A2 SSD COURSEWORK

Lakeside Escapes – Pottery Course System

DATABASE APPLICATION DEVELOPMENT

CANDIDATE NUMBER: 8380

CENTRE NUMBER: 71583

IF CODE WILL NOT RUN DUE TO MARK OF THE WEB, OPEN POWERSHELL AND TYPE: *dir -Path "[path here for project folder to unblock all files within the folder]" -Recurse | Unblock-File*

Introduction

Background

Lakeside Lodge is a privately owned luxury hotel set on a vast, lakeside estate. Recently, the owners have made strategic decisions to expand their business to include a new "entire holiday experience" and have begun to offer a range of weekend and 5-day breaks which includes glamping pod accommodation and a variety of skill-based activity courses.

While these courses are available, they are not mandatory, and the guests may book pods without a course. This being said guests who do participate in courses are a priority, as they are providing more income for the Lodge's new endeavours.

The new endeavour is managed by Ian Scott and is titled 'Lakeside Escapes' and is considered a separate business from the Lakeside Lodge. Ian is enthusiastic but unprepared for the challenge and has become aware that the lack of a computer system to assist him as the business expands is a large problem.

'Lakeside Escapes' consists of 10 standard and 6 luxury glamping pods (however one standard pod is used for a Yoga course), a pottery studio, and a painting studio, and also makes use of function rooms within the Lakeside Lodge Hotel. Guests may also pay to use the hotel facilities.

Ian also tries to produce the programme of options and activities a year in advance, the programme of which commences at the end of January. The busiest times are from Easter to Halloween, and December is also popular for craft courses. He tries to optimise the occupancy of the pods as much as possible.

Currently, all of Ian's documentation is physical, and he uses diaries to manage the bookings and billings as well as the stock of all courses. This is disorganised and repetitive and he commonly makes mistakes. It is also very time-consuming.

Pod Booking

The pods can be booked for a period of two, four, six or thirteen nights. No bookings are taken between the 20th of December and the 20th of January. Guests can make a provisional booking which can be held for three days and is only confirmed when a 50% deposit has been paid. If this is not paid on time, the dates will be made available to other guests.

If a guest has booked a pod on a previous occasion, the guest receives a 2% discount, and booking up to 6 months in advance results in a 3% discount. These are applied when the final bill is paid. Payment must be made by card.

All bookings must be made 2 months in advance of the dates to facilitate course scheduling. Currently, all information is stored in a physical diary, and the only way to determine the availability of courses and pods is by looking at this diary. It is messy and disorganised, making it very hard to visualise the availabilities at just a glance. Because the course bookings are then subsequently recorded in another diary, a lot of data is duplicated, and sometimes not filled out with information correctly, making it problematic to organise the invoice for the end of the guests' stay.

The guests' bills must be settled and paid before they sign out. In the current state, the invoices are not available as they should be, and can take some time to prepare. The calculations are manual and often mistakes are made. This prolongs the checkout process.

Yoga and Meditation

One of the available courses is Yoga and Meditation. There are eight weekend courses and eight five-day courses in the annual schedule, with one weekend and one five-day course exclusive for each skill level (Beginner, Intermediate and Advanced) and the rest open to a mixture of capabilities.

Each course can accommodate 25 guests, but these may be split into smaller groups for more advanced sessions. On five-day courses, other activities may also be available, for example, individual sessions, instructional videos, health and nutrition lectures, or cookery classes. The guests must pay for the ingredients.

Painting

Painting is the second course available to guests at 'Lakeside Escapes'. There are six weekend and eight five-day courses per annum. Two weekend courses are exclusive to each skill level. One five-day course is exclusive to each skill level, and the rest is dedicated to specific themes such as landscape, still life, sketching or portrait.

The weekend courses can be equipped for 12 guests, while the five-day courses can take up to 18. The classes include lectures on techniques and mediums, demonstrations, and opportunities to experiment. The five-day courses expand further into individual sessions, specialist sessions, and the opportunity to paint in a variety of locations on the estate.

There is also an optional guest speaker event, which is open to all guests and has ticket entry. The speaker may also provide feedback on the guest's work. Guests taking this course will get a 50% discount on entry to the event.

Finally, for five-day courses, a day trip may be planned to a location of artistic interest in the local area. Guests must pay for transport.

Guests will obviously require a lot of equipment. Use of crucial equipment is already included in the course fees, but minor equipment must be recorded by the Tutor and added to the guests' bills at the end of their stay. This is currently an inaccurate process, as not only is stock taken and recorded manually, but it is also difficult for the Tutor to keep track of what is used during the classes. There have been many occasions where there have been insufficient resources and the guests have been dissatisfied. This is less than ideal.

Pottery

Pottery is the final course open for the guests of 'Lakeside Escapes'. There are ten pottery weekends and six five-day courses. The weekends are reserved for Beginner (five weekends) and Intermediate (five weekends) skill levels, as a longer time frame is required for the more advanced skill levels to explore higher-level practices. There are three advanced and three intermediate five-day courses.

Weekend courses can accommodate 14 guests, and five-day courses have a maximum of 16 guests. Not only do the five-day courses include the development of a piece of work, but they also include individual tuition, specialist sessions, and a visit from a local sculptor. Participation in these extra options must be specified in advance, and they must coincide with the guests' free time.

Inventory needs to be kept of the studio's equipment, but the only extra charges are for additional materials such as extra clay, glazes and paints. A basic tool kit is provided. As is with painting, this equipment

is hard to manually maintain and monitor, and there is often insufficient equipment. Ian is considering combining the pottery and painting inventory.

Additionally, for those in the advanced class, there will be an exhibition of work where the guests may sell their artwork, and the best piece, judged by the tutor, the guest artist and the Scott family, will win a free weekend at the hotel. An audience must be invited, and Ian must ensure that there is catering for correct numbers, the prices on the artwork are correct, and that the sale money reaches guests whose artwork sells.

Workshops

Ian has also managed to include one-day workshops for painting and pottery, as well as allocating successive Saturdays for the completion of work. These sessions are especially popular with children in December.

Currently, Ian does not have a proper way to book these workshops, and while they have availability for 14 people, the classes are often overbooked, and the participants are unamused. Since the addition of a second day for completion, this problem has severely escalated, with the bonus issue of needing somewhere safe to keep participant's work until the second day and maintain contact with them.

If Ian wishes to continue this, he will need a way to manage booking, stock, scheduling, storage and billing more efficiently.

General Problems with Existing System

- The records for every part of Lakeside Escapes are currently all physically handwritten in a variety of books, which is inefficient and full of data duplication and human errors.
- If information is changed somewhere, or something is cancelled, the information may not be removed from all the records as it is hard to keep track of so many individual physical books.
- The calculations within the business are always done manually which is slow and prone to mistakes.
- Bills are created at the end of a guest's stay manually, meaning the guest is stuck at checkout for a long time, as the process is delayed and full of issues due to mis-recorded information.
- Inventory is not correctly managed for any course. There are often missing resources as the tutors have not been given a specific way to keep track of resource use.

Focus Area: Option 3 - Pottery

The system being developed for 'Lakeside Escapes' will focus on the third course option - Pottery. Currently, the systems for booking, inventory and billing are entirely paper-based. The system will concentrate on creating a database application to manage booking and billing for the pottery course, as the inventory is more closely linked to the painting option.

The system will incorporate a subsystem to help Ian organise the exhibitions, ensuring he can manage what the prices for each of the pieces are, what items have sold, and that the guest has received the payment much more efficiently than he currently does.

Ian is finding that his diaries are complicated and disorganised, and it is easy to overbook. The system in development will help avoid

overbooking the pottery course and the classes within it as well as make it much easier for Ian to separate guests into skill levels and class types.

Ian also needs the system to assist him with billing, so I intend for the system to include a way for all his calculations to be automatic, reducing the errors he has been making up to this point. The system should be able to create reports with the billing information for the pottery course specifically for each guest, which will mean preparing invoices for the guests signing out will have much less of a delay.

While the system will not include an inventory system, it will feature a method for the tutor to record extra resources used by the guests, which can then be added to their billing report at the end of their stay.

Problems within the Focus Area

- The entirely **paper-based system** is **extremely confusing** and hard to decipher because it is handwritten.
- Having **multiple different books** to store **handwritten data** results in **data duplication** and lots of mistakes due to **human error** when copying information.
- **Resources** used by the guests are **constantly mis-recorded** or just not recorded at all because there is **no set system** for the tutors to use.
- **No standard for formatting invoices** results in confusion and missed information, therefore **bills are often incorrect** and missing costs.
- **Bills are put together by hand** resulting in **delays** in generating bills for guests.
- All **calculations are done manually**, which naturally means lots of **human errors**.
- There is no standard method for keeping **records of sale prices** for the guest's art pieces during exhibitions, or if the piece has sold. This causes **confusion when transferring the money** back to the guests.
- There is no way to keep an accurate record of how many need to be catered for at exhibitions.

Client

For the system being developed, Ian Scott is the client. Scott needs to be able to book pottery courses, store guest details, accurately bill guests and create invoices for the course based on the equipment they use and organise the prices and payments to and from the advanced guests for their pieces in the exhibitions.

Users

The most frequent users will be receptionists or secretaries. These people will be using the system to book pottery courses, create and keep record of guests, keep track of artwork, manage exhibitions, and create bills for each pottery course booking.

The pottery course tutors will also be users of the system. The tutors will need to be able to view the guests and courses in order to prepare for a class, as well as keep an accurate record of the resources used by each guest in their course.

Ian Scott may also be a user of system, in order to view details of the courses, guests and exhibitions throughout the year.

Evaluation of Methodologies

Waterfall Model

Waterfall is a traditional method that is **performed sequentially** from planning to implementation, with each needing to be completed before the next stage. This may be the best methodology for the Lakeside Escapes system as it is **best suited to projects with well-defined requirements** and strict deadlines, as Ian from Lakeside Escapes is already very sure of the system he requires for his pottery courses.

Ian also requires **documentation** to help him understand the system, as he is moving from entirely paper-based. This documentation is a crucial part of **Waterfall**.

One disadvantage of **Waterfall** is the functional system is **delivered late** in the life cycle. This may be problematic as Ian will not be able to give **constructive feedback** to ensure the pottery course system meets his needs.

Rapid Application Development (RAD)

RAD is similar to Waterfall, but functions more iteratively, allowing **prototypes** for the client to give constructive feedback at different stages during the process. This could be helpful for Ian as he may not fully understand what the system is capable of, as he is moving from a completely paper-based system.

RAD allows for a much **shorter lifecycle** and enables **rapid delivery**, which could be suitable for the Lakeside Escapes system as **Ian requires a solution for the pottery course quite urgently**, due to the increasing disorganisation.

RAD, however, may not be suitable for Ian, as he requires a **robust and reliable** system, and the focus within **RAD** is on visual prototypes rather than architectural design, which can result in **poor documentation and programming standards**.

Dynamic Systems Development Method (DDSM)

DSDM is an **agile methodology** in which the requirements can be defined at a high level initially and could be suitable for Lakeside Escapes, as Ian is fairly **confident in his requirements**. It is also best used where the **technology is more than a year old**, which is perfect for Lakeside Escapes, as Ian is moving from **entirely paper-based** to a simple **database application**.

On the other hand, **DSDM** is also not suitable for the system, as a key feature is the fact **performance and reliability are not critical**. This is undesirable for Lakeside Escapes, as a booking system **needs to be robust and reliable so that all the pottery courses run smoothly**.

Scrum

SCRUM is another **agile methodology** that is extremely **similar to** DSDM and is, therefore, suitable for Lakeside Escapes in the same ways DSDM is. The difference comes in the way **requirements are prioritised**; in **SCRUM** if the project is **running behind**, 'should have' and **'could have' requirements will be abandoned**, with a focus on 'must have'. This makes **SCRUM** less suitable than DSDM for the Lakeside Escapes, as **Ian's requirements for the pottery course system are mostly 'must have'**.

SCRUM also does not include a **format for requirement gathering**. The **requirements** for Lakeside Escapes are **already well defined** by Ian, so this methodology is **very unsuitable.**

Extreme Programming (XP)

Possibly the most radical form of **agile**, **XP** is distinctly different to all other methodologies, as it is **geared solely to the user requirements**. It is iterative and involves **prototyping**, meaning it may be suitable for Lakeside Escapes, as there is an **opportunity for Ian to be involved with the process** along the way, ensuring the pottery course system is what he needs.

A disadvantage, however, could be the lack of a clear picture within XP. This methodology is **designed to be planned and developed gradually**, which is unsuitable for Lakeside Escapes, as Ian is already confident in what he needs the system for the pottery course to be. Another disadvantage is that XP requires a larger volume of developers, for **pair programming**. This is **not a possibility**, making XP **completely unsuitable** for the pottery course system.

Justification of Methodology Choice – Waterfall

The most suitable methodology for the pottery course system is definitely the Waterfall Model. **The Waterfall Model** is best used when the **User Requirements are clear** and unlikely to change because they can be defined at an early stage. Lakeside Escapes are very **sure of what they require for their pottery course system**, so this method might apply well.

Waterfall is also best used when the **technology is well understood**. Lakeside Escapes are very behind the times with their paper-based systems, and the **database tools** used to develop the application they need are **well established**. This means waterfall is the perfect choice.

Waterfall is also the most suitable choice when **detailed documentation** is required. This kind of system is new to Ian, so he may need lots of documentation to help him and his staff understand how to use the system to both book and bill the guests in the pottery course.

The **Waterfall** method is the most efficient method when **reliability is important**. This would be helpful for Lakeside Escapes, as **unreliability would cause catastrophic roll-on effects within the business**, as pottery course bookings, schedules and invoices would become lost or botched. This could result in a significant loss of profit and many unsatisfied guests, which would be serious seeing as the effects of the pandemic are already causing trouble for the parent business, Lakeside Lodge.

Finally, **Waterfall** is the best for **inexperienced developers**. As this is my second large project, I feel this is the methodology that will be the best fit for myself as a developer and this system as a whole.

Project Planning

Tasks and Durations

Task A –
Develop and Entity Relationship Diagram, Normalise Data and draft Tables for the Database.
Duration: 3 Days

Task B –
Create the Database in Visual Studio and add in the Tables and Data types.
Duration: 1 Day

Task C –
Create Relationships and Constraints in and between the Tables in the Database.
Duration: 2 Days

Task D –
Design and implement the frontend of the login screen and the menus.
Duration: 1 Day

Task E –
Develop the backend of the login and menus and link them together. Include Maintenance popups for all options other than the focus area.
Duration: 1 Day

Task F –
Design and implement the frontend of the Guest Management form, including the triggers for Add, Edit and Delete modes. Link it to the menus.
Duration: 2 Days

Task G –
Develop the backend of the Guest Management form, including stored procedures for queries used by the form and linking it to the Database.
Duration: 5 Days

Task H –
Design and implement the frontend of the Course Management form, including the triggers for Add and Delete modes, and link it to the menus.
Duration: 1 Day

Task I –
Develop the backend of the Course Management form, including the stored procedures for queries used by the form and linking it to the Database.
Duration: 4 Days

Task J –
Design and implement the frontend of the Equipment logging form and link it to the menu.
Duration: 1 Day

Task K –
Develop the backend of the Equipment logging form, including the stored procedures for the queries used linking it to the Database.
Duration: 2 Days

Task L –
Design and implement a report to show their participants booked into a course and identify if it is fully booked. Link it to the Course management menu.
Duration: 3 Days

Task M –
Design and implement the frontend of the Place Booking form and link it to the menus.
Duration: 1 Day

Task N –
Develop the backend of the Place Booking form including the stored procedures for the queries used in the form and linking it to the Database.
Duration: 2 Days

Task O –
Design and implement the frontend of the Artwork Management form including the triggers for add, edit and delete mode, and linking it to the menus.
Duration: 1 Day

Task P –
Develop the backend of the Artwork Management form including the stored procedures for queries used by the form and linking it to the Database.
Duration: 6 Days

Task Q –
Design and implement the frontend of the Exhibition Management form, including the triggers for Add, Edit and Delete modes and link it to the menus.
Duration: 1 Day

Task R –
Develop the backend of the Exhibition Management form including the stored procedures for the queries used by the form and linking it to the Database.
Duration: 4 Days

Task S –
Implement a search feature on each of the management forms to make finding records easier.
Duration: 4 Days

Task T –
Implement a menu strip item on all forms that allows faster navigation between forms.
Duration: 2 Days

Task U –
Design and implement an invoice report that calculates the guest's booking charges and link it to the guest menu.
Duration: 7 Days

