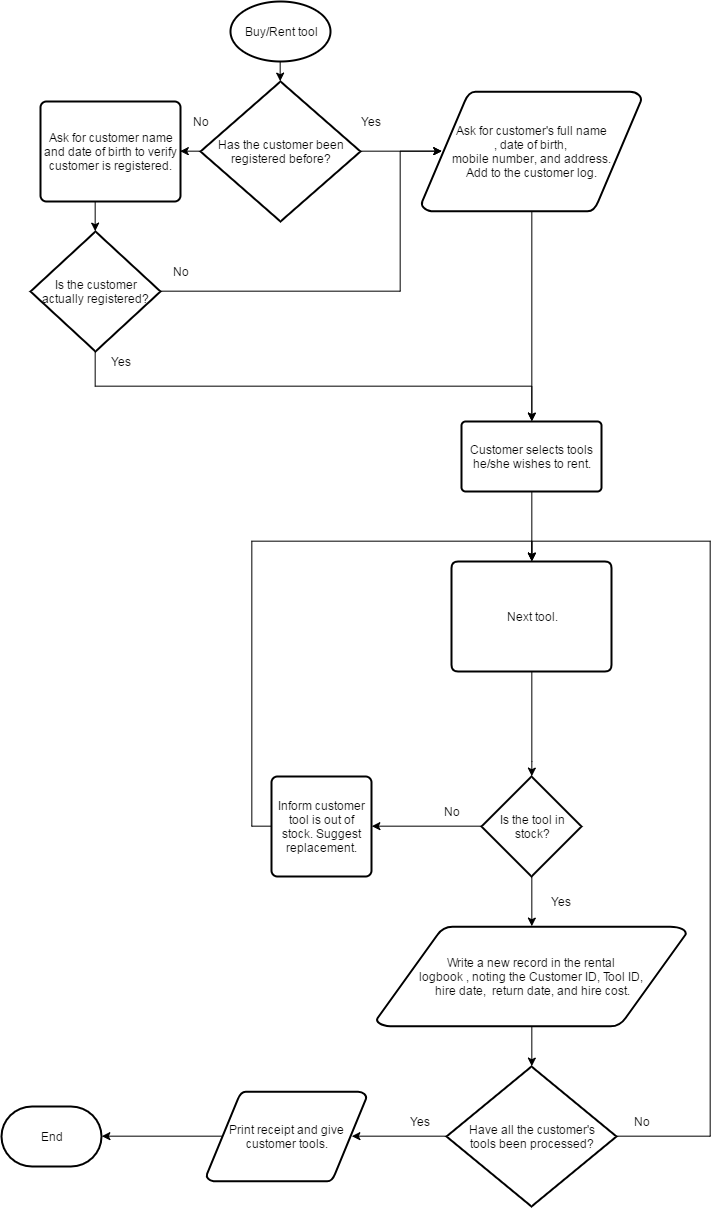
*Figure 11: Windward software customer invoice:*

Windward software’s solution allows not only for the ability to amend item details, but also **change whether an item is available for sale or rent with a simple alteration.** When designing the ToolGator inventory system, it would be prudent to bear this in mind. However, if it introduces too many difficulties, the idea may be abandoned with the only way to change a tool’s sale/rental status being to re-enter it.

Multiple security levels have been planned and discussed prior to researching Windward Software’s program, only re-affirming the decision for its inclusion.

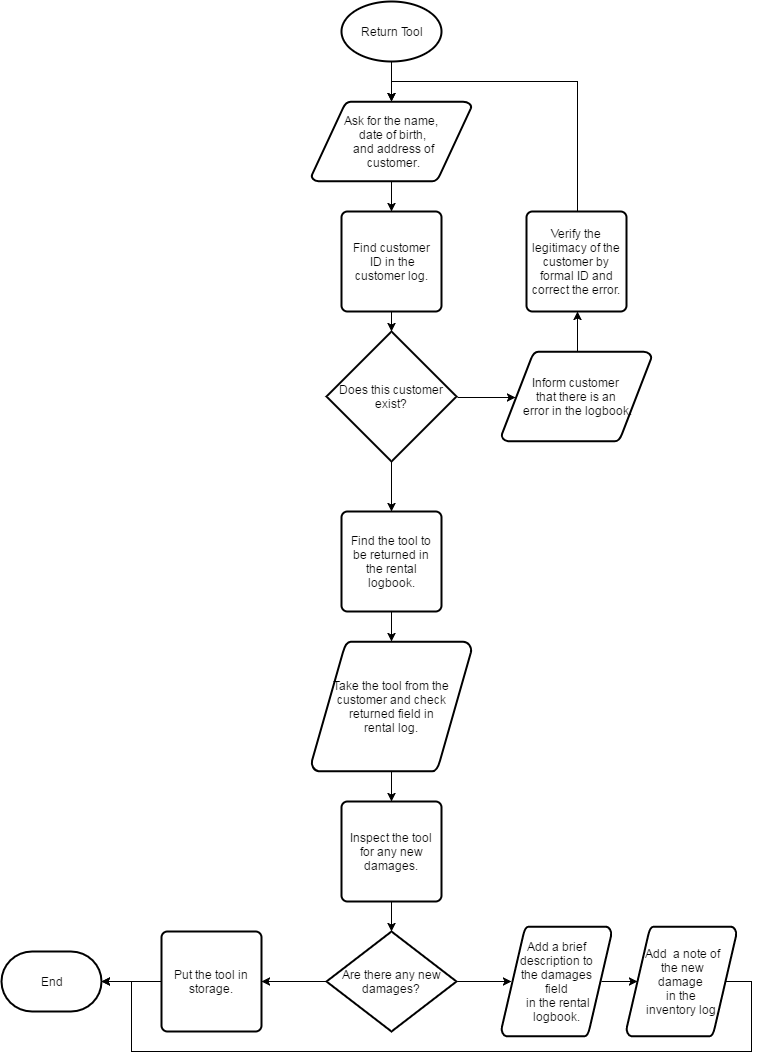
Adding new clients quickly at the desk is already necessary for the current paper based system, but it is a process a computerised system would speed up. As seen in the observation, counter staff constantly flick through different logbooks to determine if a customer is genuine; a procedure that would now be automated by verifying the customer’s name and date of birth.

Email marketing appears to be an unnecessary extraneous feature, not adding to the tool managing ability of the program. In addition, nobody appreciates marketing emails.



Current Process – Flow Diagram:

Renting a tool:

Returning a Tool:

Purpose:

The purpose of the project is to entirely replace the dysfunctional paper based system currently employed by ToolGator, while retaining all prior features as well as vastly improving customer serving and tool management efficiency. The solution should be designed to cope with a larger amount of customers and inventory without hindrance, unlike where the current system buckles when the counter staff members are required to sift through multiple log books.

Method Justification - Creation of the new Solution:

Multiple fundamental methods can be utilised in the creation of a computerised hire management system, potentially functioning as a:

* Standalone application – uses files only intended for reading/writing by the application.
* Database – utilises a database management system such as Microsoft’s Access to allow interrogation and retrieval of data.
* Database dependent application – an application that runs alongside a database, relying on it for importing and amending data.

ToolGator’s new solution will be designed as a standalone application created using VB.net in the Visual Studio integrated development environment. Data will be stored and managed by the application via various files located in its directory.

