Evaluation

The programming language and tools used

To create the system, I used Visual Basic as it provided many features over other programming languages and, thanks to prior experience with this language, I knew I could exploit these features for the systems benefit.

Visual Basic uses a GUI (Graphical User Interface) which allows the use of graphical icons – such as menus, buttons, textboxes, etc. – for the user to interact with the system. This visualisation allows for ease of use as it is simple to learn to operate it, even for the most amateur computer users. This is a benefit over a programming language such as a command line as that is a much more complex language that requires a deeper knowledge of programming languages; which would not only cause issues for the users but also for myself when creating the system itself. This is because VB, unlike a command line, allows the use of arrays and data structures to easily save and manipulate a range of data types.

Furthermore, Visual Basic can use both serial and random data access. This meant that I could not only append data onto the end of a file easily using serial access but also easily access files within a large number of records because, unlike serial access, random access allows the system to not have to go through each file individually. This links to how VB allows for files to be saved and loaded externally; data is saved within simple text documents (txt format) which meant that if the system was to, for example shut down, then the saved data would not be affected.

Not only is VB capable of handling multiple data types such as strings, integers, reals and Boolean, but it also allows for the validation of these data types and variables. This was essential for the system as it ensured the system's robustness and the data's integrity. This meant that if the user was to input an incorrect value then the system would display an error message that informs the user how to correct their input. For example, if they were to enter a date of birth with an incorrect format (such as "1/4/1978") then the system would display "ERROR: Date of birth must be in the format **/**/****", not save the data and allow the user to try again.

Comparing the system to other existing systems

While conducting my desktop based research on booking system, I had an idea of what I needed to include based on what I found from other systems. What I found was limited to the following:

- Drop down boxes to select booking type
- Selecting date and time
- Gathering data such as name, phone number and email address

- Confirming the details

However, when it actually came to creating my own booking system, there was other data and inputs that were required. These included:

- Checking what staff members are available on those dates
- Job notes for the booking describing exactly what is required from the staff member
- Generating a unique ID for the booking
- A way of adding and removing tools to the tool array

Additionally, I then had to have a way of providing invoices once a booking was completed by a member of staff. These included the booking ID, date, staff ID, customer ID, price, address, and booking notes. This is quite similar to the existing booking systems that I found during my desk-based research but I could not find how these invoices are stored. However, I found an effective solution of storing the baseline data within the Invoices Masterfile then storing the detailed invoices within individual files which proved to work well.

Good features and shortcomings

Within the system, there was a range of strengths and weaknesses.

Firstly, the bookings weren't able to hold more than one staff member for each booking. This meant that either bookings were handled by only one staff member, or another work-around would have to be found; the issue with the work-around is that it would have to somehow be incorporated into the staff payroll. Similarly, another issue that the system has is that staff members are registered as unavailable for a whole day if they have a booking on that day. Luckily, as most bookings are for several hours, this doesn't have a massive impact but it is not ideal for a worker to only have 1 booking per day.

Alternatively, I worked hard to make my system as robust as possible. This is why I included many validation checks and "Try-Catch" sequences to minimise the possibility of the program not running. This included when adding data or, for example, when an integer is expected but instead a string is entered. This gives me the confidence to say that I effectively looked for solutions for my system.

Moreover, another issue with the system is that it only used linear searches and did not include binary searches. This decreased the efficiency of searches on large sets of data and makes it more time-consuming than it ideally should be. Speaking of efficiency, it would have been better to store the rate of pay within the staff text file rather than have the user enter it manually. However, comparing the stored rate of pay from the yearly payroll Masterfile with the entered rate of pay (which is what the system does) also works quite well.

A bigger strength of my system is that the searching forms for customers, if a two or more users have the same surname, uses a combination of the customer's surname and postcode to identify and load said customer's data (I did the equivalent for staff data). This is to, 1)

avoid the confusion of using IDs and 2) allow easy identification for same-surname data and thus provide a more convenient system.

Issues I had with the implementation

During my implementation of the system, I encountered some issues that lead me to create alterations compared to that of my investigation and design work. From my investigation, I replaced minimum and maximum stock levels with “reorder stock level” and “reorder quantity” as this allowed for more control of how much of an item is reordered and also made more sense as there is no reason to have a maximum stock level.

Furthermore, within the design, I combined the forms for “Searching Staff Logs” and “Searching Staff Logs by Date” into one form (and the equivalent for customer logs) as this allowed ease of usage rather than having one form that is basically the extension of another. More so, some new forms that I created that were not included within the design are “Search Bookings”, “Un-invoiced Bookings”, “Search Invoices”, “Search Monthly Payroll”, and “Search Yearly Payroll”. I added these because I thought that they were vital for the system to include as much useful functionality as possible.

Evaluation of aims

Below is a list of all of the aims and objectives of my system:

- Log in screen - Forgot password - Adding new customer and/or staff member (admin) - Adding new equipment (admin) - Editing customer or staff details (admin) - Editing booking details (admin) - Editing stock details/levels (staff) - Searching for a customer or staff member (admin) - Searching for booking/invoice details - Searching for tools and equipment (i.e. stock levels) - Searching for unpaid bookings/invoices (admin)	- Searching for workers logs (admin and the workers who’s details those are) - Delete a customer or staff member (admin) - Cancelling/postponing bookings - Placing bookings for customers by searching for available booking slots - Customers can pay for bookings/invoices - Calculating payroll - Showing when stock is nearing reorder levels - Logging out

Upon reflection of these aims, I feel as though I have accomplished all that I aimed to do:

The log in screen allows for the user to enter their details, function to use if they have forgotten their password, and the appropriate menu is opened if the details are correct. This

prevents any unauthorised access to the system and therefore decreases the likelihood of a security breach.