Task V –
Create easy to understand user guide and link it to help sections in the menu strip.
Duration: 4 Days

Task W –
Construct a thorough test plan table including test data and space for results and actions.
Duration: 5 Days

Task X –
Perform testing with reference to the test guide, noting results and where action is required.
Duration: 3 Days

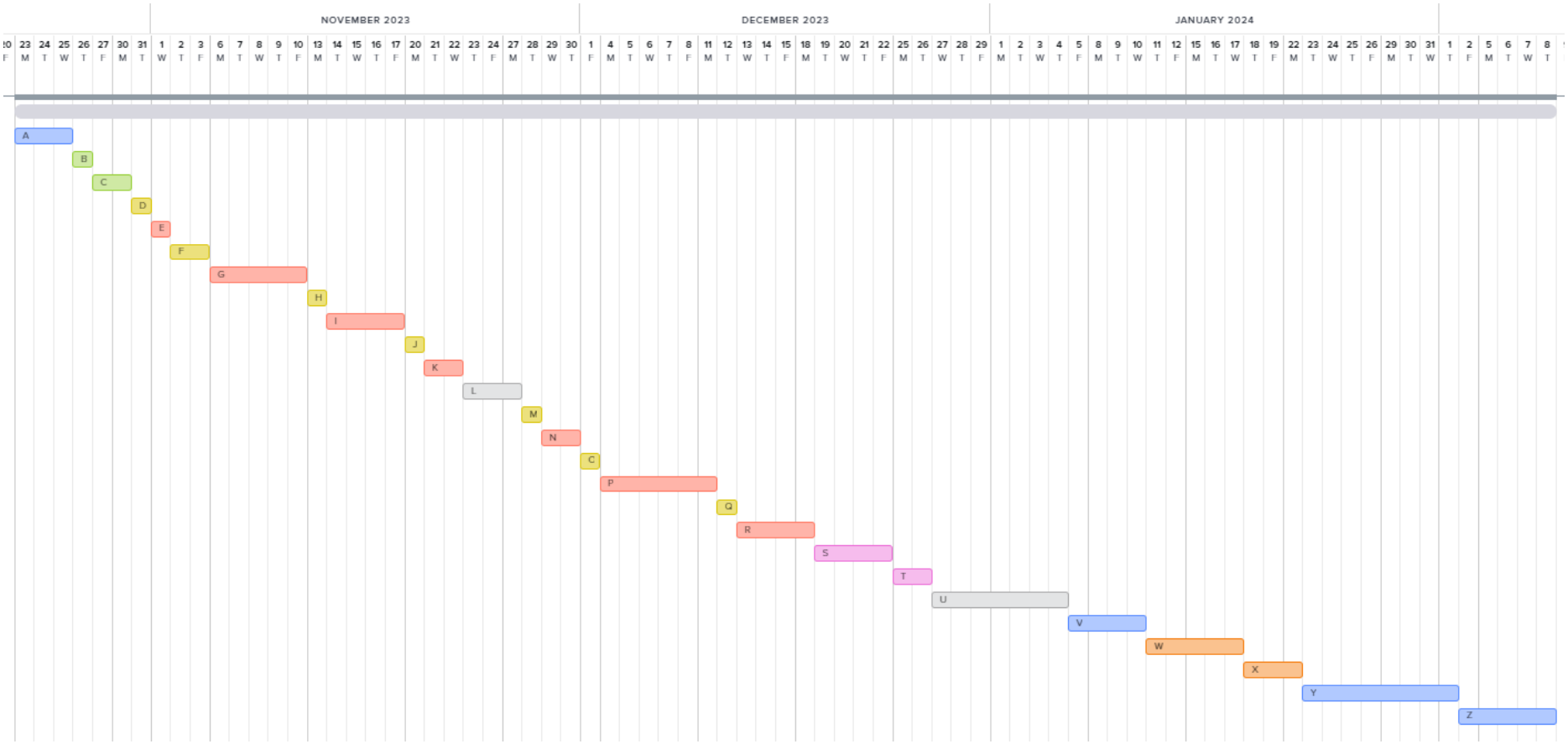
Task Y –
Evaluate and resolve all test actions, documenting resolutions and linking them to the test plan table via hyperlinked bookmarks.
Duration: 8 Days

Task Z –
Evaluate completed solution and ensure it is ready for deployment.
Duration: 5 Days

Resources

- Visual Studio 2022
- RDLC Report Designer 2022
- Microsoft SQL Server 2022
- Draw.io (planning and drawing diagrams)
- Notion (activity and issue logging)
- ibisPaint (storyboarding)
- Synology FileStation (sharing files between home and class)
- GitHub (enabled working on different devices at once, and only push changes when they work)

Gantt Chart



Blue – Documentation | Green – Database | Yellow – Frontend | Red – Backend | Grey – Reports | Pink – improvements and additions | Orange – Testing

User Requirements

Functional User Requirements

Essential Requirements

1. The system requires a menu strip to help the user navigate the system and menus. This will include a logout, exit and help button, plus the ability to open other forms quickly.
2. The system should have easy to understand menu that can implement the other systems being developed with ease.
3. The user should be able to manage existing and new guests of the pottery course:
    - 3.1. The user needs to be able to view all existing guests and their details.
    - 3.2. The user needs to be able to enter a name or part of a guest name and view all guests whose name includes the entered characters.
    - 3.3. The user needs to be able to sort guests by skill level or country.
    - 3.4. The user needs to be able to add a new guest and their information as and when required.
    - 3.5. The user needs to be able to delete a guest, but only when that guest has no existing artwork.
    - 3.6. The user needs to be able to change a guest's details with ease, but keep the same ID.
    - 3.7. If editing a guest's details, you should not be able to change the skill of someone with a future booking, as courses have specific skill types.
    - 3.8. If a user chooses to delete a guest who has attachment to courses, their bookings, past and future, should also be deleted.
    - 3.9. The user should be able to click on a record when deleting or updating to fill in information in order to mitigate error.
    - 3.10. When sorting by country, the format of addresses should be verified using a custom exception to avoid crashing the system.
4. The user needs to be able to manage existing and new pottery courses and the information surrounding them:
    - 4.1. The user needs to be able to view all existing pottery courses.
    - 4.2. The user needs to be able to enter a course ID, and get a report showing the course, and the names and phone number of the guests enrolled in it, as well as the date the booking was made.
    - 4.3. The user needs to be able to enter two dates and view all courses between those dates.
    - 4.4. The user needs to be able to create new pottery courses, but only if there is space for it in the current year.
    - 4.5. The user needs to be able to delete existing pottery courses, but only if there are no guests assigned to this course.
    - 4.6. The user needs to be able to assign a guest to a pottery course, but only if the course has space, to avoid overbooking, and the guest's skill level matches the course. Double booking will be handled by this also.

4.7. The booking cannot be made between December 20th and January 20th.
4.8. The system should automatically filter out courses that are not suitable to the guest being booked.

5. The user needs to be able to manage existing and new pieces of artwork:
   5.1. The user needs to be able to view a record of all existing artwork, and the information about it.
   5.2. The user needs to be able to search for an artwork ID, and show the Artwork ID, Guest name, and price, as well as whether it has sold or not. The price should be deducted from the Guest's first booking after the exhibition.
   5.3. The user needs to be able to create records of a piece of artwork but only if there is a guest to be associated with it at the time of creation, and that guest has been booked into at least one course.
   5.4. When an artwork is created for an advanced guest, it should be automatically added to the nearest future exhibition.
   5.5. The user needs to be able to delete a piece of artwork. If it is deleted, it should be cascade deleted from all other places. If it is linked to a past exhibition, it cannot be deleted.
   5.6. The user needs to be able to edit the price of an advanced artwork.
   5.7. The user needs to be able to change an artwork's exhibition, but only if the artwork was created by an advanced guest, has not sold, and the new exhibition has not happened.
   5.8. The system should force the user into the advanced view when in update mode, as advanced artworks are the only ones with details that can be edited.
6. The user needs to be able to manage details about existing and new exhibitions:
   6.1. The user needs to be able to view details about each exhibition, including the artwork assigned to it.
   6.2. The user needs to be able to create an exhibition, but it should be limited to one exhibition annually.
   6.3. The user needs to be able to delete an exhibition but only if the exhibition has no artwork assigned to it, and the date has not yet passed.
   6.4. The user needs to be able to update an exhibition's date and catering number.
   6.5. The system should disable the past and all artwork views when in update or delete mode, as past exhibitions, cannot be deleted or edited.
7. The user needs to be able to log what equipment a guest uses, and it should be specific to the current course.
8. The user needs to be able to print off an invoice including the guest's course cost and the added cost of equipment used. This should be an automatic calculation. If the guest sold artwork in the most recent exhibition, and this is their first course their selling price should be deducted from the bill.
9. The system needs to ensure all data entered into the database is entered correctly and attempt to correct it or ask for re-entry if it is not.
10. The user should be able to log into the system with a username and password for security. There should also be a password visibility button.

Non-Essential Requirements

11. The user should not be able to enter letters or symbols into phone number or ID fields. The field should also automatically contain the letter that precedes the code or the + before a phone number, and this should be unable to be deleted by the user.
12. In the management of artworks, the user needs to be able to search for artworks by a specific guest ID.

Non-Functional User Requirements

13. The client needs the pottery system to be intuitive and easy to use as he is moving from a paper-based system. It should include a help document to demonstrate how to use the specific aspect of the application.
14. The system should be secure and comply with GDPR Standards.
15. The colour scheme needs to be clean, consistent and simple.
16. The application should have the company logo as the icon on every form.
17. The system should have a consistent house style.
18. The system should be suitably fast and reliable.

System Design and Planning

Normalisation of Data

Unnormalised Data – 0NF

CourseID, CourseType, StartDate, CourseSkillLevel, CourseCost, CourseCapacity, FullyBooked?, GuestID, GuestForename, GuestSurname, GuestAddress, GuestContactNo, GuestSkillLevel, DateBooked, TimeBooked, ItemID, ItemCost, NumberUsed, EquipmentCost, TotalItemCost, ArtworkID, SalePrice, Sold?, ExhibitionID, ExhibitionDate, CateringNo

Derived Fields: FullyBooked?, TotalItemCost, EquipmentCost

1st Normal Form

Removing Repeating Groups and Non-Atomic Attributes. Calculated/Derived Fields are also ignored from this point forward.

Course_Guests [CourseID, GuestID, CourseType, StartDate, CourseSkillLevel, CourseCost, CourseCapacity, GuestForename, GuestSurname, GuestAddress, GuestContactNo, GuestSkillLevel, DateBooked, TimeBooked]

Exhibition [ExhibitionID, ArtworkID, GuestID, SalePrice, Sold?, ExhibitionDate, CateringNo]

Guest_Equipment [CourseID, GuestID, ItemID, ItemCost, NumberUsed]

2nd Normal Form

Remove Partial Key Dependencies.

Course [CourseID, CourseType, StartDate, CourseSkillLevel, CourseCost, CourseCapacity]

Guest [GuestID, GuestForename, GuestSurname, GuestAddress, GuestContactNo, GuestSkillLevel]

Guest_Booking [GuestID*, CourseID*, DateBooked, TimeBooked]

Equipment [ItemID, ItemCost]

Guest_Equipment [GuestID*, CourseID*, ItemID*, NumberUsed, TotalItemCost]

Artwork [ArtworkID, SalePrice]

Guest_Artwork [ArtworkID*, GuestID*]

Exhibition [ExhibitionID, ExhibitionDate, CateringNo]

Exhibition_Artwork [ExhibitionID*, ArtworkID*, Sold?]

3rd Normal Form

Remove Transitive Dependencies – ***no change from 2NF***

Course [CourseID, CourseType, StartDate, CourseSkillLevel, Course Cost, Capacity]

Guest [GuestID, GuestForename, GuestSurname, GuestAddress, GuestContactNo, GuestSkillLevel]

Guest_Booking [GuestID*, CourseID*, DateBooked, TimeBooked]

Equipment [ItemID, ItemCost]

Guest_Equipment [GuestID*, CourseID*, ItemID*, NumberUsed, TotalItemCost]

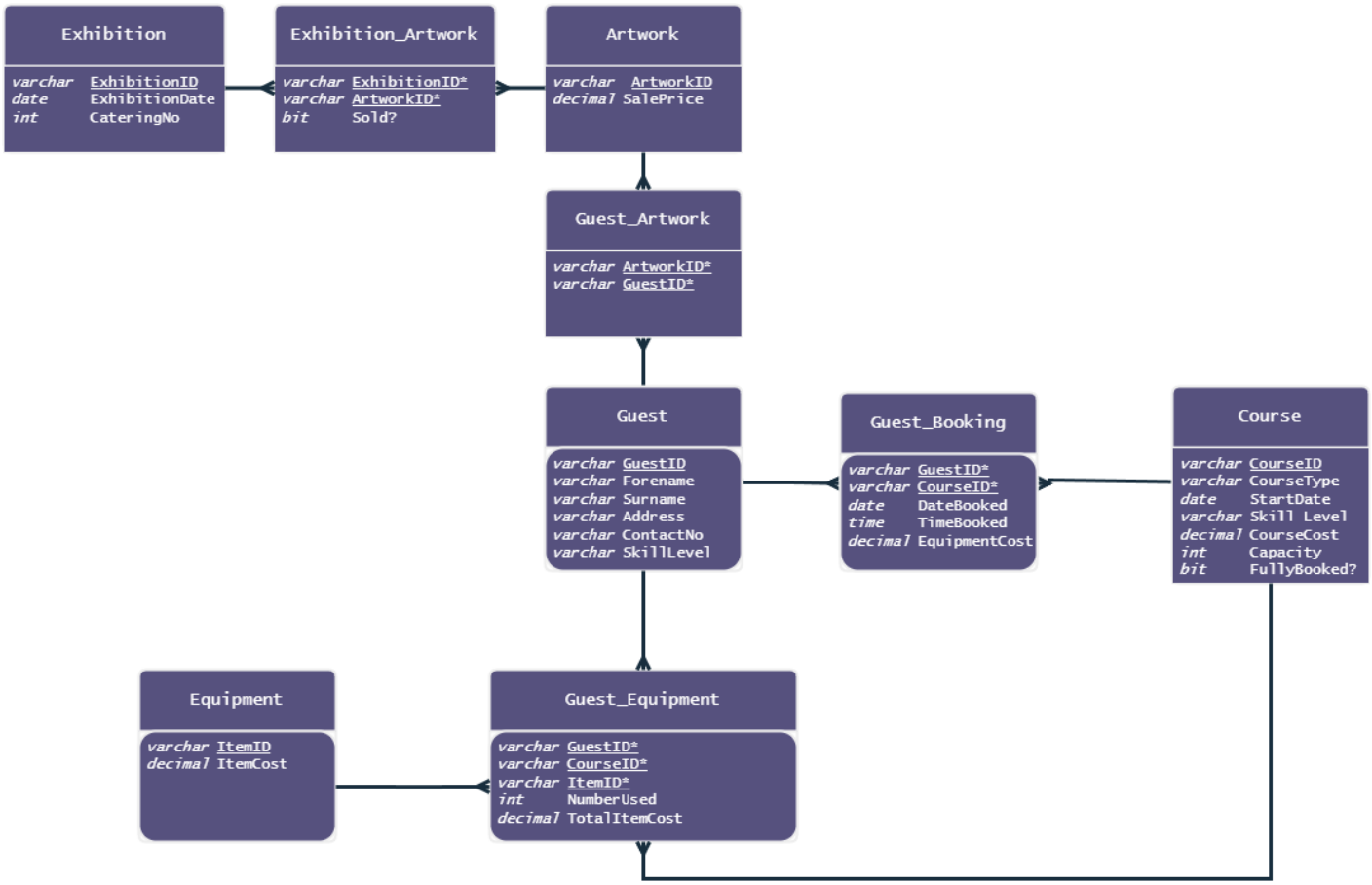
Artwork [ArtworkID, SalePrice]

Guest_Artwork [ArtworkID*, GuestID*]

Exhibition [ExhibitionID, ExhibitionDate, CateringNo]

Exhibition_Artwork [ExhibitionID*, ArtworkID*, Sold?]