A standalone application offers a greater degree of freedom; providing the developer with full creative control over all aspects of the program, thus allowing for the implementation of aforementioned features such as **loyalty bonuses**, **administrator security levels**, and a changeable **rental cost algorithm**. Secondly, system maintenance should be a more simple process for the end user in comparison to a database/program hybrid system due to the fact that, in the event of errors, either side of the system could be responsible.

Summary of Objectives:

Upon completion, the program should be able to:

* **Backup and Archive Data**
* **Allow for Customer Loyalty Points**
* **Manage multiple security levels for staff members**
* **Encrypt customer details and staff account information**
* **Automatically delete customer details if they have not made a transaction within the span of a year.**
* **Allow reading of the inventory by other computers on a local area network.**
* **Automatically create unique records for each tool added to the inventory.**
* **Allow staff members to amend additional information about tools with ease.**
* **Suggest a tool’s name based on prior items in the inventory when adding a new price of equipment.**
* **Allow a system administrator to change the hire cost algorithm.**
* **Highlight customers who have not returned their tools in their allotted time.**
* **View and amend customer details and require a customer to be registered before any rental transaction.**
* **Generate monthly statements for each customer.**
* **Change whether an item is available for sale or rent with a simple alteration.**
* **Require the creation of an administrator account on first startup.**

Success Criteria:

A success criteria aids the final evaluation of a program, determining whether it meets the needs of the end user, as well as aiding development by ensuring that the objectives are met.

**Suitability:**

Program suitability is measured by whether it meets the requirements set out by the company, and by extension, the objectives of the solution. Meeting a minimum of all these requirements demonstrates the program’s suitability; potentially demonstrated by comparing the success criteria to a pre-production specification or by an extended trial by staff members.

**Usability:**

Usability is a reference to the learning curve of software, its operation, and an intuitive nature of the user interface. This demands clear navigational paths through the program’s forms and features; making use of self-descriptive buttons and labelled text fields. All forms must be formatted consistently (for example, the position of the back/cancel buttons would remain constant) and text should be suitably clear with a large enough font size so that the software can be effectively operated by a large range of users, taking into account poor eyesight and similar disabilities.

Navigational buttons should be kept separate from the rest of the contents on a form, warranting that users do not make accidental erroneous inputs when trying to navigate through the program.

**Performance:**

Program performance is directly related to the efficiency of its running. All code that is written should not be unnecessarily complex and called too frequently that it hinders computer performance, using a large amount of system resources. This can be measured in Task Manager’s performance tab, with each of the following indicating problems with the program:

* High Current CPU Frequency – indicates an intense workload and overly complex code.
* High RAM usage – indicates that the program occupies a large space in the Random Access Memory, caused by having too much program data open and ready for access at once.
* High Disk usage – indicates that the program is excessively writing to the disk if this lasts for a long duration.

Software reliability is a related factor, unhandled exceptions and fatal errors that may cause the system to crash majorly impact performance. A stable and high performing system puts data at much less risk due to data loss as a result of crashes.

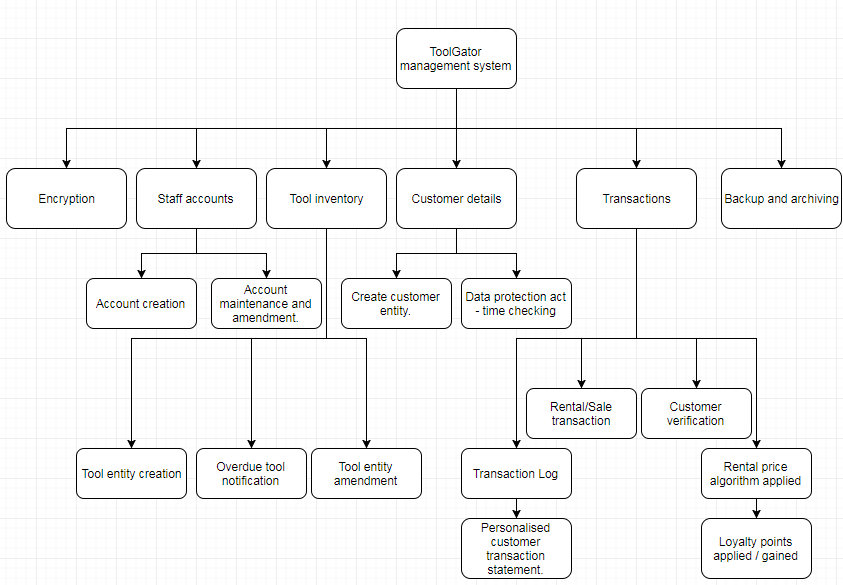
Success Checklist:

In order to gauge the program’s success related to the success criteria, objectives will be referred to during development and checked when applicable features are completed:

|  |  |  |
| --- | --- | --- |
| Does the program require an admin account to be created on first start up? |  |  |
| Does the program provide an admin to create other accounts for staff of various security levels? |  |  |
| Does the admin have access to all areas of the program? |  |  |
| Can the program backup and archive data? |  |  |
| Does the program have the ability to award loyalty points to customers? |  |  |
| Can the loyalty point algorithm be set by an admin? |  |  |
| Are customer details and staff information encrypted? |  |  |
| Are customer details removed from the system after a year in accordance with the data protection act? |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Design**

The proposed system is complex, therefore the program should be divided into multiple smaller modules. Each module should be responsible for a sub-section of the program. The objectives set out in the investigation will form the basis for the aim of each module. I have graphically set out the modules in a hierarchical format to show how each module relates.



**Staff Accounts:**

This area of the program is responsible for the accounts staff members use to access the system. Admins will be able to create accounts for other staff members and amend their details if needed. The program will have to adapt to the security clearance of the user, only allowing access to relevant areas to avoid accidental or malicious damage. All staff members will need to be provided with a window in which their login details can be entered.

**Tool Inventory:**

The tool inventory module relates to the management of all the tools available for rent or purchase. It requires the ability to create unique records for each tool, including relevant information such as base cost, manufacturer, and availability. An administrator should be able to amend information about any tool.

**Customer Details:**

All customers should be registered before rental transactions for security purposes, therefore customer details should be stored and used for verification prior to rental agreements. The program should check the last time the customer used ToolGator on a regular basis, notifying the administrator if the customer has not returned for over a year so that details can be removed in accordance with the Data Protection Act.

**Transactions:**

The transaction modules may potentially be the most complex aspect of the program. There are multiple forms and files that need to work in conjunction with each other.

As mentioned before, customer details should be confirmed prior to each rental agreement, this would be done by asking the customer for their details and determining if their details are already stored. If the customer is unregistered, their details should be entered into the system before any rental agreement, but not necessary for sale.

A rental price algorithm set by an administrator should be used for rental transactions, calculating how much the customer should pay based on a number of variables such as rental period and tool base cost. At this point, loyalty points can be applied to the customer’s account. These can be exchanged for money off future transactions.

Each transaction should be recorded in a file and linked to a form in the program for viewing. The transaction file should not be edited as it is simply a history of all sales and rentals.

At the each month, the program should search the transaction log and produce a statement for each customer containing any payments.

**Backup and archiving:**

All data should be backed up to a specified location on a regular basis following son father grandfather backup rotation scheme. This contingency planning prevents major data loss in the case of emergency.