The system allows data to be appended to files and (in circumstances where necessary) places the new data into alphabetical/date order. Thanks to VB's use of arrays, the system does this by transferring the data from the arrays in place into the appropriate text files; these arrays allow for a variety of data types which gives the system more flexibility.

The system allows for the searching, editing, and deleting of items within the same form for every individual aspect (i.e. customers, staff, passwords, equipment, bookings) – payroll and invoices also do this, minus the deleting. The data is loaded from the external text files and the user is able to manipulate and save edits of any of the values. I feel as though this is a huge improvement to the paper-based system since it is less of a hassle to change old data.

Customers are able to place bookings through an admin user; this reduces the likelihood of miss-inputs as admin users will be more aware of what is and is not allowed within textboxes. The system checks if staff members are available on the selected day and, if not, then they are removed from the options of the booking. This prevents double bookings and upholds the company's integrity. After the booking is completed, an invoice is printed and handed to the customer which they can then pay for.

Staff has the ability to check which pieces of stock are nearing their stock reorder levels so that new ones can be ordered. They are also able to display all the equipment that requires testing; this allows for a better control and grasp over the company's stock.

Admin is able to generate the payroll for staff members. Payrolls are calculated based on the number of hours booked and not booked; staff are paid at a lesser rate for the hours that they are not booked. Individual monthly payroll text documents are created but the data is also appended to the appropriate yearly payroll Masterfile. If necessary, the user can then search for both the individual payroll documents or the yearly payroll data for a specific staff member.

Evaluation of testing

I put in a lot of time into testing the Handy Man Hiring's system to ensure that all the aspects of the system were as efficient and fully functional as they should be. Thanks to this I identified many errors which I would not have usually noticed and to which I reflected and provided improvements. I did this testing by first constructing a test plan for myself which I would then follow, I created this by making my way through the system and identifying where it could be exploited. This included testing buttons, input boxes, textboxes, verification, and validation.

However, due to time constrictions I was not able to complete all the testing I had initially planned and also meaning that I was not fully satisfied with the thoroughness of said testing. Most un-tested parts were minute, nevertheless, I did not test every validation instance; it was completed for adding customers, staff, equipment, and bookings but it was

not completed for searching for each of the previously listed. I do believe that they are functional as they use the same validation coding as the adding forms, but it is not necessarily a guarantee. This is something that I would approach differently if I was to redo the project.

Improvements that could be made

Firstly, as mentioned within "Good features and shortcomings", I would solve the issue of not being able to add more than one staff member to a booking. To do this I would use an input box that would allow the user to enter the number of additional staff (up to a limit of 3), ask the user to select the staff, then enter those staff members into a textbox (this would avoid any unnecessary textboxes). Furthermore, to allow a staff member to work more than one booking per day, I would make them available for placing bookings by giving them a 30-minute gap in between bookings and, if booking time is outside of this gap, displaying the staff member within the "Place booking" form.

A feature that I would add is the ability to copy bookings. For example, if a customer requests for duplicate bookings to repeat what they had done previously such as mowing the lawn, then the booking could be searched and copied which would open the "Place booking" form and duplicate everything from the searched booking into it. I think that this would be a convenient tool to have as it avoids the need to repeat the same entries multiple times.

Moreover, another improvement that I would make is, for logging in, to change using a clue to instead use a security question and answer. I feel that using a clue isn't the most appropriate way of assisting a user as a clue cannot always describe what the password is. I would replace it with an input box that displays the staff member's security question and requests them to enter the answer. If they are successful in logging in, the system asks them to contact an admin member and consider altering their password to be more memorable.

Finally, I would refine the system's messages by having each message box have custom titles depending on the situation (e.g. "ERROR", "Warning!", "Confirmation"). I would also make the message structures easier on the eyes – for example, if displaying a list of tools that need reordering, rather than doing a list using commas and sentences I would instead use bullet points. Another example would be to show the equipment that requires testing within one message box rather than individual message boxes to provide an ease of usage.

How would you change your approach in the future to avoid problems you experienced during this project?

If I was to produce a similar project to this one, there are several things I would do differently upon reflection.

Firstly, I would configure a plan of action for my implementation. This would ensure that I manage my time better and that I am able to keep up with deadlines. It would also assist me in visualising how much work I have left and an approximate time frame for each piece of work.

Secondly, I would make sure to investigate the documentation of the paper-based system more thoroughly to make certain that the new system isn't lacking any functionality compared to the original system. Likewise, I would communicate more with the manager of the company to ensure that the system is being created and requirements are being followed up to their expectations.

As well, I would make my aims and objectives more detailed. This would allow me to get a better picture of my next steps and to reflect back on the things that I've completed to see if there is anything I can improve or alter to follow said aims and objectives more appropriately.