Entity Relationship Diagram



Entities

Exhibition –
This entity will contain the attributes of the Exhibitions from **Requirement 6.** Its cardinality is a one-to-many relationship with **Exhibition_Artwork**, a junction, or weak, entity that allows **Exhibition** to have a many-to-many relationship with **Artwork.**

Artwork –
This entity will contain the attributes of the Artworks created by Guests in **Requirement 5.** Its cardinality is a one-to-many relationship with the weak entities **Exhibition_Artwork** and **Guest_Artwork.** These allow **Artwork** to have a many-to-many relationship with **Exhibition** and **Guest.**

Guest –
This entity will contain the attributes of the Guests taking part in the pottery courses, from **Requirement 3.** Its cardinality is a one-to-many relationship with junction entities **Guest_Artwork, Guest_Booking** and **Guest_Equipment,** which then allow **Guest** to have a many-to-many relationship with **Artwork, Course** and **Equipment.**

Course –
This entity will contain the attributes of each Course that can be booked by guests, from **Requirement 4.** Its cardinality is a one-to-many relationship with both junction entities **Guest_Booking** and **Guest_Equipment.**

These allow the **Course** to have a many-to-many relationship with both **Equipment** and **Guest.**

Equipment –
This entity will contain the attributes of each possible piece of extra equipment that can be used by the guests as seen in **Requirement 7.** Its cardinality is a one-to-many relationship with the weak entity **Guest_Equipment.** This allows the illusion of a many-to-many relationship with both **Guest** and **Course.**

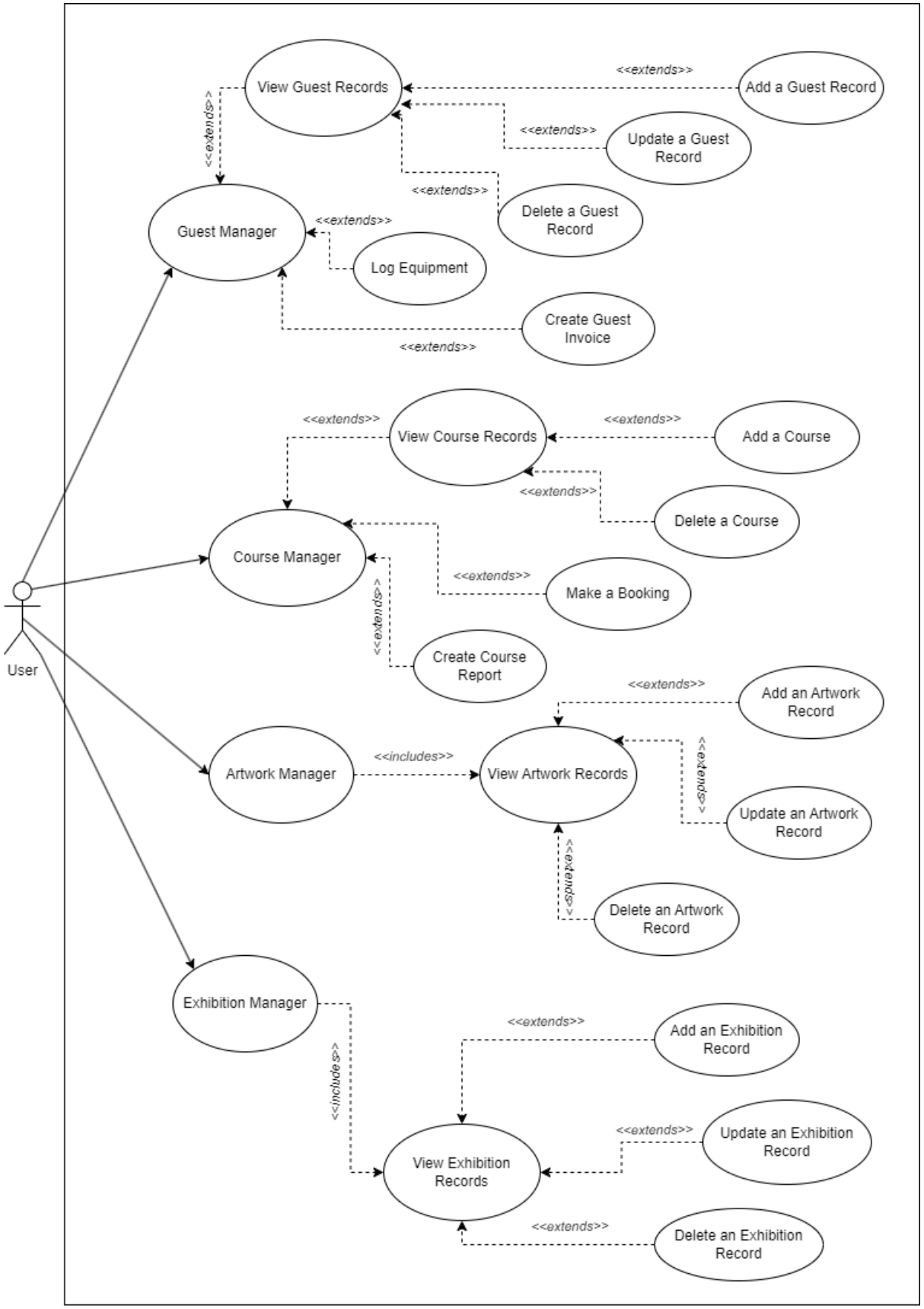
Data Dictionary

Field Name	Data Type	Description	Length	Validation	Example of Data
Exhibition					
ExhibitionID	Varchar	Primary key identifier for exhibitions	15	Must be unique and not null	EX001
ExhibitionDate	Date	The date an annual exhibition will take place	10	maximum one per year, not null	01/12/2006
CateringNo	Int	The number of people who need catered for at an exhibition	Max 2,147,483,648	Must be positive	77
Exhibition_Artwork					
ExhibitionID	Varchar	Primary key identifier for exhibitions	15	Must be unique and not null	EX001
ArtworkID	Varchar	Primary key identifier for artworks	15	Must be unique and not null	A0808
Sold?	bit	Has the artwork been sold at an exhibition	1	0 or 1, not null	0
Artwork					
ArtworkID	Varchar	Primary key identifier for artworks	15	Must be unique and not null	A0808
SalePrice	Decimal	The sale price of an artwork	10,2	Must be positive	55.00
Guest_Artwork					
ArtworkID	Varchar	Primary key identifier for artworks	15	Must be unique and not null	A0808
GuestID	Varchar	Primary key identifier for Guests	15	Must be unique and not null	G2240
Guest					

GuestID	Varchar	Primary key identifier for Guests	15	Must be unique and not null	G2240
Forename	Varchar	Guest's Forename	50	Not null	Johnny
Surname	Varchar	Guest's surname	50	not null	Knoxville
Address	Varchar	Guest's full address, separated by commas	15	Three parts, separated by commas, not null	1447 Wright Road, Los Angeles, USA
ContactNo	Varchar	Guest's contact number	50	Numeric with + at start, not null	+4075073226
SkillLevel	Varchar	Guest's skill level	15	Must be either Advanced, Intermediate or Beginner, not null	Advanced
Guest_Booking					
GuestID	Varchar	Primary key identifier for Guests	15	Must be unique and not null	G2240
CourseID	Varchar	Primary key identifier for Courses	15	Must be unique and not null	C020
DateBooked	Date	The date a booking is placed	10	not null, date only	01/02/2006
TimeBooked	Time	The time a booking is placed	8	not null, time only	16:24:00
EquipmentCost	Decimal	The cost of equipment used, calculated.	10,2	Must be positive	67.00
Course					
CourseID	Varchar	Primary key identifier for Courses	15	Must be unique and not null	C020
CourseType	Varchar	The type of course	8	Must be either five-day or weekend, not null	Five-Day
StartDate	Date	The start date of the course	10	not null, date only	22/02/2006
SkillLevel	Varchar	The skill level of people attending the course	12	Must be either Advanced, Intermediate or Beginner, not null	Advanced

CourseCost	Decimal	The base cost of the course	10,2	Not null, must be positive	400.00
Capacity	Int	The maximum number of attendees	Max 2,147,483,648	Not null, must be positive	16
FullyBooked?	bit	Whether the course is full or not	1	1 or 0, not null	1
Guest_Equipment					
GuestID	Varchar	Primary key identifier for Guests	15	Must be unique and not null	G2240
CourseID	Varchar	Primary key identifier for Courses	15	Must be unique and not null	C020
ItemID	Varchar	Primary key identifier for Items	15	Must be unique and not null	I5002
NumberUsed	Int	How many units have been used	Max 2,147,483,648	not null	8
TotalItemCost	Decimal	The total cost of one type of equipment used	10,2	Not null	32.00
Equipment					
ItemID	Varchar	Primary key identifier for Items	15	Must be unique and not null	I5002
ItemCost	Decimal	The cost of one unit of an item	10,2	Must be positive, not null	4.00

Use Case Diagram



System Story-Boarding

Initial Plan Client Feedback and Improvements

My initial design was functionally similar to the final design. A variety of menus would lead into each form used to manipulate the database. Each section would include a different form for adding, editing and deleting records. When I brought this design to my client, they suggested that this was functionally good, but clunky and not very user friendly, as it was very irritating to have to return to a menu in order to update or delete a record you had just added.

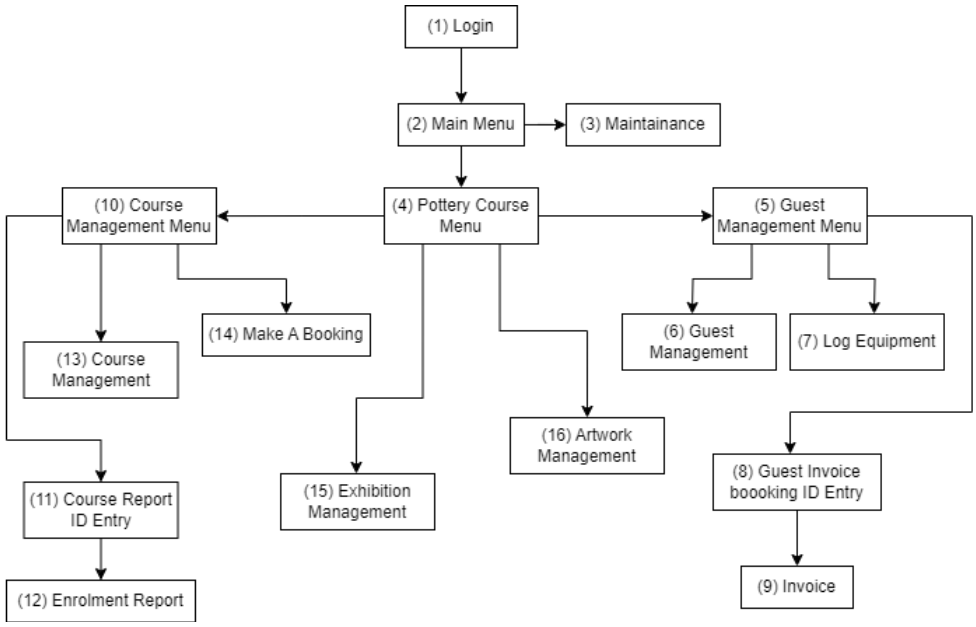
My initial design also featured a View Records form, meaning to search for a record, you would have to search through that view, remember the ID, and go back to the edit or delete forms. I realised myself that this was awkward and not very useful.

Both these also meant there would be many more forms, which could even slow down the response times of my application.

In my designs below you can see my solution was to compress all of these issues into one form. On each management form, you will be able to switch between add, edit and delete 'modes' using a button. Each form will also allow you to view and sometimes search through records. Selecting a record while in an editing mode will autofill details.

I believe these are all solutions that will greatly increase the usability of my system.

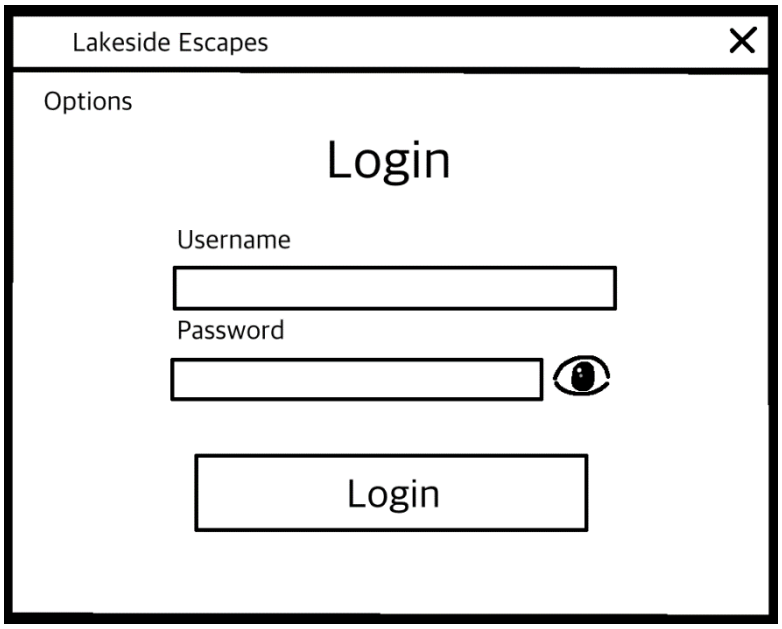
Final Design



House Style

For my solution I have decided upon a house style. For usability and simplicity, I have decided to go with a neutral colour scheme of blacks, greys, and whites. In Data Grid Views, I will modify the selections, so they highlight in grey instead of blue to fit in with the rest of the forms. Important information on forms will be highlighted in red. Every form will use the font family Yu Gothic, as it is clean and easy to read in both large and small font. Each form will feature the Lakeside Escapes Logo as the form icon. All forms will have no minimise or maximise button and will be centred to the screen.

(1) Login Page



Controls and Events

1. **Menu Strip –**
   This menu strip will allow the User to **hover** the mouse over the word **Options**, triggering a **hover event** to open a sub menu that will consist of the options 'Exit' or 'LOGIN DETAILS'. Exit will have a **Click event** that will open a **message box** where the user can confirm that they want to exit. Confirming will **close** the whole application. Denying will **close** the **message box**. LOGIN DETAILS will open a **message box** that tells the user the login required to enter the application. In a real system this would not exist, it is added to aid testing and marking.
   This satisfies requirement 1.

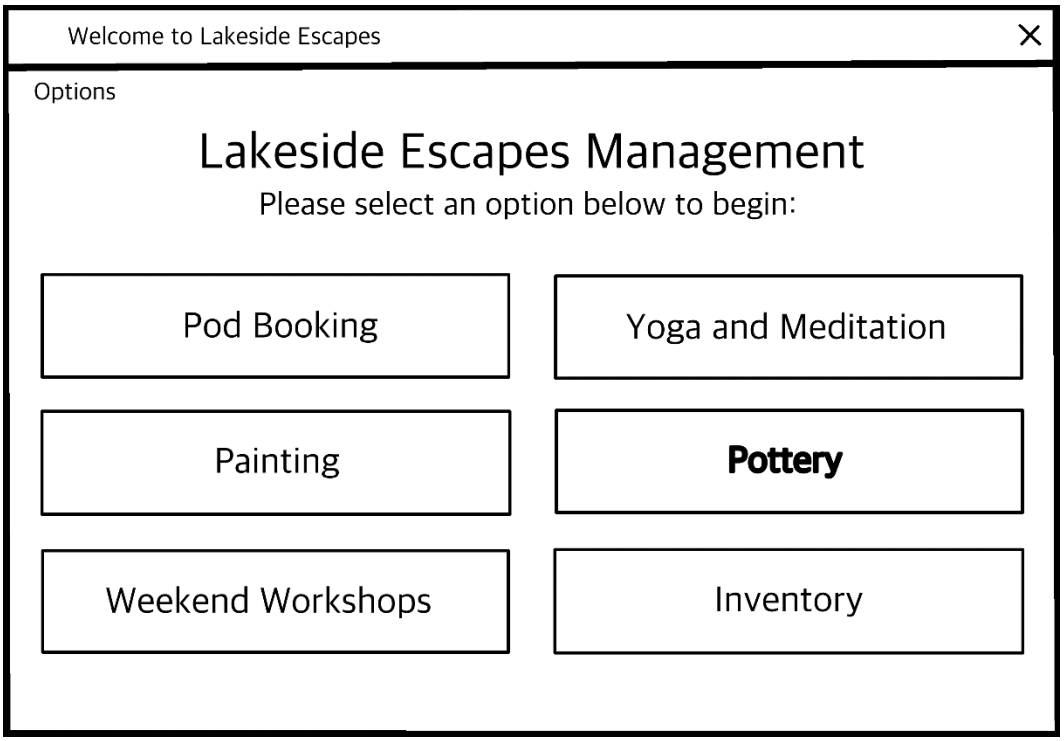
2. **Text Box (Username) –**
   This **text box** will allow the user to type in a Username using the keyboard as an **input**.