**Encryption:**

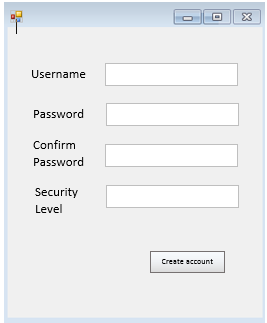
Customer details and staff logins should be encrypted as a layer of protection against data theft. The program will store the key internally, a decision that can be made due to the program working on a purely local level with physical access required to steal data.

Form design – input and output:

The system must be able to process data for different entities input by user. This will be achieved using a GUI (*graphical user interface)* consisting of multiple forms. These forms are responsible for inputs and outputs of the system and should be designed intuitively for the end user’s ease of use.

I have drafted the forms required in the program using blank forms from visual studio and added appropriate features using Microsoft Word.

**Account Creation:**



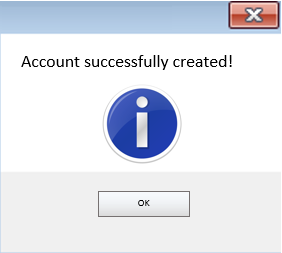
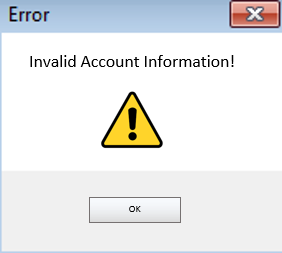
If this is the first time the program has been run, then this is the first form that is displayed. An account must be created with administrator level security to proceed. This account will have access to all areas of the program, including the ability to create other administrator accounts.

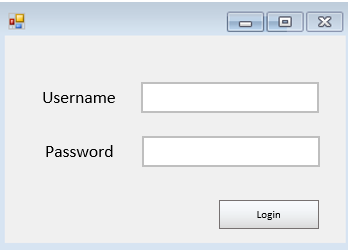
The username, password, and security level should be stored in an external file that can be read by the login form. In addition, the data stored in the login and passwords file should be encrypted to prevent access to administrator accounts by staff members.

The text boxes will have a presence check as well as double entry verification to ensure that a record can be created with all of the required fields, therefore no problems should be encountered when reading the data. Furthermore, the username should be a unique identifier, and an account should not be created if another account has the same username.

The password text box should only display a designated password character such as an asterisk so that onlookers cannot discover the password.

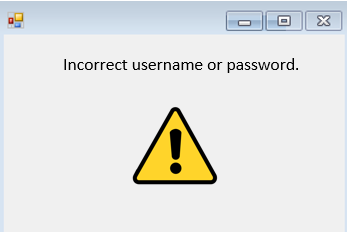
When the create account button is pressed, a message box should be displayed notifying the user if a staff account was successfully created.

**taff Login:**



If this is not the first time this program has been launched, this is the first form with which the user is greeted, otherwise this form is displayed after an account has been created.

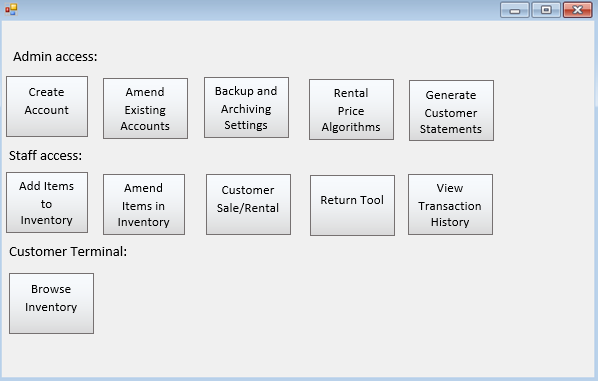
It has a simple design of two input fields and a login button. When the login button is pressed, the program should decrypt the file containing the user details, then linearly search through each record, comparing the user input with that of the current record. If the username (used as a primary key) matches, then the input password and password on file are compared. If both username and password match, then the security clearance of the respective account is read from file, and assigned to a global variable in a cross-form module sho that it can be accessed from all areas of the program.

If no errors are met during this process, and all conditions are met, then the login is successful and a message box should be shown. Otherwise, a message box prompting the user to re-enter login details should be shown.

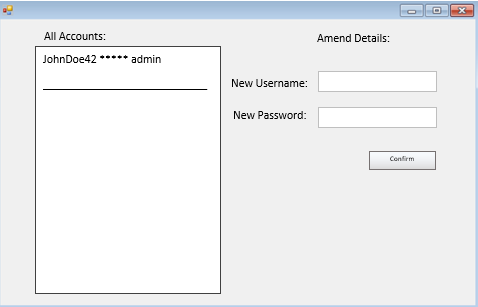


**Main Panel:**

The main panel should greet the user after a successful login. From here, all areas of the program can be accessed by an administrator. Regular staff accounts will have more limited access to areas of the program, buttons leading to areas above the current user’s clearance level (found by reading the global security clearance module) will be unavailable. Thirdly, the customer terminal security setting should only have the option to browse the inventory, acting as an electronic in-store catalogue.

This form will only consist of buttons, commands and actions can only be made via other forms.

**Amend Existing Accounts:**

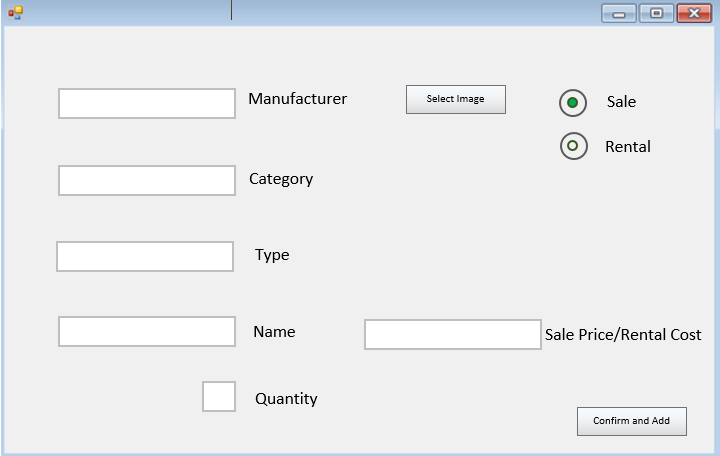


The form that can amend existing accounts is only accessible to administrators. There would be a list box displaying all accounts with obscured passwords to provide a layer of protection against onlookers. It should be impossible to find the password in plain text for an account from the program or the encrypted file, as a result, an admin needs the ability to set new login details for users.

To amend details, account details can be selected from the listbox, then are altered via the input boxes.

When the confirm button is pressed, a new file with the amended contents is created. Presence checks should be carried out on each field, ensuring that no blank lines are created in the file to avoid file reading errors.

**Add Tools to Inventory:**



Arguably one of the most important forms in the program, new tools and all of their required input information are added to the inventory from here. The user can input information by navigating the form with the mouse or by tabbing through the dropdown menus. Previous inputs should be stored in in external files and read to populate the fields, with suggestions based on the current selection in the prior fields. For example, the suggestions in the ‘name’ dropdown box should only match the names previous tools that fall in the specified manufacturer, category, and type.

Radio buttons act like a switch between adding a tool with a sale flag or rental flag, a field in a record that will be created.

The quantity field simple denotes how many unique records of this tool will be added to the inventory file.

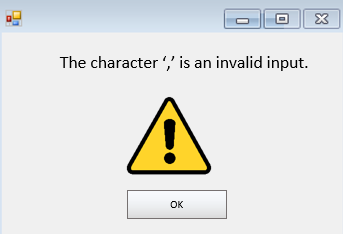
The Sale Price is the flat value that would be used for an item for sale, excluding VAT. On the other hand, the Rental Cost isn’t a flat value that the customer pays, rather it is a constant value for each tool that is used in the rental cost calculation (set on a different form).