3. **Text Box (Password) –**
   This Text Box will let allow the user to enter a password using the keyboard as an **input**. By default, the **PasswordChar property** will be true, but this will be switched on and off with the **picture box**.
   This satisfies part of requirement 10.

4. **Picture Box (Password Visibility) –**
   This **picture box** will have a **click** event which will toggle the property **PasswordChar** in order to make the password visible and hidden to the user.
   This also satisfies part of requirement 10.

5. **Button (Login) -**
   This **button** will have a **click** event which will **validate** the login information and based on the result open an error popup or the main menu.
   This satisfies requirement 10.

(2) Main Menu



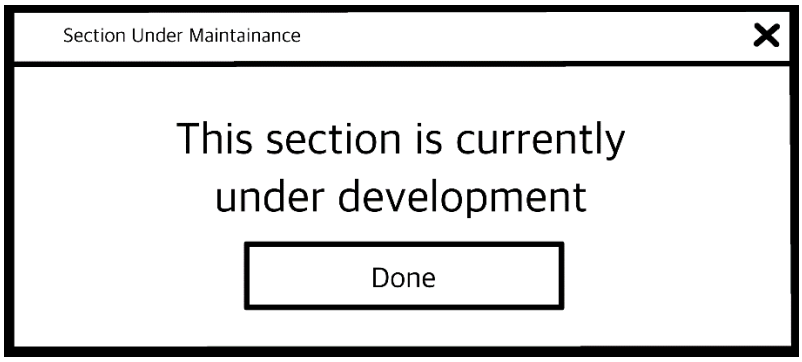
Controls and Events

1. **Menu Strip –**
   This **menu strip** will allow the User to **hover** the mouse over the word **Options**, triggering a **hover event** to open a sub menu that will consist of the options 'Logout', 'Exit' or 'Help'. Each option will have **click** events. Logout will close the menu and reopening the login form when clicked. Exit will open a **message box** asking for user confirmation. If confirmed, the whole application will close. If denied, the **message box** will close. Help will open the user guide document in the user's browser.
   This satisfies requirement 1.

2. **Buttons (Pod Booking, Yoga and Meditation, Painting, Weekend Workshops and Inventory) –**
   These **buttons** will all function in an identical way. They will make use of a **common methods class** and have a **click** event that when used will open a 'Maintenance' form, stating that the section is under development. This allows for the system to be adapted or have other systems added on to it from these **buttons.**
   This satisfies requirement 2.

3. **Button (Pottery) –**
   This **button** will function as the entry point to the main system and will have a **click** event. When clicked, it will close the current menu form and then open a second menu dedicated to the Pottery Course Management.
   This satisfies requirement 2.

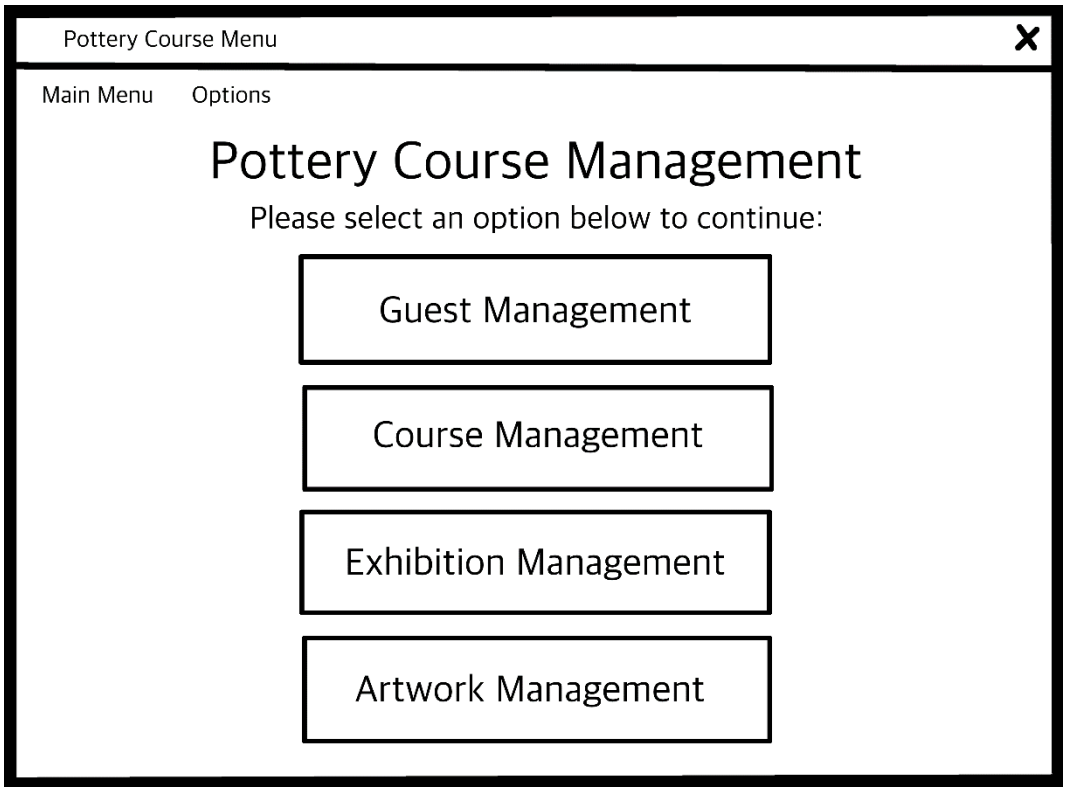
(3) Maintenance



Controls and Events

1. **Button (Done) –**
   This **button** will have a **click** event that when clicked will close the current form and reopen the main menu. This form could be removed from the system for each **button** and replaced with another system for each element of the business to complete the system.
   This satisfies requirement 2.

(4) Pottery Course Menu



This is one of the forms impacted by the change in design after I received professional feedback. Initially the Artwork Management and Exhibition Management buttons would have led to menus with Add, Edit and Delete options, but I decided to compress it when designing the storyboards into one comprehensive form for each, removing the requirement for a separate menu.

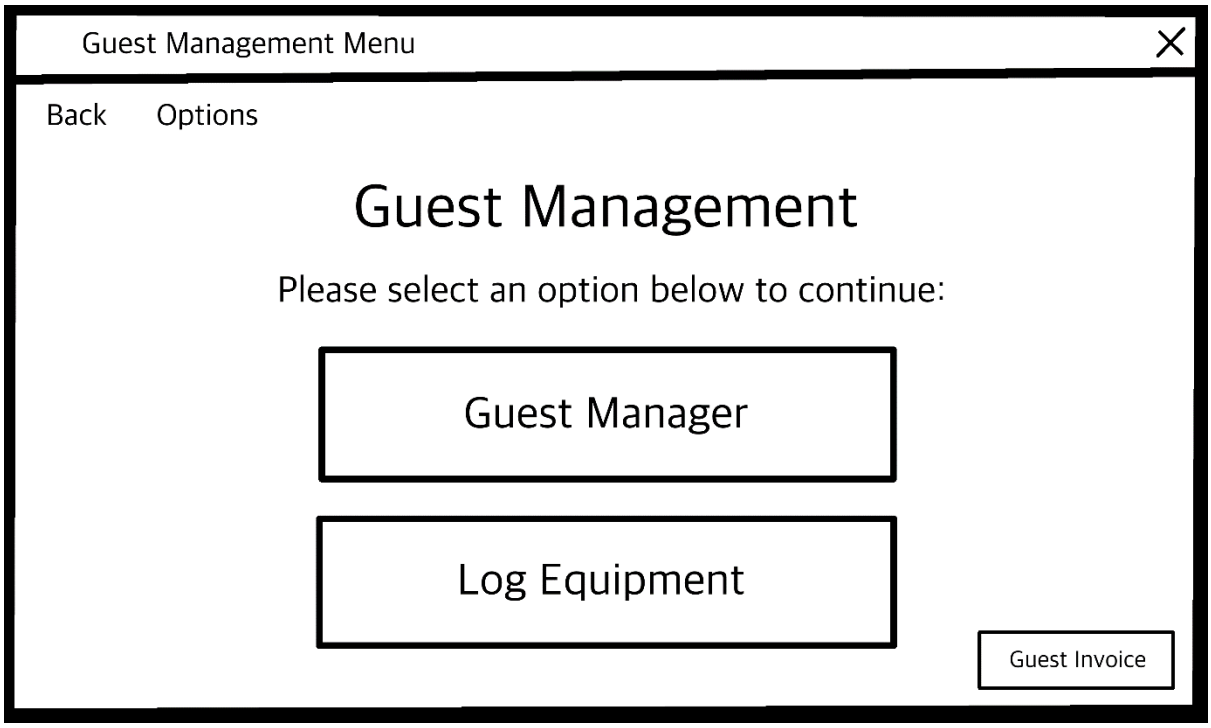
Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 2 options. 'Main Menu' has a **click** event, and when clicked will close the current menu and reopen the Main Menu. 'Options' functions identically to the options in the previous menus, with a **hover** event followed by **click** events for each submenu option, the same as before.
   This satisfies requirement 1.

2. **Buttons (Guest Management and Course Management) –**
   These **buttons** will have **click** events, and when clicked will close the current form and open the next menu. For Guest Management this will be the Guest Menu. For Course Management this will be the course menu.
   This satisfies requirement 1.

3. **Buttons (Exhibition Management and Artwork Management)-**
   These buttons will have click events, and when clicked will close the current form and open the respective management form. For Exhibition Management that will be the Exhibition Manager, for Artwork Management that will be the Artwork Manager.
   This satisfies requirement 1.

(5) Guest Management Menu



This is one of the forms impacted by the change in design after I received professional feedback. Initially this form featured individual buttons for Add, Edit and Delete guest, but was compressed in to one Guest Manager.

Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 2 options. 'Back' has a **click** event, and when clicked will close the current menu and reopen the Pottery main menu. 'Options' functions identically to the options in the previous menus, with a **hover** event followed by **click** events for each submenu option, the same as before.
   This satisfies requirement 1.

2. **Button (Guest Manager)-**
   This **button** will have a **click** event, and when clicked will open the Guest Manager form.
   This satisfies requirement 1.

3. **Button (Log Equipment)-**
   This **button** will have a **click** event and when clicked will open the Log Equipment form.
   This satisfies requirement 1.

4. **Button (Guest Invoice)-**
   This **button** will have a **click** event and when clicked will open the ID entry form for the Guest Invoice.
   This satisfies requirement 1.

(6) Guest Management

This form was impacted by the changes made based on the feedback I received. The form now contains which buttons to toggle Add, Update and Delete modes, instead of this featuring on separate forms.

Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 4 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Guest Management Menu. The 'Options' **button** functions identically to previous forms. The 'View' **button** will have a **hover** event that opens a submenu, the options within which have **click** events. Clicking on each option will perform the respective sort, for example, by skill level, cycling from beginner, to intermediate to advanced on each **click**. The second type of sort is Country, which will use an **Address class** that inherits **Icomparable** to sort by the **attribute** country. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1, 3.3 and 4.7.

2. **Combo Box (Search By)-**
   This **combo box** will allow the user to select an option to choose a search type. The Drop-Down Type will be **DropDownList** as this removes the need for validation, as the user cannot type in the box. It will have a **TextChanged** event that will trigger the search type to swap into the selected type.
   This satisfies requirement 3.2.

3. **Text Box (Search Forename and Surname) –**
   These **text boxes** will allow the user to search the records in the **data grid view**. The **text box** will have a **TextChanged** event, so that a **query** can be ran after each keystroke, filtering down the results in the Data Grid View as the user **inputs** with the keyboard.
   This satisfies requirement 3.2.

4. **Text Box (Search Guest ID)**
   This **text box** will have the same function and **events** as the one above, however it will also have an extra section of code in the **TextChanged** event, which will check what the input was, and remove anything that is not a number. It will also replace the 'G' at the start of the ID after every change, ensuring the **format** remains 'Gxxxx'.
   This satisfies requirement 3.2 and 11.

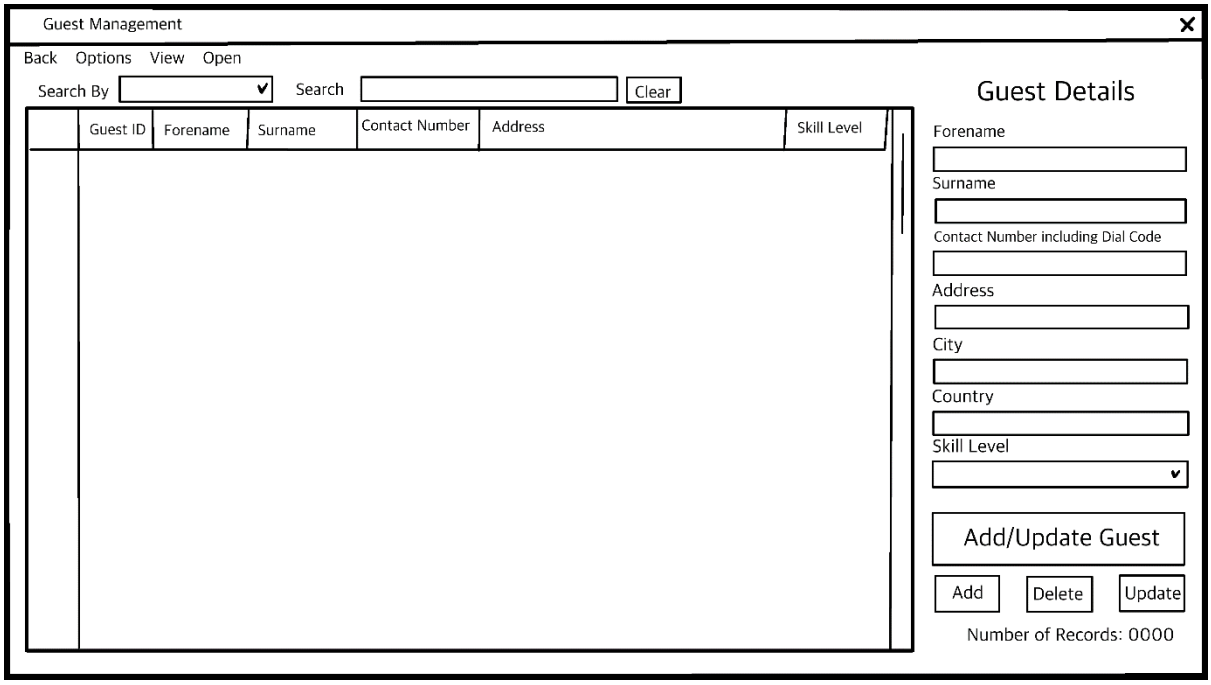
5. **Button (Clear)-**
   This **button** will have a **click** event that will empty all the search boxes, and perform a **query** on the **Data Grid View** data source to reset the records to all.
   This satisfies requirement 3.2.

6. **Buttons (Add, Delete, Update)-**
   These **buttons** all will have **click** events to toggle in and out of each mode. Clicking when a button is enabled will disable the button and switch the form into the selected mode.
   This satisfies requirement 3.4, 3.5 and 3.6.

7. **Data Grid View –**
   This **Data Grid View** will display the Guest details from the **Guest table** in a neat manner. Its **Data Source** will change as a result of searches and sorts. It will have a **SelectionChanged** event that reacts differently depending on the active mode. In Add mode the selection will be cleared to give the illusion of selections being disabled. In Delete mode the whole row will select, and autofill details. The same occurs in Update mode. **MultiRowSelect** will be False.
   This satisfies requirement 3.1 and 3.9.

8. **Label (number of results)-**
   This **label** has no events but will have its text updated every time the **Data Grid View** is refreshed, displaying the count of the visible records.
   This satisfies requirement 3.1.

a. Add/Update Mode



Controls and Events

1. **Text Boxes (Forename and Surname)-**
   These **text boxes** will allow the user to **input** a guest's Forename or Surname in Add Mode or change a guest's Forename or Surname in update mode. They do not have any events.
   This satisfies requirement 3.4 and 3.6.