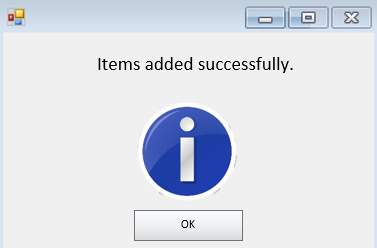
An image should be able selected for each tool; a feature that would be used in the ‘browse inventory’ form. This should increase ease of navigation for customers in said form as well as identification. Despite this, it would be a purely superficial feature.

Upon pressing the confirmation button, each field should be validated with a presence check to avoid blank fields in the inventory record, as well as a format check to ensure that whatever character that is used to separate fields in the record is not entered to avoid additional and incorrect records.

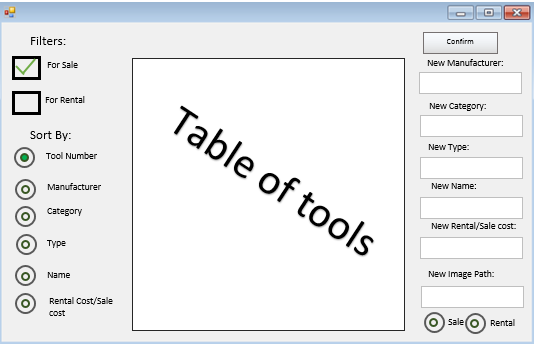
The user should be prompted if any of these inputs are invalid and records should not be made. In this case, a message box should display the invalid inputs. Otherwise, a success box should appear.

*If the file uses a comma to separate fields in a record:*





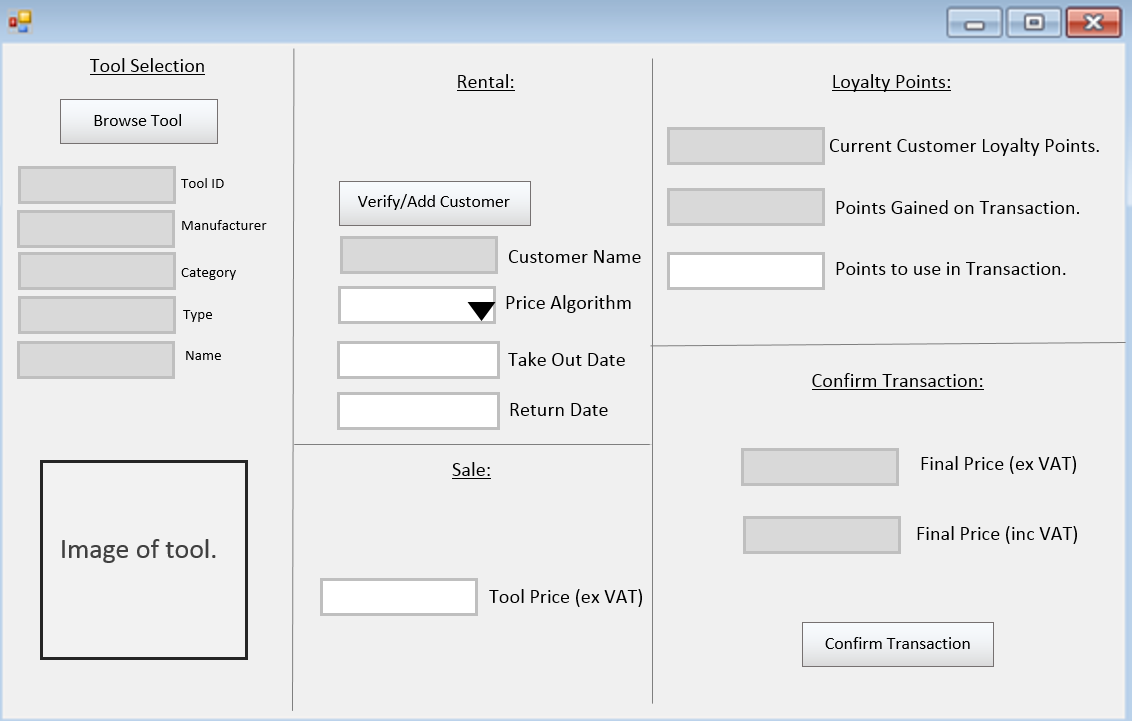
**Amend Inventory:**

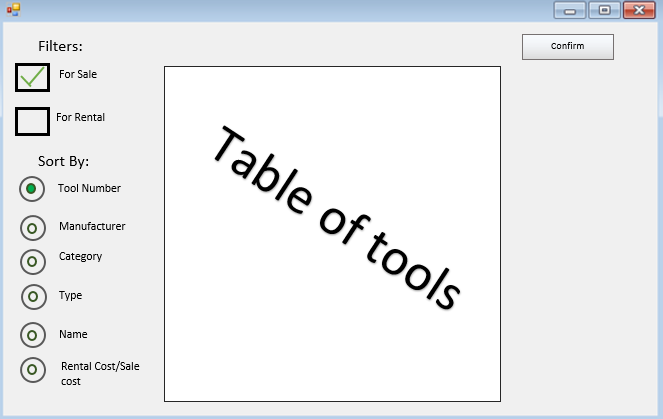


This form hosts similar processes to that of the ‘add to inventory’ form, but with its purpose to show a detailed view of all entities in the inventory with the ability to edit any record (with the exception of its number / primary key).

The table in the centre of the form should have the ability to be filtered and sorted by the controls on the left. Selecting a record to amend should be simple, the user should be able to simple click on the record and its fields should appear in the boxes to the right. From here, the new details can be entered. Upon pressing the confirmation button, the input fields are verified and validated using the same methods as in the ‘add to inventory’ form; giving an appropriate message box if successful or unsuccessful and then writing to file.

**Customer Sale/Rental:**



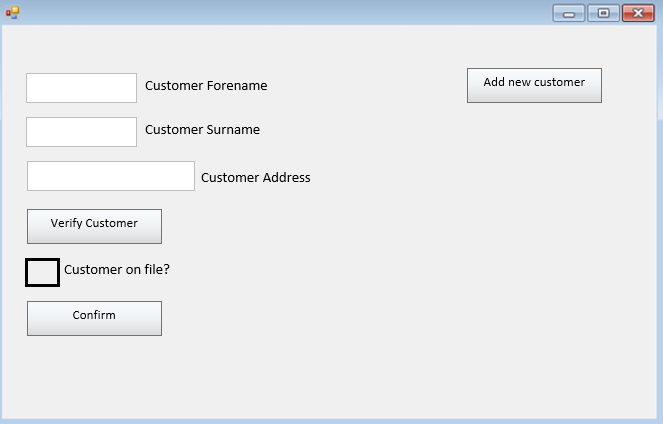
A complex form at first glance, the customer sale/rental window facilitates the process of a sale or rental of a tool. Firstly, the staff member finds the tool the customer wishes to rent or buy. This should open a secondary form which would look similar to the ‘amend inventory’ window; only that it lacks the ability to amend tools.

When the appropriate tool is selected, confirm will return the user to the previous form.

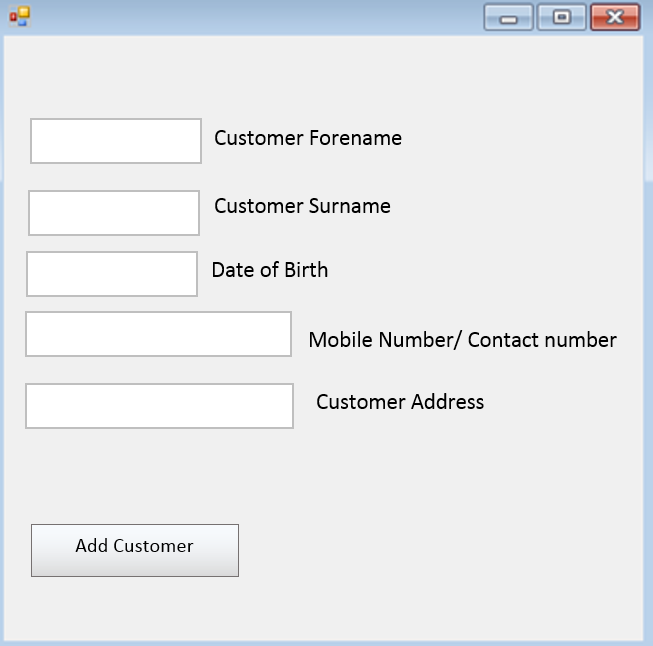
The text fields will be automatically filled with the tool details from the record. In addition, the image box should display the tool. All this adds a layer of confirmation.

Based on whether the selected tool was available for rent or sale, the rental or sale areas of the form will display. In the case of a rental transaction, the customer must be verified as a layer of protection against theft, malicious use, and to make an entry into the transaction log.

Rental:



The Verify/Add customer button leads to the form above. The staff member should ask the customer their name and address. The customer file should then searched for a record containing all three details, returning a flag if the record exists. In this case, the user may return to the previous form.

If the customer is not on record, suitable checks should be carried out on the customer outside the scope of the program. If the customer passes these checks, then the ‘add new customer’ form can be used to input the relevant data.

This form should also be accessible from the main switchboard; avoiding navigation through the Customer Sale/Rental form if a customer wishes to be registered.

Subsequently, the user should select a pricing algorithm for the rental transaction. These should be set by an administrator. The algorithm would function on the tool base cost and rental period; determined by the booking and return date.

Sale:

The sale process should be much simpler. The tool price excluding VAT should be displayed. There is no need for registering the customer since the tool will not be returned.

Loyalty Points:

Loyalty points should be awarded and spent in the next section. The algorithm for earning points should be set by the administrator, as well as how much each loyalty point is worth.

Points should reduce the pre-tax price.

Confirm Transaction:

Rental:

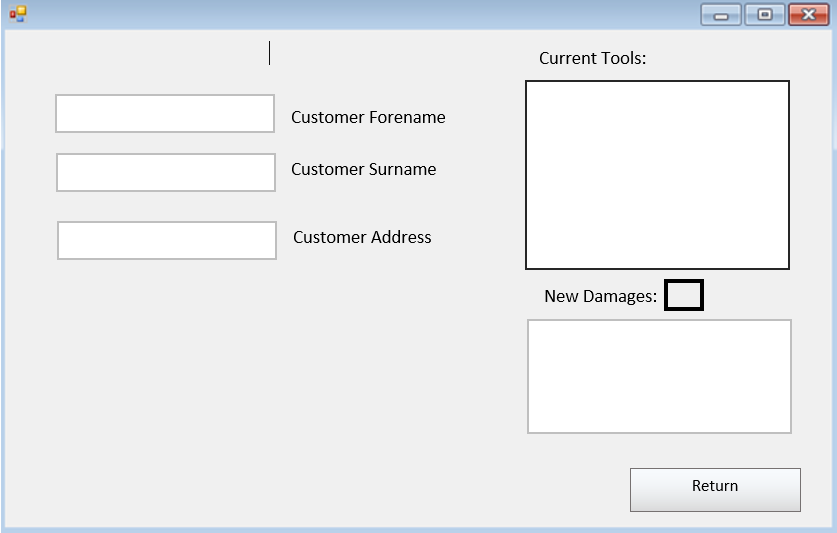
An entry should be made in the rental transaction file with the fields ‘Rental ID’, ‘Customer ID’, ‘Tool ID’, ‘Hire Date’, ‘Return Date’, ‘Hire Cost’, ‘Returned’, and ‘New Damages’.

Sale:

If the transaction is concerning the sale of an item, the an entry should be made in the sale transaction file with the fields ‘Rental ID’, ‘Customer ID’, ‘Tool ID’, and ‘Sale Date’.

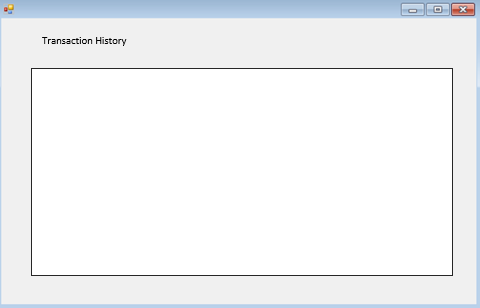
**Return Tool:**

Returning a tool should be a simple process. The customer’s details should be entered to find the customer ID, then this key should be used to find the current tools in the transaction log and mark them as returned. Each returned tool should be marked as returned in the inventory log. New damages should be noted in both logs.



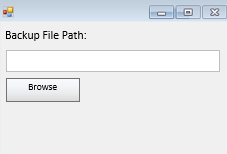
**View Transaction History:**

This form should show the history of all transactions.



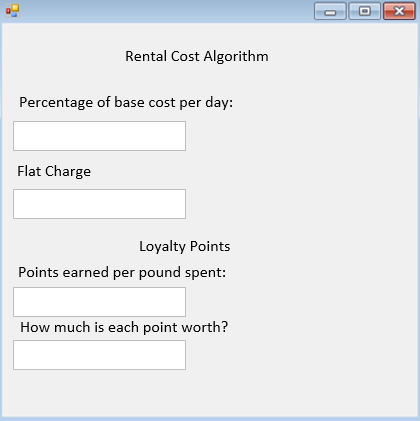
**Backup and Archiving:**

A very simple form, it should be accessed by an admin to set the location of the program’s automatic backups.



**Rental Algorithm and Loyalty Points:**

The administrator should be able to access this form. It controls the charge customers will pay for rental transactions as well as loyalty points.



**Data Dictionary:**

Tool Inventory:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Tool ID | Integer | Unique member identifier of tool. | Primary Key | *None – Automatically Generated* |
| Manufacturer | String | Manufacturer of the tool | - | Character Check/ Presence Check |
| Category | String | Is the tool a power tool? Hand tool? | - | Character Check/ Presence Check |
| Type | String | What is the tool? Is it a saw? Drill? Hammer? | - | Character Check/ Presence Check |
| Name | String | The name of the tool assigned by manufacturer. | - | Character Check / Presence Check |
| Sale/Rental Status | String | A self-documenting field stating whether a tool is available for rent or sale. | - | Presence Check. |
| Base Cost | Float | The base cost, used either as a value in the rental price algorithm or the pre-tax price of an item for sale. | - | Character Check / Presence Check |
| Image File Path | String | The file path to an image of the tool entity. | - | Character Check / Presence Check |
| Damage Notes | String | Any current notes concerning the tool. | - | Character Check. |
| Currently Rented | String | Is the current tool currently available for rent? | - | *None – Automatically Generated* |

The Tool Inventory file is used to store the details of all the tools in ToolGator’s stock. This is what the system will read from and write to. Multiple forms will access this file:

* Add Tool
* Customer Rental/Sale
* Return Tool
* Browse Tool
* Amend Inventory

The number of records should constantly increase over time as items are added to the inventory. Therefore this file needs to, for all intents and purposes, hold an infinite number of records.