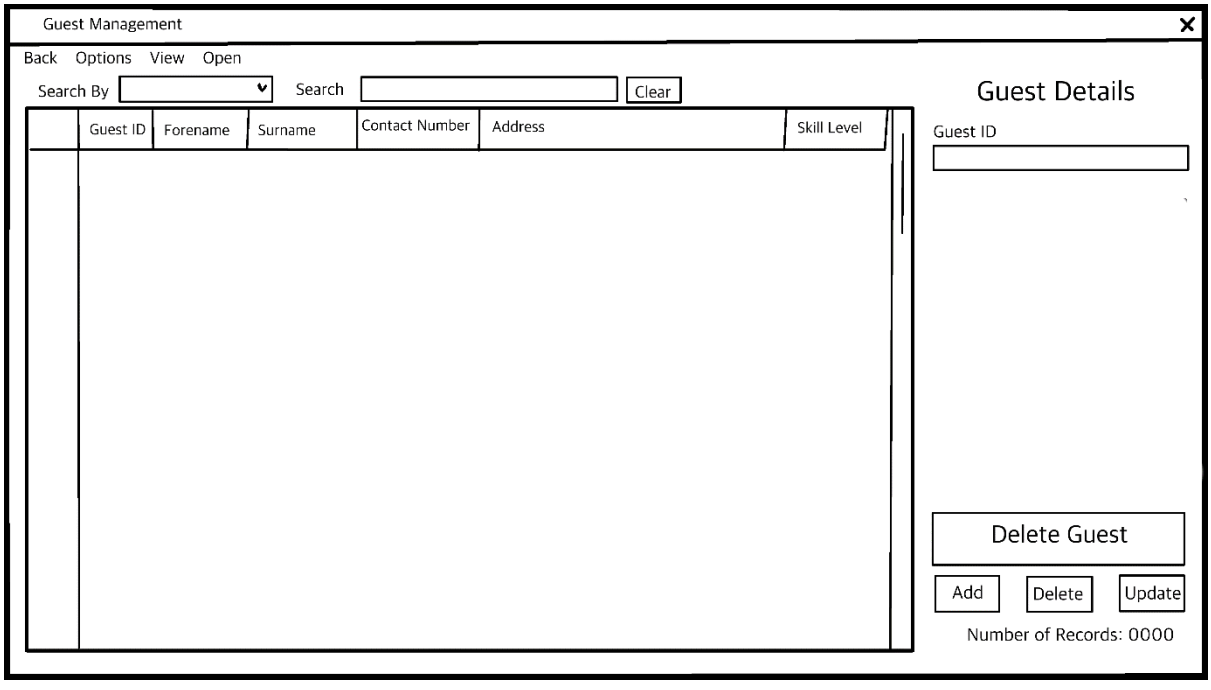
2. **Text Box (Contact Number)-**
   This **Text Box** will allow the user to **input** or update a guest's phone number. It will have a **TextChanged** event which will handle **input** and ensure the '+' cannot be removed and that only numbers can be entered.
   This satisfies requirement 3.4, 3.6 and 11.

3. **Text Boxes (Address, City and Country)-**
   These **text boxes** will function the same as the forename and surname, except will, when added to the system, concatenate into one single string, separated by commas. They will have a **text changed** event to avoid a comma being entered, as I use that char as a separator.
   This satisfies requirement 3.4 and 3.6.

4. **Combo Box (Skill Level)-**
   This **combo box** will allow the user to set a guest's Skill Type. It will be of drop-down style **DropDownList** which will **minimise errors** by restricting the **input** to a set choice. There will be an **event** when a record is selected in update mode which will check, using a **query**, that the user has no future bookings. If they do, changing the skill type will be disabled until a time which they have no future booking.
   This satisfies requirement 3.4, 3.6 and 3.7.

5. **Button (Add/Update Guest)-**
   This **button** will have a **click** event which will trigger the **methods** and validation to either provide feedback or add/update the database with a new or updated record.
   This satisfies requirement 3.4 and 3.6.

b. Delete Mode



Controls and Events

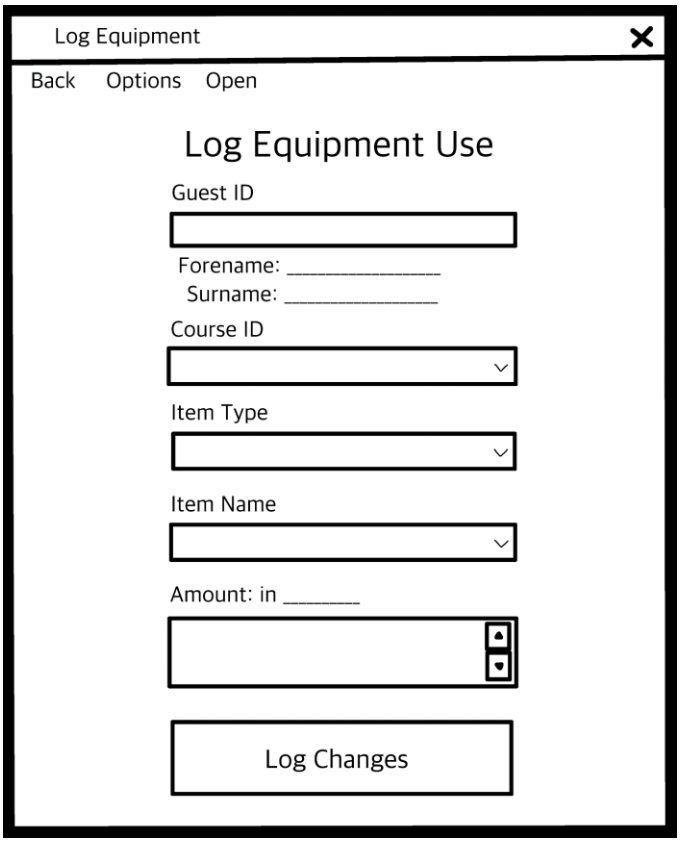
1. **Text Box (Guest ID)-**
   This **text box** will allow the user to delete a guest by selecting a record in the **data grid view** which will then populate the text box. It will have a **KeyPress** event that handles any typing by just ignoring it, and it will also have a **TextChanged** event, which will check what the **input** was, and remove anything that is not a number. It will also replace the 'G' at the start of the ID after every change, ensuring the **format** remains 'Gxxxx'.
   This satisfies requirement 3.5 and 11.

2. **Button (Delete Guest)-**

   This **button** will have a **click** event which will trigger the **methods** and validation to either provide feedback or delete a record from the database. Deleting a guest with bookings will cascade delete the bookings.

   This satisfies requirements 3.5 and 3.8.

(7) Log Equipment



Controls and Events

1. **Menu Strip -**
   This **menu strip** will contain 3 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Guest Management Menu. The 'Options' **button** functions identically to previous forms. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1 and 4.7.

2. **Text Box (Guest ID)-**
   This **text box** will have a **TextChanged** event, which will check what the input was, and remove anything that is not a number. It will also replace the 'G' at the start of the ID after every change, ensuring the **format** remains 'Gxxxx'. It will also validate the ID after each text changed, and if it finds a valid ID, will return the Forename and Surname to populate the labels. The other options on the form will subsequently become available.
   This satisfies requirement 11 and 7.

3. **Labels (forename and surname)-**
   These **labels** will populate and change visibility based on the result of the guest ID validation. They have no events.
   This satisfies requirement 7.

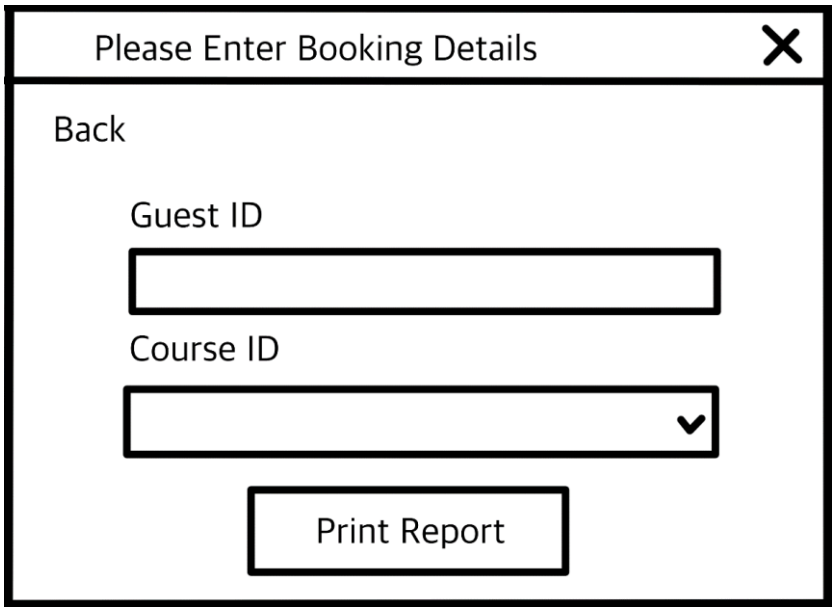
4. **Combo Boxes (Course ID, Item Type, and Item Name)-**
   These **combo boxes** will populate and adjust visibility based on the ones prior. All will have drop down style **DropDownList** to limit **input** to the existing options and disable typing as an **input**. Course ID will populate based on the user's bookings when a valid ID is entered. A selected course will make item type and name visible, and selecting a type will then populate item name with items of that type. These **combo boxes** will have **TextChanged** events that trigger these relative changes.
   This satisfies requirement 7.

5. **Label (Amount)-**
   This **label** will have no events but will change based on the item type selected as each type is measured in different amounts.
   This satisfies requirement 7.

6. **Numeric Up Down (amount)-**
   This **numeric up down** will allow users to **input** the amount if an item used, but will have a **KeyPress** event to disable typing, meaning it can only be adjusted using the arrow buttons in the **control** itself. It will have a **minimum** property of 0 and a **maximum** property of 9999 and will **increment** by 0.25.
   This satisfies requirement 7.

7. **Button (Log)-**
   This **button** will only be enabled when information has been correctly selected, and will have a **click** event, which when clicked will **validate** the information and either prompt an error message or update the database.
   This satisfies requirement 7.

(8) Guest Invoice ID Entry



Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 1 option. 'Back' has a **click** event, and when clicked will close the current screen and reopen the guest management menu.
   This satisfies requirement 1.

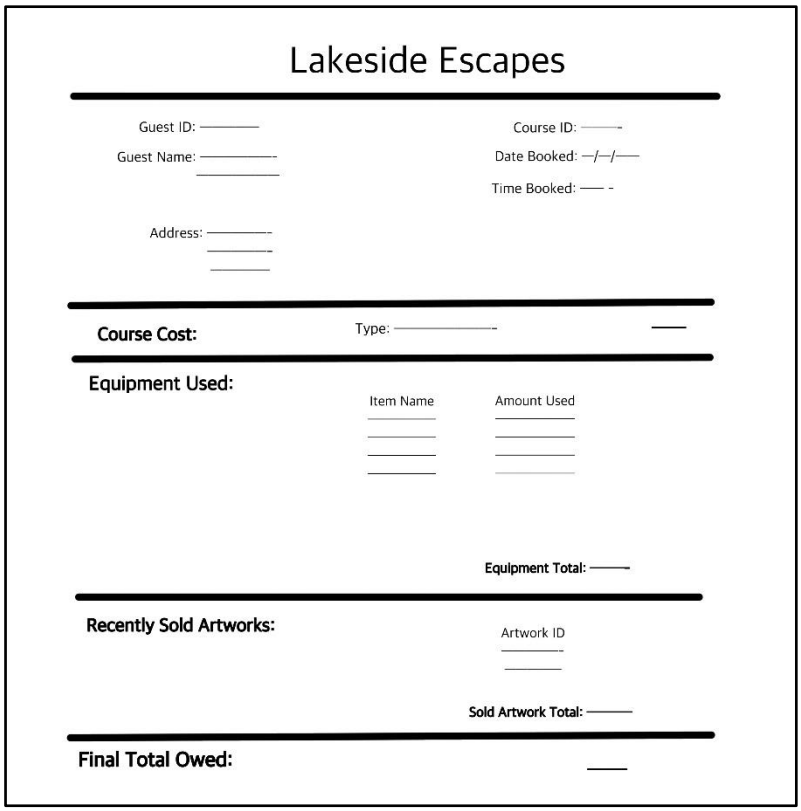
2. **Text Box (Guest ID)-**
   This will function identically to the **text box** for guest ID **on Log Equipment**, but instead of retuning forename and surname, it will populate the **combo box** with available courses, or **outputs** an error message in relation to the guest located in the database.
   This satisfies requirement 8.

3. **Combo Box (Course ID)-**
   This **combo box** will be populated with available courses when a guest ID is correctly entered above. It will have a drop-down style of **DropDownList** so the user **input** will be restricted to preset data, mitigating error. It will have a **TextChanged** event which will enable or disable the print button.
   This satisfies requirement 8.

4. **Button (Print Report) –**
   This **button** will be enabled when the user has entered the rest of the data. It will have a **click** event which will then close the current form and open the Guest Invoice viewer.
   This satisfies requirement 8.

5. **Label (No Bookings) –**
   This **label** will only be visible when the user has no bookings, over the course **combo box** to make it inaccessible.
   This satisfies requirement 8.

(9) Invoice



Controls and Events

1. **Button (Done) –**
   This **button** will have a **click** event to close the current form and open the guest menu.
   This satisfies requirement 8.

Report Design and Parameters

This report will have a variety of parameters passed into it because of queries which will return results based on the guest ID and Course ID previously entered.

1. GuestIDInvoice
2. GuestForename
3. GuestSurname
4. Addressline1
5. AddressLine2
6. AddressLine3
7. CourseID
8. DateBooked
9. TimeBooked

   These first 7 will be used to fill the invoice information at the top of the invoice, such as guest name and address.

10. CourseCost
11. CourseType

    These two will be used to add the course base information to the second section of the invoice.

12. usedEquipmentTrue

    This Boolean is used to toggle the visibility of the equipment Tablix, which lists the equipment used, amount and cost.

13. TotalEquipPlusCost

    This is the equipment cost which will show at the bottom of the equipment section.

14. TotalSoldArtwork

    This is the total deduction amount which will show at the bottom of the artwork section.

15. FinalInvoiceTotal

    This is the final total which will show at the bottom of the invoice.

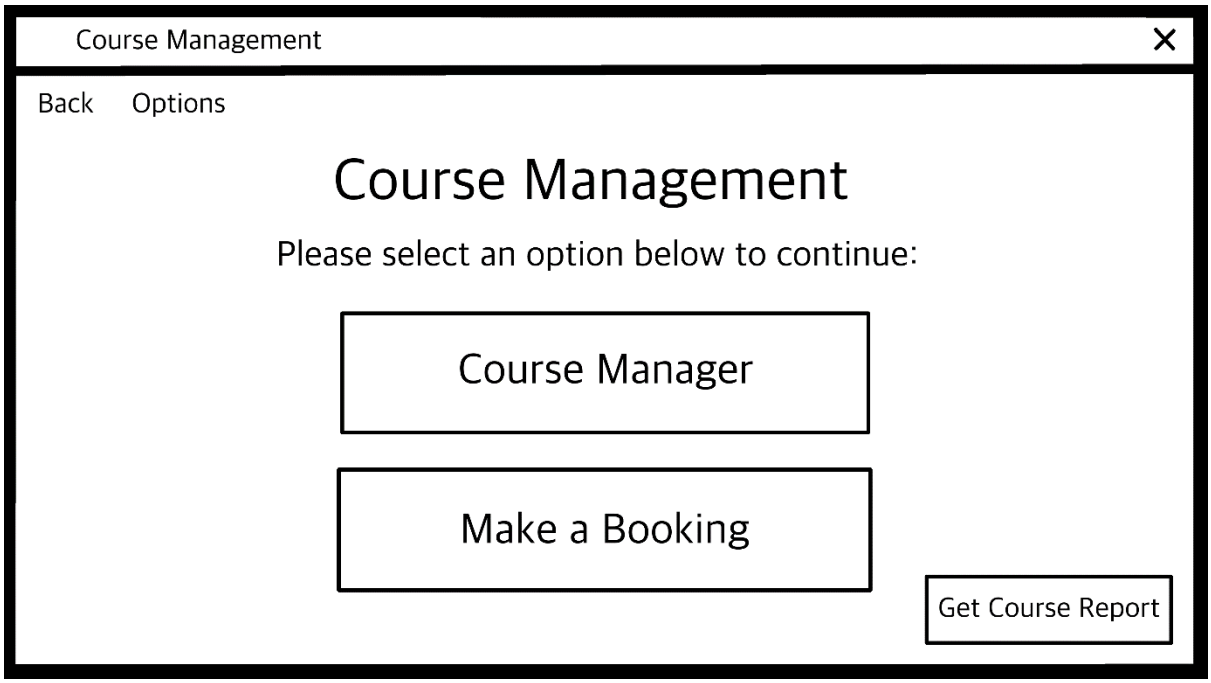
16. RecentSoldTrue

    This Boolean value toggles the visibility of the artwork sold Tablix.

17. (Dataset) InvoiceEquipment
18. (Dataset)SoldArtworks

    These Datasets are used to populate the Tablix in the report.