Each time a tool is added to the inventory, a record is appended to this file.

This file is edited whenever a tool is taken out or returned, changing the ‘Currently Rented’ field. This ensures that customers do not attempt to take out a tool that is already rented as the system should not allow this to happen.

The primary key is the Tool ID; a unique identifier. The program should handle all tools using their Tool ID instead of utilising a combination of other fields such as name, category, and manufacturer. This is important if multiple tools of the same type are in stock.

The inventory file shall be accessed serially. This is because in almost all proposed uses of this file, the program displays its entire contents in a list or table and any searching is facilitated within the program, meaning there is a negligible performance impact in access methods at this point. Random access is not necessary, as there is no confidential data stored in this file.

Customer Log:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Customer ID | Integer | Unique member identifier of customer. | Primary Key | *None – Automatically Generated* |
| Forename | String | Forename of registered customer | - | Character Check/ Presence Check |
| Surname | String | Surname of registered customer | - | Character Check/ Presence Check |
| Registration Date | String | Date at which customer was registered with Toolgator’s system. | - | *None – Automatically Generated* |
| Contact Number | String | Contact telephone number of customer. | - | Character Check/ Presence Check |
| Address | String | Customer’s address. Can be used for ensuring the system can distinguish between two customers of the same name. | - | Character Check/ Presence Check |
| Current Loyalty Points | Integer | The current points earned by the customer | - | *None – Automatically Assigned* |

The customer log file will contain all necessary details per customer per record. When a new customer is registered, their details are taken from text fields and appended to this file to create a new record. This file is used in the rental process to verify customers and store loyalty points. When verifying a customer, the file is searched linearly to find a match with the input customer details on the form and their forename, surname, and address on file. The primary key in this file (Customer ID) is similar to that in the Tool Inventory; it is an identifier that is only used by the program. It is unsuitable for the customer to remember an ID, hence they cannot be searched for using such a parameter. Instead, in processes that require customer input, the file is searched linearly looking for a matching forename.

This is a file that can be searched either linearly or using a binary search, depending on the search subject.

Rental Transaction Log:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Rental ID | Integer | Unique member identifier of rental transaction. | Primary Key | *None – Automatically Generated* |
| Tool ID | Integer | Unique member identifier of tool. | Foreign Key | *None – Automatically Generated* |
| Customer ID | Integer | Unique member identifier of customer. | Foreign Key | *None – Automatically Generated* |
| Hire Date | String | Date of rental. | - | *None – Automatically Generated* |
| Return Due Date | String | Tool due date. | - | Character Check/ Presence Check |
| Tool Returned | String | A self-documenting field stating whether the tool has been returned. | - | *None – Automatically Generated* |
| Hire Cost (ex VAT) | Float | The price paid by the customer excluding VAT. | - | *None – Automatically Generated* |
| Hire Cost (inc VAT) | Float | The price paid by the customer including VAT. | - | *None – Automatically Generated* |
| Loyalty Points Used | Integer | The amount of loyalty points used by the customer. | - | *None – Automatically Generated* |
| Total Loyalty Point Discount | Float | The total money off the hire cost earned by spending loyalty points. | - | *None – Automatically Generated* |

The rental transaction log is simply a record of all rental transactions that have taken place. Whenever a tool is taken out, a record is appended to the log. In addition, when an item is returned, its ‘Tool Returned’ field is marked as true.

This file serves little functional purpose in the processes of the program, but it does provide a way to track overdue tools and see the tools taken out by each customer.

Despite the fact that there is a unique Rental ID, this file also contains the Tool ID and Customer ID from the Tool Inventory and Customer files. This is as in the entity relationship diagram.

The file is added to serially, therefore it is already sorted by time. This is important as it allows the program to easily find a tool’s most recent record in the transaction log by searching backwards.

Sale Transaction Log:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Sale ID | Integer | Unique member identifier of sale transaction. | Primary Key | *None – Automatically Generated* |
| Tool ID | Integer | Unique member identifier of tool. | Foreign Key | *None – Automatically Generated* |
| Sale Date | String | Date of sale | - | *None – Automatically Generated* |
| Sale Price (ex VAT) | Float | The price paid by the customer excluding VAT. | - | *None – Automatically Generated* |
| Sale Price (inc VAT) | Float | The price paid by the customer including VAT. | - | *None – Automatically Generated* |

The Sale Transaction Log is very similar to the rental transaction log, but it should seldom be accessed. It is simply a record of all sales, added to serially. It is not used for verification purposes.

VAT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Value Added Tax | Float | The current VAT rate. | - | Character Check/ Presence Check |

The VAT file contains nothing but the current rate of VAT. The file is read whenever a tool is hired or sold to calculate a price including VAT, as well as for a field in transaction files.

Loyalty Points Settings File:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Loyalty Point Value (In £) | Float | The value assigned to each loyalty point. | - | Presence Check, Character Check |
| Loyalty Points Earn Rate (per £) | Integer | The amount of loyalty points earned by registered customers per pound. | - | Presence Check, Character Check |

The Loyalty Points Settings File would only contain a single record with the fields ‘Loyalty Point Value’ and ‘Loyalty Points Earn Rate’. These values can be changed by an administrator.

Rental Algorithm Settings File:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Percentage of base cost per day. | Float | The percentage of the tool’s base cost that would be paid per day of hire. | - | Range Check, Presence Check, Character Check |
| Flat Charge | Float | A flat charge | - | Presence Check, Character Check |

The Rental Algorithm Settings File would only contain a single record with the fields ‘Loyalty Point Value’ and ‘Loyalty Points Earn Rate’. These values can be changed by an administrator.

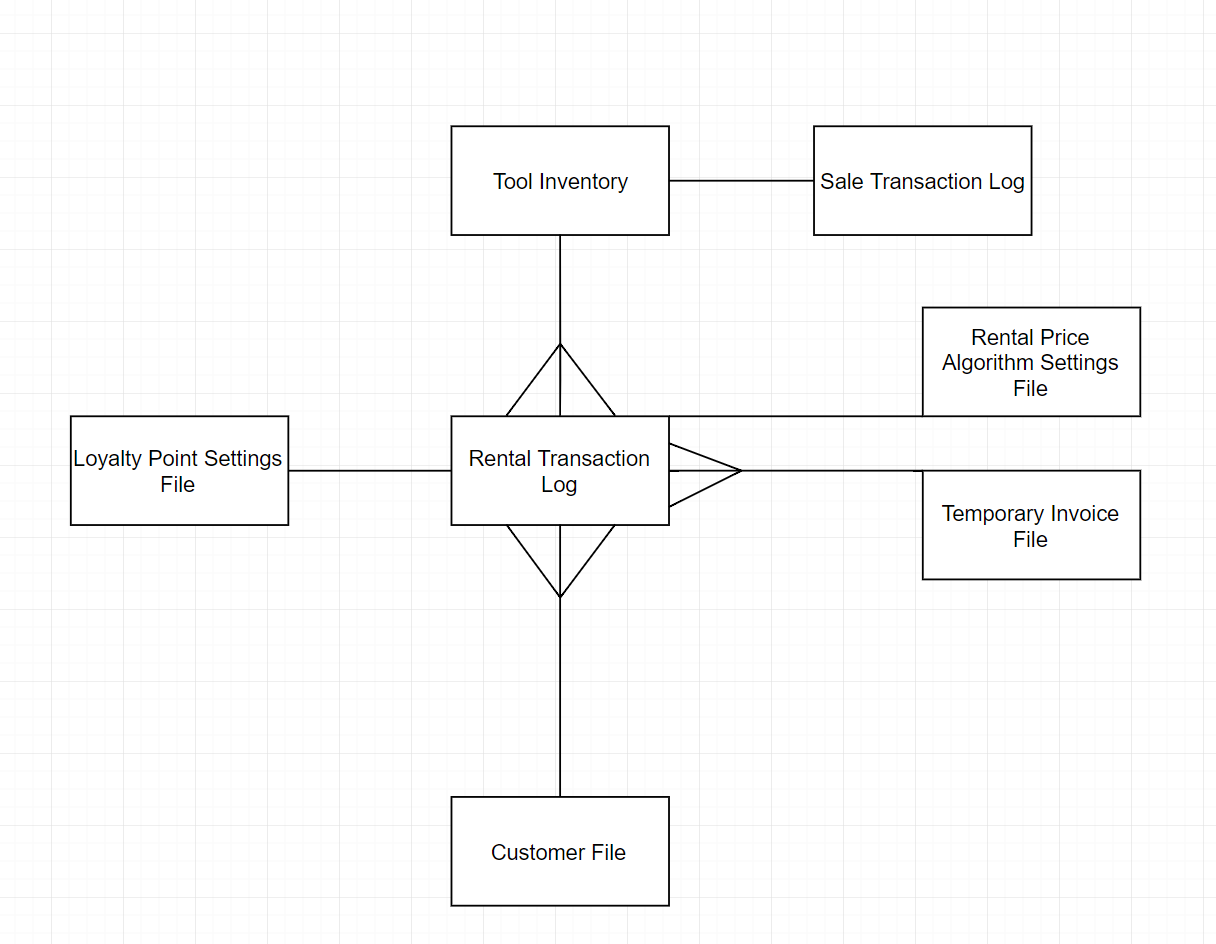
Login Details File:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Description** | **Key** | **Validation** |
| Username | String | A unique string to identify a staff account. | Primary | Presence Check, Character Check |
| Password | String | A secret string used as a security step for login. | - | Presence Check, Character Check, Double Entry Validation |
| Security Clearance | String | The clearance level of the account: admin, staff, or customer terminal. | - | Presence Check |

The Login Details File is a single file containing the login details and security level of all staff members with an account. When an account is created, its details should be appended serially to this file. A serial method should be used because it is random who will be logging in at any point, therefore there would be no benefit with other search methods.

During the search of the file, the username is used as a primary key. The program should compare the user’s input with that on file, then proceed to compare the password to that on file. If the file is searched without a match, an error message should be displayed.

Entity-Relationship Diagram:



Pseudo-code processes:

**Presence Check Pseudocode:**

If TextFieldContents = empty OR Combobox Selection = empty Then

Output “Ensure all information has been entered.”

Endif

**Character Check Pseudocode:**

Set badCharacter = “,”

Set badCharacterFound = false

For each character in TextFieldContents text

If character = “,” Then

badCharacterFound = True

End If

End Loop

If badCharacterFound = True Then

Output “Do not use the character:” + badCharacter

End if

**Range Check Pseudocode:**

Set x

Set y

If NOT (TextFieldContents > x AND TextFieldContents < y) Then

Output “Value must be between” + x + “and” + y

End if

**Double Entry Validation:**

If NOT (TextFieldContents1 = TextFieldContents2) Then

Output “Ensure entered data matches”

End if

**Processes Pseudocode:**

**Create Account Button Pressed:**

Set Username = UsernameTextField

Set UsernameValidation = UsernameTextFieldValidation

Set Password = PasswordTextField

Set PasswordValidation = PasswordTextFieldValidation

Set SecurityLevel = SecurityLevelTextField

If NOT (Login file exists)

Create Login File

End if

If Username = Username Validation AND Password = PasswordValidation

Append to file (Username + Password + SecurityLevel)

End if

**Encrypt Document**

Set i = 0

Set Keyword = “Frigate”

Foreach record in file

Foreach character in record

Set CharacterAscii = ASCII value of character

CharacterAscii += ASCII value of letter (i) in Keyword

If CharacterAscii > 127

CharacterAscii = CharacterAscii – 127

Set NewCharacter = Character value of CharacterAscii

Write character to temporary file

Set i += 1

End for

End for

Delete previous file and rename temporary file to previous file’s name.

**Decrypt Document**

Set i = 0

Set Keyword = “Frigate”

Foreach record in file

Foreach character in record

Set CharacterAscii = ASCII value of character

CharacterAscii += ASCII value of letter (i) in Keyword

If CharacterAscii < 0

CharacterAscii = 127 – Character Ascii

Set NewCharacter = Character value of CharacterAscii

Write character to form in program

i += 1

End for

End for

**Login Button Pressed**

Set Username = UsernameTextField

Set Password = PasswordTextField

Foreach record in file

If Username = First field in record

If Password = Second field in record

Set SecurityClearance = Third field in record

SHOW Switchboard form

HIDE Login form

End if

End if

End for

OUTPUT “Username or password incorrect”

**On Program Startup**

If Login File Exists

SHOW Login form

HIDE Splash Screen

End If

If Not (Login File Exists)

SHOW Account Creation form

HIDE Splash Screen

End if

**Display Login Accounts:**

Decrypt login details

Foreach record in file

Display record in list box

End for

**Edit Login Account Button Press:**

Set n = number of records

Set NewUsername = NewUsernameTextField

Set NewPassword = NewPasswordTextField

Set NewSecurityLevel = NewSecurityTextField

Set Line = record number of selected line

For i = 0 to Line - 1

Copy record to temporary file

End for

Append new NewUsername, NewPassword, NewSecurityLevel

For j = Line to n

Copy record to temporary file

End for

Delete previous file

Rename temporary file to name of previous file

**Add tool to inventory button press:**

Set Manufacturer = ManufacturerTextField

Set Category = CategoryTextField

Set Type = TypeTextField

Set Name = NameTextField

Set Quantity = Quantity Text Field

If Sale = true

Set Status = Sale

Set Price = SalePriceTextField

Else

Set Status = Rent

Set Price = RentPriceTextField

End if

Set ToolImagePath = ImageFieldPath

Set DamageNotes = “”

Set Currently Rented =False

If Inventory file exists

Set RecordNumber = First Element of last line in file

Else

Create Inventory File

Set RecordNumber = 0

End if

For I = 0 to Quantity

Append to record to inventory file (record ID, Manufacturer, Category, Type, Name, Status, ToolImagePath, DamageNotes, CurrentlyRented)

Record Number +=1

End for

**Backup and Archive Settings:**

Set backupPath = BrowseFunction()

BrowseFuntion()

Open explorer window

Return (Selected File Path to Folder)

End Function

Delete previous backup settings file

Create new backup settings file with backupPath

**Loyalty Points Settings:**

Set PointValue = ValueTextField

Set EarnRate = EarnTextField

Delete previous Loyalty Points Settings file

Create new Loyalty Points Settings file with PointValue and EarnRate fields

**Rental Algorithm Settings:**

Set BaseCostPerDay = BaseCostPerDayTextField

Set FlatCharge = FlatChargeTextField

Delete previous Rental Algorithm Settings file

Create new Rental Algorithm Settings file with BaseCostPerDay and FlatCharge fields

**Filter tools for temporary inventory file**

Declare toolDetailsArray (0 to 9)

For each line in file

Set toolDetails = line

Set toolDetailsArray = split line with comma

If toolDetailsArray(5) = “Sale” and sale filter = true

Write toolDetails to new file.