(10) Course Management Menu



Controls and Events

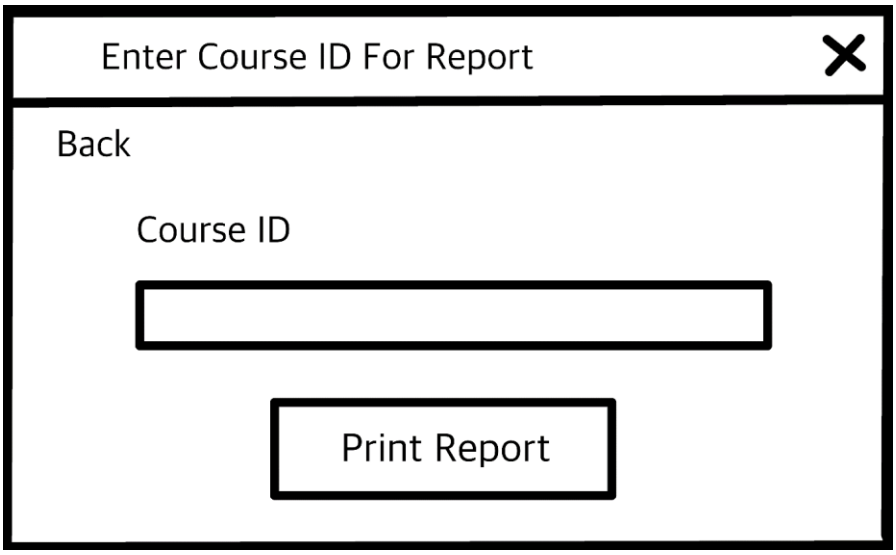
1. **Menu Strip –**
   This **menu strip** will function identically to the guest management menu strip.
   This satisfies requirement 1.

2. **Button (Course Manager)-**
   This **button** will have a **click** event, and when clicked will close this menu and open the course management form.
   This satisfies requirement 1.

3. **Button (Make a Booking)-**
   This **button** will have a **click** event and when clicked will close the current form and open the Booking form unless it is between 20th December and 20th January. Between these dates an error message will be **output**, and the menu will not close.
   This satisfies requirement 1 and 4.7.

4. **Button (Course Report)-**
   This **button** will have a **click** event and when clicked will close the current form and open the course ID entry form for the report.
   This satisfies requirement 1.

(11) Course Report ID Entry



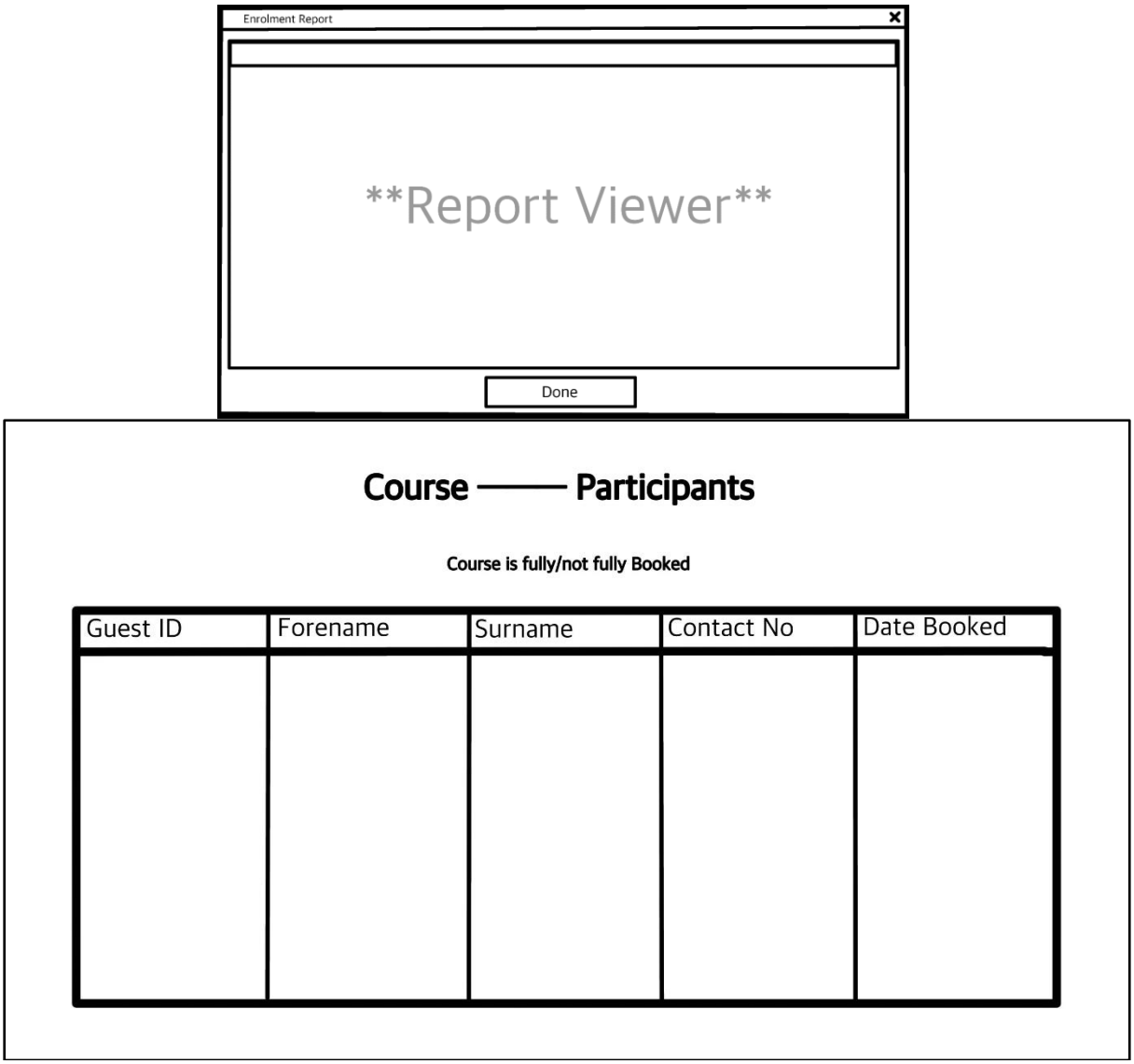
Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 1 option. 'Back' will have a **click** event, and when clicked will close the current screen and reopen the course management menu.
   This satisfies requirement 1.

2. **Text Box (Course ID)-**
   This will function identically to the **text box** for guest ID **on Log Equipment**, but instead of **inputting** a guest ID it will be a course ID, so the **TextChanged** event will adjust to 'Cxxx'.
   This satisfies requirement 4.2 and 11.

3. **Button (Print Report)-**
   This **button** will have a **click** event and when clicked will close the current **input** form and open the report form.
   This satisfies requirement 4.2.

(12) Enrolment Report



Controls and Events

1. Button (Done) –

This **button** will have a **click** event to close the current form and open the course menu.

This satisfies requirement 8.

Report Design and Parameters

This report will have a variety of parameters passed into it because of queries which will return results based on the Course ID previously entered.

1. PassedID
   This is the ID that will appear at the top of the form.
2. FullyBooked?
   This is the Boolean that will determine the text stating booked or not booked.
3. (dataset) Enrolment
   This is the dataset that will populate the participants Tablix.

(13) Course Management

Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 3 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Course Management Menu. The 'Options' **button** functions identically to previous forms. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1 and 4.7.

2. **Radio Button (enable search)-**
   This **radio button** will manage the form's search function. It will have a **checkChanged** event that when triggered will enable or disable the search based on the current state. It also will have a **click** event, which is used to trigger the **checkChanged** event.
   This satisfies requirement 4.3.

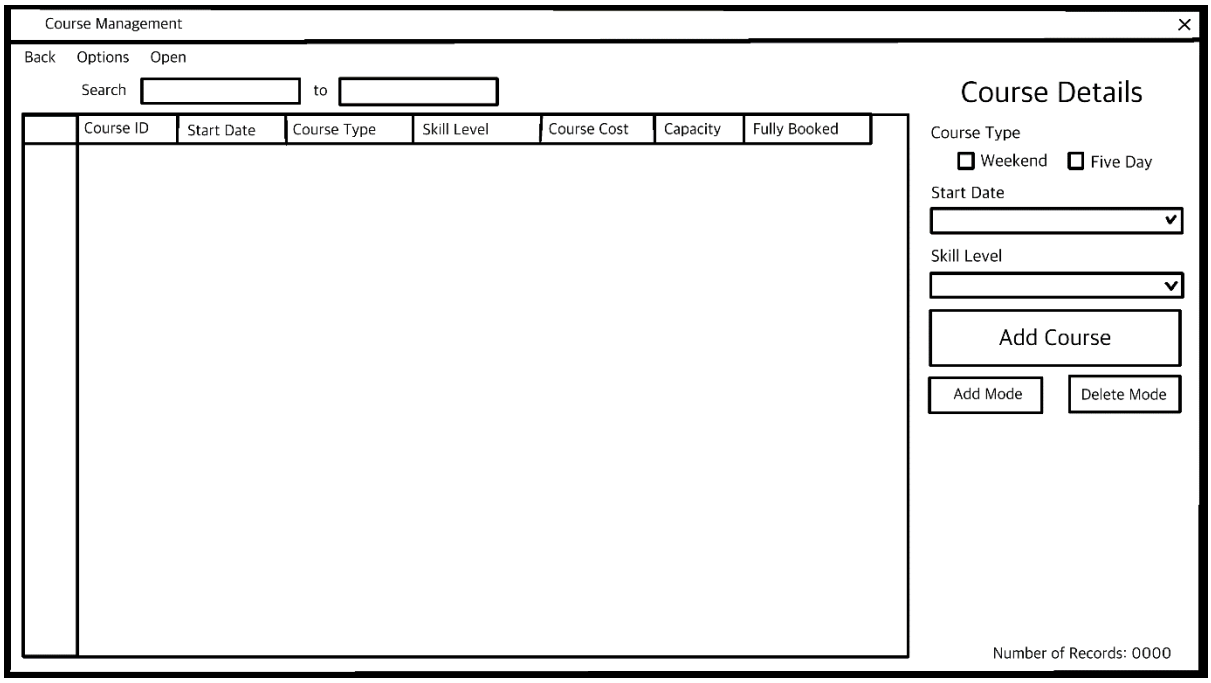
3. **Date Time Picker (Start Search and End search)-**
   These **date time pickers** will be used to select the start and end date for the search. They will have **ValueChanged** events that ensure the Dates selected are valid and then will perform the search using a **query**.
   This satisfies requirement 4.3.

4. **Data Grid View –**
   This **Data Grid view** will display the existing courses. Its **Data Source** will change because of searches and sorts. It will have a **SelectionChanged** event that reacts differently depending on the active mode. In Add mode the selection will be cleared to give the illusion of selections being disabled. In Delete mode the whole row will select, and autofill details. **MultiRowSelect** will be False.
   This satisfies requirement 4.1.

5. **Label (Record count)-**
   This **label** will work identically to the record counter on the guest manager form.
   This satisfies requirement 4.1.

6. **Buttons (Add mode and Delete mode)-**
   These **buttons** will switch modes in the same manner that occurs in the guest manager, just minus the update mode.
   This satisfies requirement 4.4 and 4.5.

a. Add Mode



Controls and Events

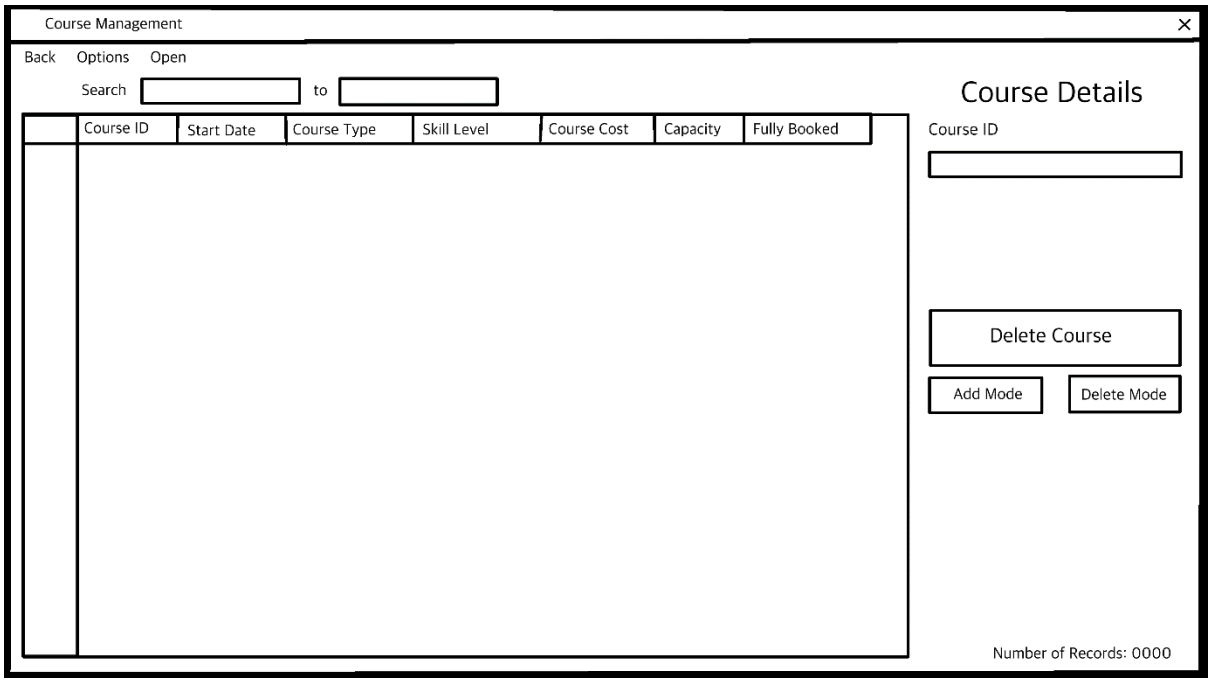
1. **Check Box (Weekend and Five-Day)-**
   These **check boxes** will allow the user to choose a course type to add. They both will have **click** events which will check if the other is checked, and if it is, uncheck it and then check itself. This will ensure only one can be selected at a time.
   This satisfies requirement 4.4.

2. **Date Time Picker (Start Date) –**
   This **date time picker** will allow the user to select the start date for the new course. It will have a **ValueChanged** event to ensure it is a future date and has not already been used. The actual day is validated when the **button** is clicked.
   This satisfies requirement 4.4.

3. **Combo Box (Skill Level) –**
   This **combo box** will allow the user to select a skill level for the course and is populated with options based off the course type ticked. It will be of Drop-Down Type **DropDownList**, to mitigate error and disable typing. It has no other **events**.
   This satisfies requirement 4.4.

4. **Button (Add Course) –**
   This **button** will have a **click** event that will then **validate** the information selected, ensuring there is space in the year and the start day is correct etc. If it is **valid** the user will be notified that the add was successful.
   This satisfies requirement 4.4.

b. Delete Mode

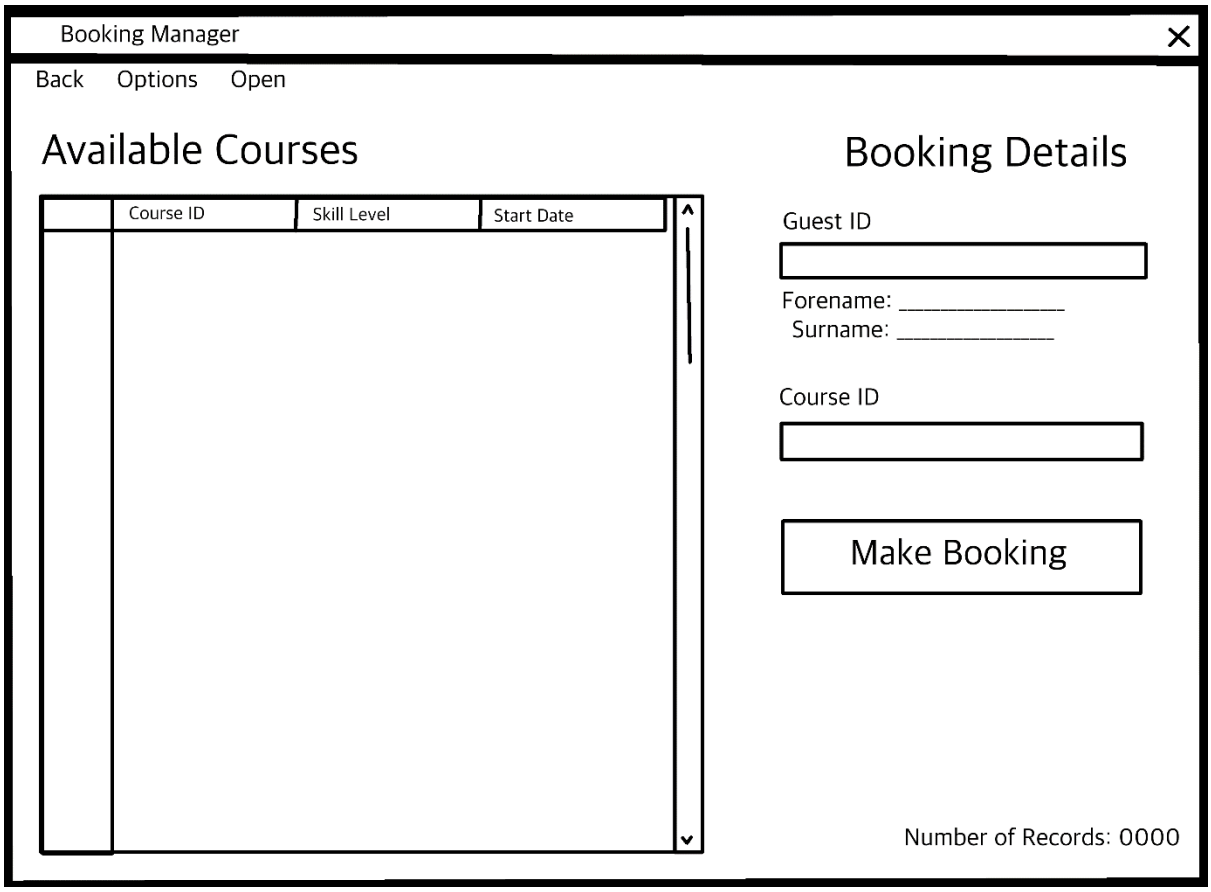


Controls and Events

1. **Text Box (Course ID)-**
   This **text box** will function identically to the Guest ID text box for deleting a guest, but with the course ID format 'Cxxx' instead of guest. You can select a record from the Data Grid view to populate it.
   This satisfies requirement 4.5.

2. **Button (delete Course)-**
   This **button** will have a **click** event and when clicked will **validate** the ID entered an ensure it has no guests. If it does not, the course will be deleted.
   This satisfies requirement 4.5.

(14) Make A Booking



Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 3 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Course Management Menu. The 'Options' **button** functions identically to previous forms. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1.

2. **Data Grid View –**
   This **Data Grid View** will show the information for each course available. If a guest ID is correctly entered, the results in the data grid view will filter to suit the guest. It will have a **SelectionChanged** event to clear selections, to give the appearance that you cannot interact with the data grid view.
   This satisfies requirement 4.6 and 4.8.

3. **Label (record Number)-**
   This **label** will function the same as the other record count labels in the system.
   This satisfies requirement 4.6.

4. **Text Box (Guest ID)-**
   This **text box** will work in the same manner as the Guest ID **text box** in the equipment logger. It will have the same **events** and return the forename and surname in the same manner.
   This satisfies requirement 4.6.

5. **Labels (forename and Surname)-**
   These **labels** will work in the same manner as the labels on the equipment logger form, changing as a guest ID is entered.
   This satisfies requirement 4.6.

6. **Combo Box (Course ID)-**
   This **combo box** will function identically to the **combo box** on the equipment logging form, but instead of populating with the bookings, it will populate with available courses that have not been booked. Everything else is identical.
   This satisfies requirement 4.6 and 4.8.

7. ***Button (Make Booking)-***
   This button will be enabled once all info has been entered and will have a click event that attempts to place the booking. The form will refresh after clicking to avoid accidental double booking. Double booking and overbooking will be handled so it is impossible.
   This satisfies requirement 4.6.

(15) Exhibition Management

Controls and Events

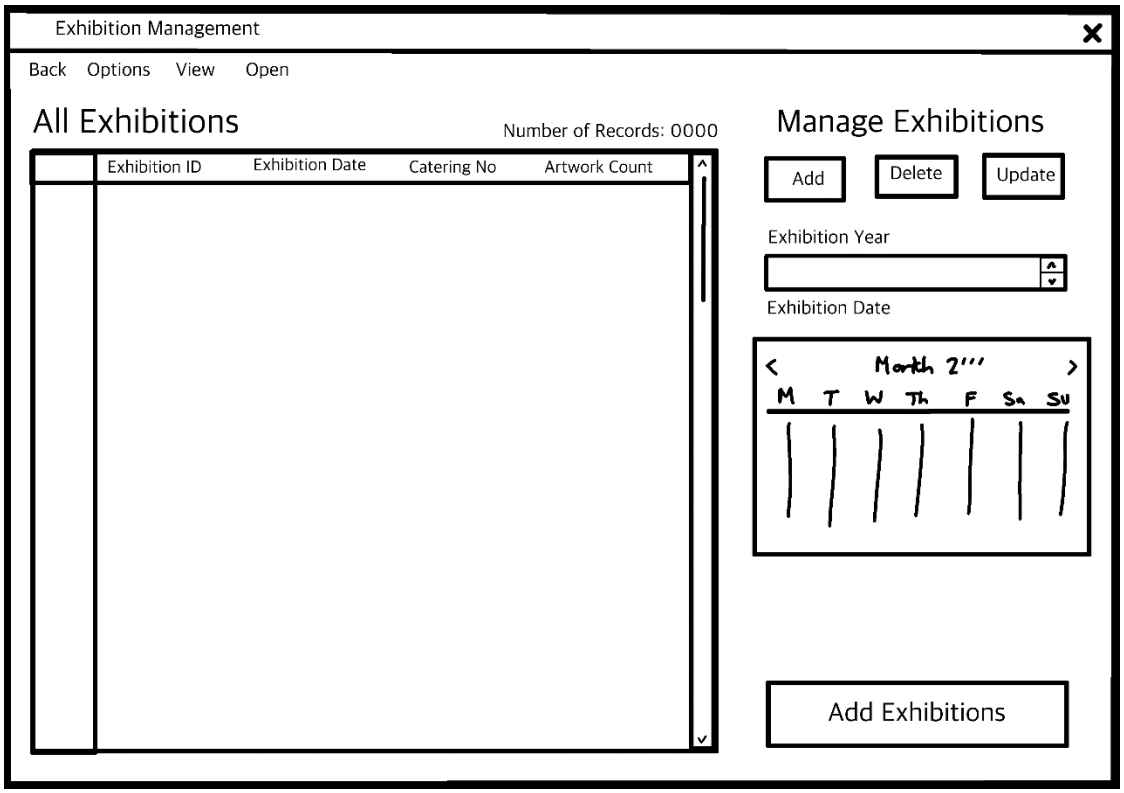
1. **Menu Strip –**
   This **menu strip** will contain 4 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Pottery Menu. The 'Options' **button** functions identically to previous forms. The 'View' **button** will have a **hover** event that opens a submenu, the options within which have **click** events. Clicking on each option will make the selected **data grid view** available. These may be disabled based on the mode, as past exhibitions cannot be updated. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1, 6.1, 6.4 and 4.7.

2. **Data Grid Views (Past, Future and All) –**
   These **Data Grid views** will allow the user to view each category of records of Exhibitions. It will function in the same manner as the other **data grid views** in guest and course management, with the selection disabled using **events** in add mode, but allowed in delete and update to autofill details.
   This satisfies requirement 6.1.

3. **Label (Record Number)-**
   This **label** will function identically to all other record counters.
   This satisfies requirement 6.1.

4. **Buttons (Add, Update and Delete mode)-**
   These **buttons** will function identically to the other mode-switching buttons in the guest and course managers but will also disable **data grid views** on click events, limiting some modes to some views.
   This satisfies requirements 6.1, 6.2, 6.3, 6.4 and 6.5.

a. Add Mode



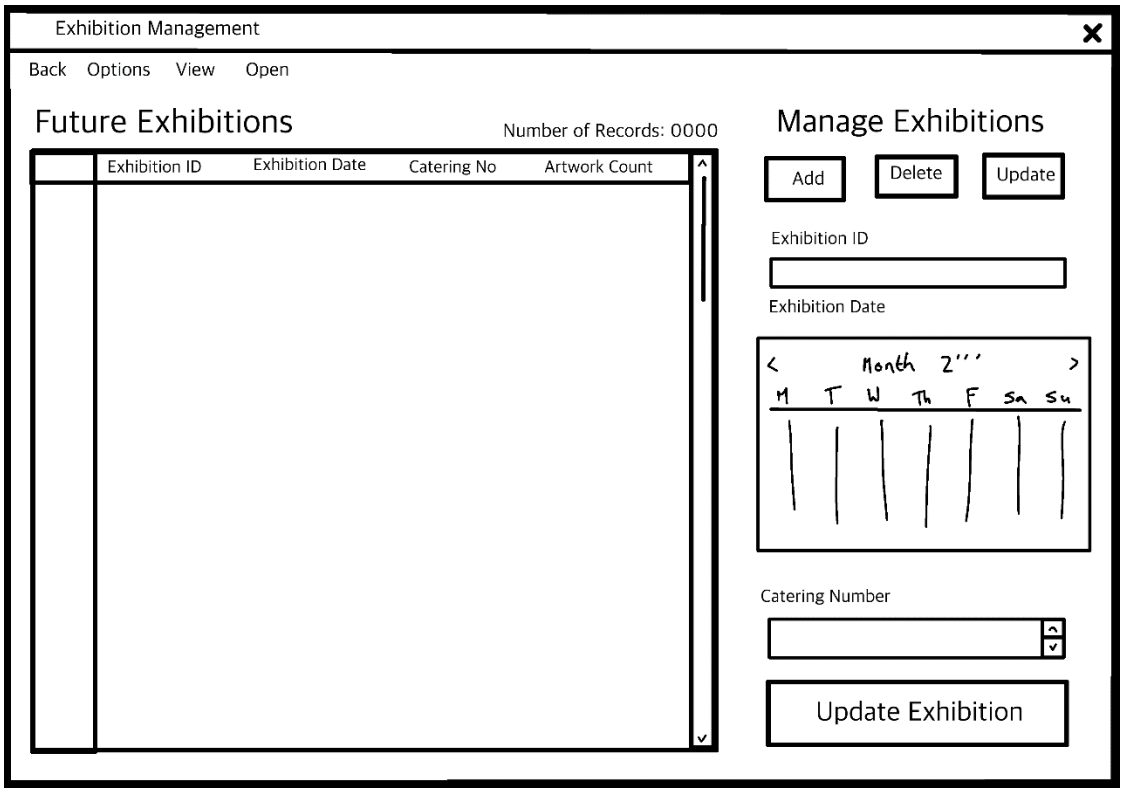
Controls and Events

1. **Numeric Up Down (Exhibition Year)-**
   This **numeric up down** will allow the user to select a year. It will have a **ValueChanged** event to check if the year is already in use, which will highlight it in red. It will also have a **minimum** of the current year. It will have a **KeyPress** event to disable typing, so value can only be changed with arrow buttons.
   This satisfies requirement 6.2.

2. **Month Picker (Exhibition Date)-**
   This **month picker** will be used to select a date in December in the selected year. The arrows to change month will be disabled, and a **MouseDown** event to catch the location of a click to handle and disable the ability to click on the month label to change to year view.
   This satisfies requirement 6.2.

3. **Button (Add Exhibition)-**
   This **button** will have a **click** event and when clicked will attempt to create a new record for this exhibition.
   This satisfies requirement 6.2.

b. Update Mode



Controls and Events

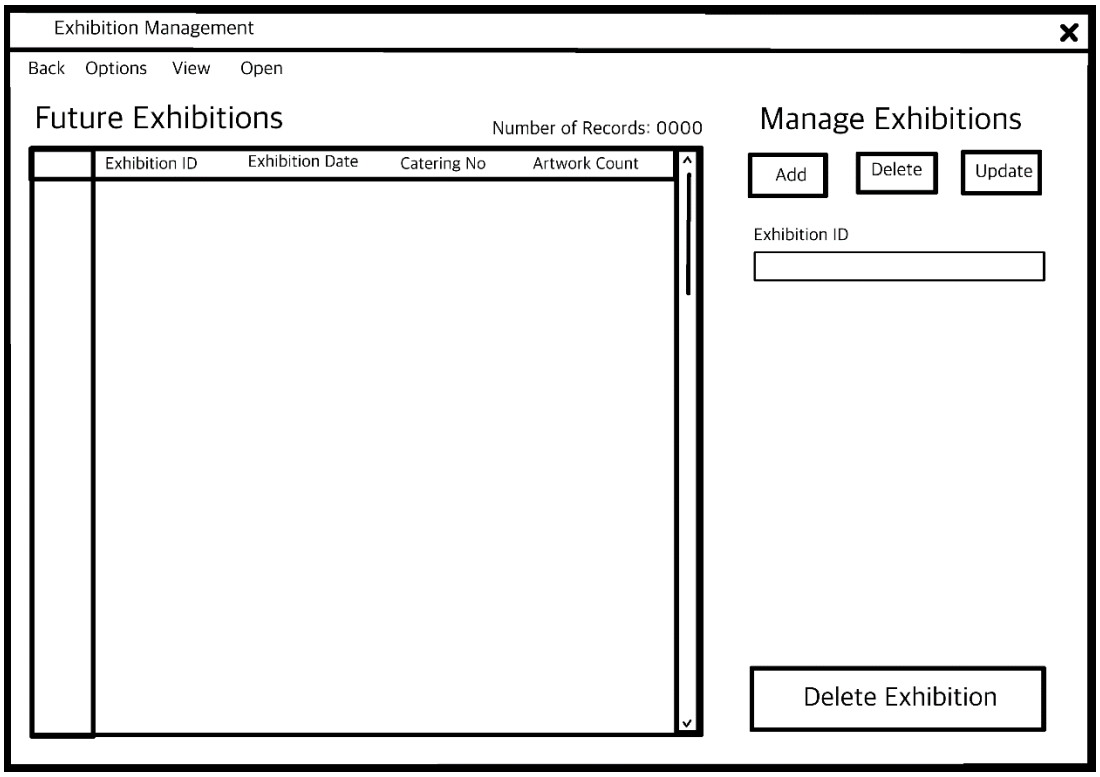
1. **Text Box (Exhibition ID)-**
   This **text box** will work in the same manner as the course ID delete **text box**, where you are unable to type and select a record from the **data grid view** instead. The only difference is the **event** that keeps the format correct formats to 'EXxxx'.
   This satisfies requirement 6.4.

2. **Month Picker (Exhibition Date)-**
   This will work identically to the previous **month picker**, except for changing the date rather than setting it. The year cannot be altered.
   This satisfies requirement 6.4.

3. **Numeric Up Down (catering Number)-**
   This will work the user to alter the catering number and functions in the same way as the year **numeric up down**, where typing is disabled, and it can only be altered with the arrows as **input**.
   This satisfies requirement 6.4.

4. **Button (Update Exhibition)-**
   This **button** will work the same as the add **button**, but for updating an existing record.
   This satisfies requirement 6.4.

c. Delete Mode



Controls and Events

1. **Text Box (Exhibition ID)-**
   This **text box** will function identically to the previous.
   This satisfies requirement 6.3.

2. **Button (Delete Exhibition)-**
   This will function identically to the previous add and update **buttons**, but also performs a check that it is a future exhibition with no artworks.
   This satisfies requirements 6.3.

(16) Artwork Management

Controls and Events

1. **Menu Strip –**
   This **menu strip** will contain 4 different **buttons**. The 'Back' **button** functions the same as the ones in the previous forms, closing the current form and opening the Pottery Menu. The 'Options' **button** functions identically to previous forms. The 'View' **button** will have a **hover** event that opens a submenu, the options within which have **click** events. Clicking on each option will make the selected **data grid view** available. These may be disabled based on the mode, as non-Advanced artworks cannot be updated. The 'Open' option will have a **hover** event which opens a submenu that has an option for every other management form within the system, minus the current one. **Clicking** on each will close the current form and open the respective one.
   This satisfies requirement 1 and 4.7.

2. **Combo Box (search by)-**
   This **combo box** will function identically to the previous search combo box, but with the options 'Artwork ID' and 'Name'.
   This satisfies requirement 5.2.

3. **Text Box (Search Bar)-**
   This **text box** will function identically to the guest ID search box, with the same **events**, performing a search after each keystroke, and when in Artwork ID mode formatting the text to 'Axxxx' and limiting **input** to numbers.
   This satisfies requirement 5.2.