If toolDetailsArray(5) = “Rent” and sale filter = false

Write toolDetails to new file.

Next

**Display tools in table**

Set x = 0

Set y = 0

Declare toolDetailsArray (0 to 9)

For each line in file

Set x = 0

Set toolDetails = line

Set toolDetailsArray = split line with comma

For Each field in toolDetailsArray

Table(x, y) = field

x += 1

Next

y += 1

Next

**Amend Inventory**

Set ID = First element of selected line

Set I = 0

For i = 0 to Set ID

Append line to temporary file

Next

Append (ToolIDTextField text, ManufacturerTextField text, CategoryTextField text, TypeTextField text, NameTextField text, Sale/RentalField text, BasecostTextField text, imagePathTextField text, notesTextField text, currentlyRentedTextField text)

For I = SetID to End of file

Append line to temporary file

Next

Delete previous file

Rename temporary file to that of the original file.

**Customer Sale / Rental**

Open form ‘Tool Table’

Declare toolDetailsArray (0 to 9)

Set ManufacturerTextField = toolDetailsArray(1)

Set CategoryTextField = toolDetailsArray(2)

Set TypeTextField = toolDetailsArray(3)

Set NameTextField = toolDetailsArray(4)

Set ImageBox = Get image from path toolDetailsArray(7)

If toolDetailsArray(5) = Rental

Open form ‘verify customer’

Set CustomerNameTextField = CustomerArray(1)

Set pricePerDay = Rental Algorithm File(0) \* toolDetailsArray(6)

Set flatCharge = Rental Algorithm File(1)

Set cost = (returnDateTextField value- HireDateTextField value) \* pricePerDay + flatCharge

Set loyaltyPointRate = Loyalty Points file(1)

Set LoyaltyPointsEarned = loyaltyPointRate \* cost

Set cost = cost – loyaltyPointsTextField text \* Loyalty Points file(0)

Set costVAT = cost + cost \* (Vat file(0) / 100)

Append to transaction log (Rental ID, toolDetailsArray(0), CustomerArray(0), HireDateTextField value, returnDateTextField value, false, cost, costVAT, loyaltyPointsTextField text, (loyaltyPointsTextField text \* Loyalty Points file(0))

For i = 0 to toolDetailsArray(0) -1

Copy record from inventory file to temporary file

Next

Write to temporary file (toolDetailsArray(0), toolDetailsArray(1), toolDetailsArray(2), toolDetailsArray(3), toolDetailsArray(4), false, toolDetailsArray(6), toolDetailsArray(7), toolDetailsArray(8), toolDetailsArray(9))

For I = toolDetailsArray(0) + 1 to End of file

Copy record from inventory file to temporary file

Next

End if

If toolDetailsArray(5) = Sale

Set Cost = toolDetailsArray(6)

Set costVAT = cost + cost \* (Vat file(0) / 100)

Append to transaction log (Sale ID, toolDetailsArray(0), currentDate, Cost, costVAT)

For i = 0 to toolDetailsArray(0) -1

Copy record from inventory file to temporary file

Next

For I = toolDetailsArray(0) + 1 to End of file

Copy record from inventory file to temporary file

Next

Delete previous file

Rename temporary file to that of the original file.

Form Tool Table:

Set x = 0

Set y = 0

Declare toolDetailsArray (0 to 9)

For each line in file

Set x = 0

Set toolDetails = line

Set toolDetailsArray = split line with comma

For Each field in toolDetailsArray

Table(x, y) = field

Set x += 1

Next

Set y += 1

Next

Return Selected Tool

Verify Customer:

Set Forename = ForenameTextField Text

Set Surname = SurnameTextField Text

Set Address = AddressTextField Text

Set found = false

Foreach line in customer file

CustomerArray(0 to 6)

if CustomerArray (1) = Forename AND CustomerArray (2) = Surname AND Address = CustomerArray (5)

Set found = true

Return (CustomerArray (1), CustomerArray (2))

End if

Next

If found = false

Open ‘Add customer’ form

End if

Add Customer Form:

Set Forename = ForenameTextField Text

Set Surname = SurnameTextField Text

Set Address = AddressTextField Text

Set Birthdate = SurnameTextField Text

Set number = ContactNumberTextField Text

Append to customer log (Forename, Surname, Address, Birthdate, number, 0)

**Return Tool:**

Set Forename = ForenameTextField Text

Set Surname = SurnameTextField Text

Set Address = AddressTextField Text

Set Notes = NotesTextField Text

Declare customerDetailsArray (0 to 6)

Declare transactionDetailsArray (0 to 9)

Declare toolDetailsArray (0 to 9)

Declare ToolIDsArray

Declare FinaltoolDetailsArray(0 to 9)

Set i = 0

*//Search customer file for customer ID using these*

For each line in customer file

Set customerDetailsArray = split line with comma

If customerDetailsArray(1) = Forename AND customerDetailsArray(2) = Surname AND customerDetailsArray(5) = Address

Set Customer ID = customerDetailsArray(0)

End if

Next

*// Using customer ID, we now search for the Tool IDs that this customer is currently renting in the transaction log.*

For each line in transaction file

Set transactionDetailsArray = split line with comma

If Customer ID = transactionDetailsArray(2)

ToolIDsArray(i) = transactionDetailsArray(1)

End if

Set i += 1

Next

*// Now we can get the tools from the inventory that this customer is renting and display them in a box for selection.*

For each line in inventory file

Set toolDetailsArray = split line with comma

If toolDetailsArray(0) = ToolID

Add toolDetailsArray to box

End If

Next

*// Now we need to return the selected tool. Select tool and get tool ID. Mark as returned in inventory and add notes.*

Set toolDetailsArray = split selected line with comma

Set ID = toolDetailsArray*(0)*

For each line in inventory file

Set toolDetailsArray = split line with comma

If toolDetailsArray(0) = ID

toolDetailsArray(9) = “false”

End if

For i = 0 to toolDetailsArray(0) -1

Copy record from inventory file to temporary file

Next

toolDetailsArray(8) = toolDetailsArray(8) + Notes

Append to temporary file (toolDetailsArray)

For i = toolDetailsArray(0) + 1 to end of file

Copy record from inventory file to temporary file

Next

Next

Delete original inventory file

Rename temporary file to name of original file

*// Find most recent tool in transaction log with that tool ID and mark as returned.*

For i = 0 to transaction file length

Set j = transaction file length – i

Set transactionDetailsArray = split line j with comma

If transactionDetailsArray(1) = ID

transactionDetailsArray(5) = true

For i = 0 to transactionDetailsArray (0) -1

Copy record from transaction file to temporary file

Next

Append to temporary file (transactionDetailsArray)

For i = transactionDetailsArray (0) + 1 to end of file

Copy record from transaction file to temporary file

Next

Break Loop

End if

Next