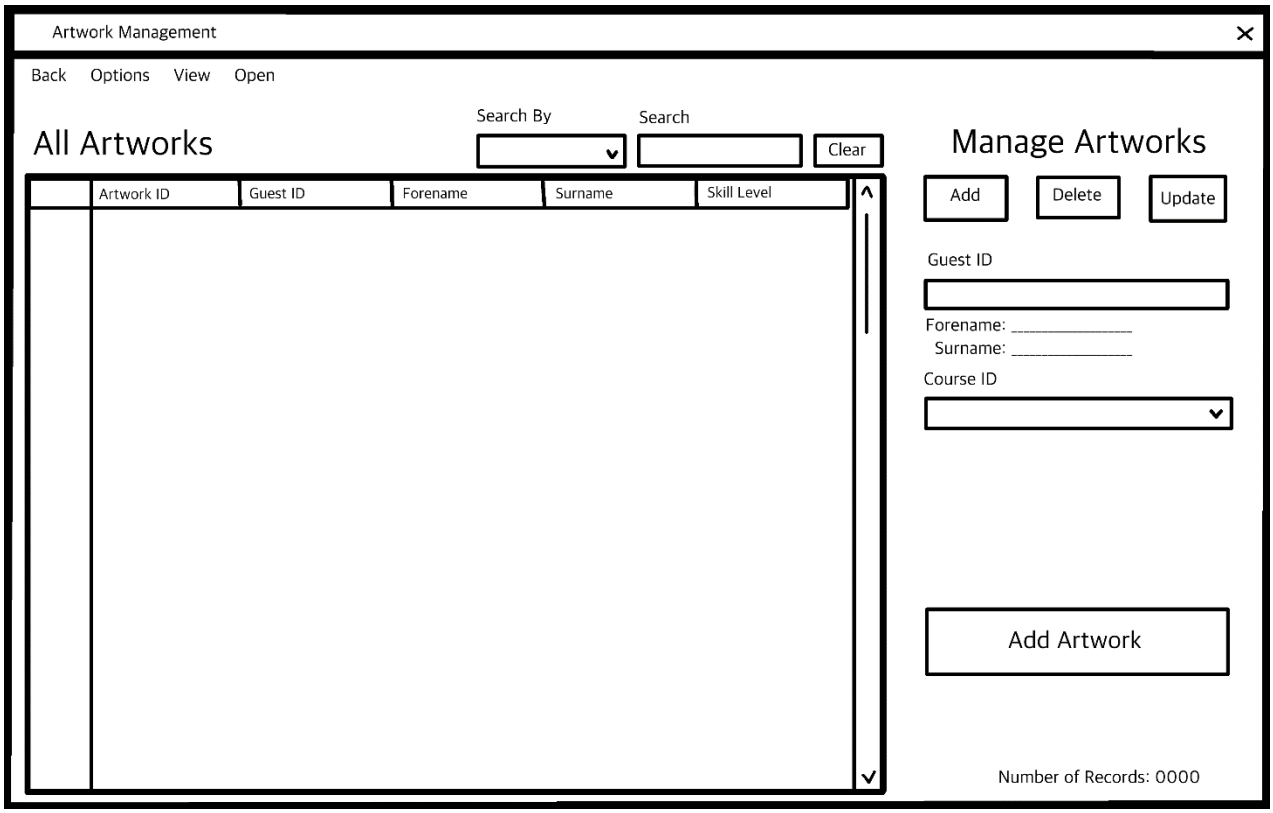
4. **Button (Clear Search)-**
   This **button** functions identically to the clear **button** on the guest manager form.
   This satisfises requirement 5.2.

5. **Data Grid Views (All and Advanced)-**
   These two **data grid views** will allow the user to view basic information about all artworks on one, and extra information such as cost and sold? On advanced artworks in a different view. Both will have **SelectionChanged** events to give the illusion of no selection in add mode and allow selection in update and delete mode. It will also have **MultiRowSelect** disabled.
   This satisfies requirement 5.1.

6. **Label (Record Count)-**
   This **label** will function identically to the other record count labels.
   This satisfies requirement 5.1.

7. **Buttons (Add, Update and Delete Modes)-**
   These will function identically to the other mode swap **buttons** on other management forms.
   This satisfies requirement 5.3, 5.5, 5.6, 5.7, 5.8.

a. Add Mode



Controls and Events

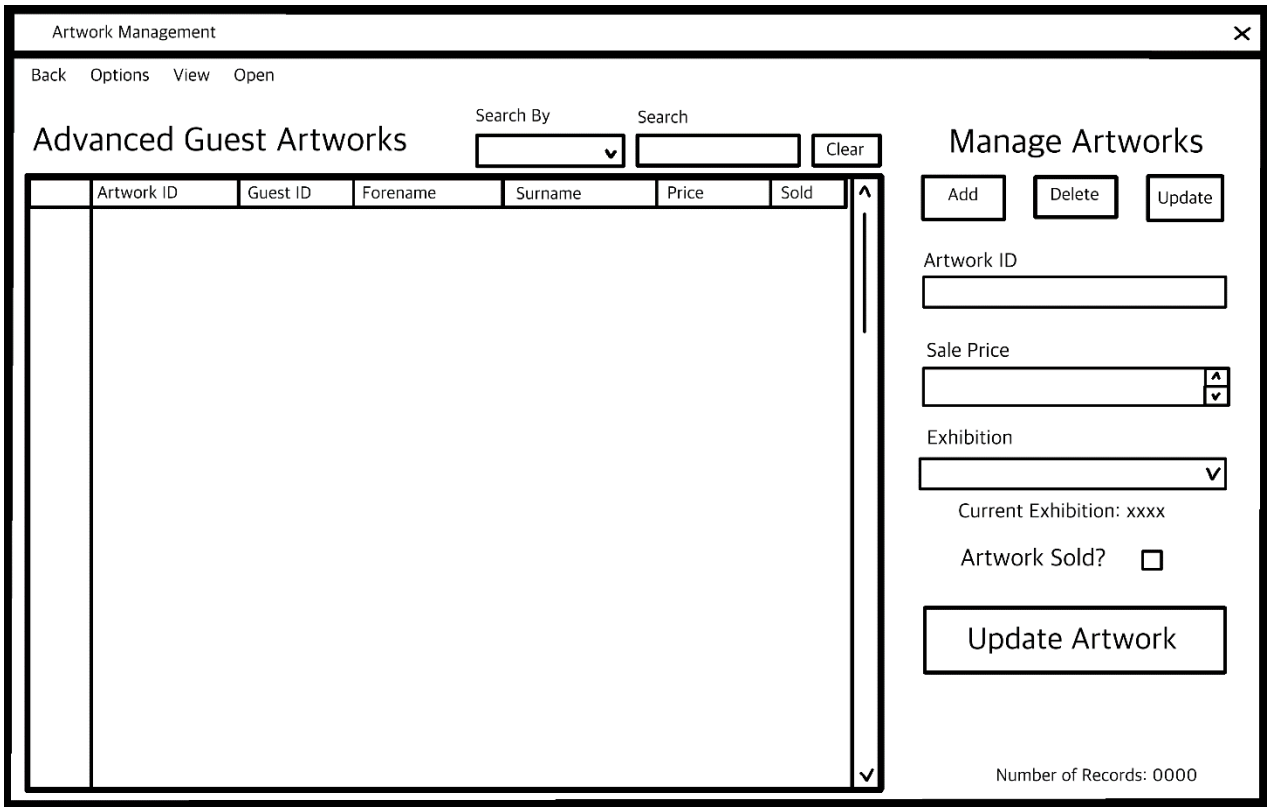
1. **Text Box (Guest ID)-**
   This will function identically to the guest ID **Text Box** on the Equipment logging form, returning the forename, surname and populating the course **combo box.**
   This satisfies requirement 5.3.

2. **Labels (Forename and Surname)-**
   These will function identically to the **labels** on the equipment logging form.
   This satisfies requirement 5.3.

3. **Combo Box (Course ID)-**
   This will function identically to the **combo box** in the equipment logging form, populating with the courses the guest is booked into.
   This satisfies requirement 5.3.

4. **Button (Add Artwork)-**
   This will function identically to the other Add **buttons** such as exhibition or guest.
   This satisfies requirement 5.3.

b. Update Mode



Controls and Events

1. **Text Box (Artwork ID)-**
   This will work identically to the previous update ID entry, where selecting a record will autofill the ID and the rest of the details. You cannot type as an **input**.
   This satisfies requirement 5.6 and 5.7.

2. **(Custom) Numeric Up Down (Sale Price)-**
   This will be a custom **class** that will allow me to add a pound symbol to the start of the value in the numeric up down. It will allow the user to update the price of an artwork, but only one that is advanced.
   This satisfies requirement 5.6 and 5.7.

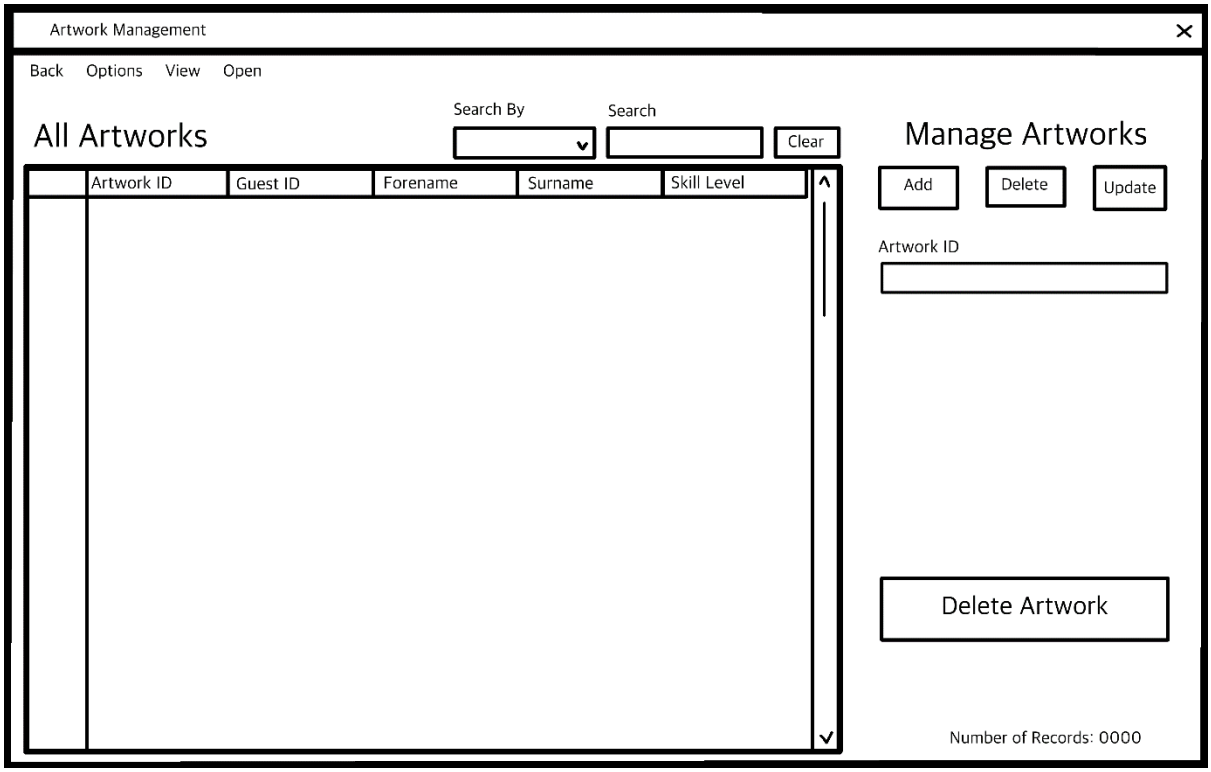
3. **Combo Box (Exhibition)-**
   This **combo box** will populate with available exhibitions to allow the user to change the artwork's exhibition. It will be of drop-down type **DropDownList**, to avoid **input** errors.
   This satisfies requirement 5.6 and 5.7.

4. **Label (Current Exhibition)-**
   This **label** will change to show the current exhibition to allow the user to see the current one before changing it in case it was a past exhibition that is not available in the **combo box.**
   This satisfies requirement 5.6 and 5.7.

5. **Check Box (Artwork Sold?)-**
   This will allow the user to mark an artwork as sold, but once checked and updated it cannot be unchecked.
   This satisfies requirement 5.6 and 5.7.

6. **Button (Update Artwork)-**
   This **button** will function identically to the exhibition update **button**, but for updating an artwork. Only advanced artworks can be updated.
   This satisfies requirement 5.6 and 5.7.

c. Delete Mode



Controls and Events

1. **Text Box (Artwork ID)-**
   This will work identically to the Exhibition ID **Text Box** on the exhibition manager, selecting from the **data grid view** will fill the box. You cannot type as an **input** to minimise error.
   This satisfies requirement 5.5.

2. **Button (Delete Artwork)-**
   This **button** will work identically to the **buttons** like it but will perform a check that the artwork is not linked to past exhibitions. If the artwork is deleted, it will be **cascade** deleted from all other locations in the database.
   This satisfies requirement 5.5.

Testing

Test Plan

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
Login Page							
1.1	Does Login Button work	10	Enter Correct Login details, Enter incorrect Login details, Enter nothing	VALID: Username "LakesideEscapes" Password "Passw0rd" INVALID: Username "xxx" Password "xxx" EXTREME: Enter nothing	1.Successful login and menu should open. 2.Failed login error message should pop up. 3.Failed login error message should pop up	Pass	
1.2	Does visibility button work	10	Type anything into the password box and click on the visibility eye two times.	"Password" and click button	The first click should make the characters visible, and the second should hide them again	Pass	
1.3	Does the options work	1	Click on the options button in the menu strip and 1. Click on exit 2. Click on Login Details	Button Click	When clicking on options a second menu should drop down. Clicking on exit should prompt user confirmation. Confirmation should close the application. Clicking on Login Details will open a popup that shows the username and password.	Pass	
Main Menu							
2.1	Does Pod Booking button work	2	Click on the pod booking button	Button Click	The maintenance form should open	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
2.2	Does Painting button work	2	Click on the painting button	Button Click	The maintenance form should open	Pass	
2.3	Does Weekend Workshops button work	2	Click on the weekend workshops button	Button Click	The maintenance form should open	Pass	
2.4	Does Yoga and Meditation button work	2	Click on the yoga and meditation button	Button Click	The maintenance form should open	Pass	
2.5	Does Pottery button work	2	Click on the pottery button	Button Click	The Pottery Course menu should open	Pass	
2.6	Does Inventory button work	2	Click on the inventory button	Button Click	The maintenance form should open	Pass	
2.7	Does Options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
Maintenance							
3.1	Does the Done	2	Click on the done button	Button Click	The maintenance popup should close and the main menu should reopen.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	button work						
Pottery Course Menu							
4.1	Does the Main Menu button work	1	Click on the Main Menu button in the menu strip	Button Click	The current menu should close and the main menu should open	Pass	
4.2	Does the options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
4.3	Does the Guest Management button Work	2	Click on the Guest Management button	Button Click	The Guest Management menu should open	Pass	
4.4	Does the Course Management button work	2	Click on the Course Management button	Button Click	The Course Management menu should open	Pass	
4.5	Does the Exhibition Management button work	2	Click on the Exhibition Management button	Button Click	The Exhibition Manager should open	Pass	
4.6	Does the Artwork Management	2	Click on the Artwork Management button	Button Click	The Artwork Manager should open	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	button work						
Guest Management Menu							
5.1	Does the Back button work	1	Click on the Back button in the menu strip	Button Click	The current menu should close and the pottery menu should reopen	Pass	
5.2	Does the Options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
5.3	Does the Guest Manager button work	2	Click on the Guest Manager button	Button Click	The Guest Manager form should open	Pass	
5.4	Does the Log Equipment button work	2	Click on the Log Equipment button	Button Click	The Log Equipment form should open	Pass	
5.5	Does the Guest Invoice button work	2	Click on the Guest Invoice button	Button Click	The ID entry form for the Guest Invoice should open	Pass	
Guest Management							
6.1	Does Back button work	1	Click on the back button in the menu strip	Button Click	The current form should close and the Guest	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
					Management menu should open.		
6.2	Does Options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
6.3	Does View button work	1, 3.3	Click on the View button, and then the Sort by Skill option. Then click on Sort By Country	Button Click	Clicking on view should open a second menu. Selecting Sort By Skill should cycle through the skill levels each time it is clicked, putting each one at the top of the records. Sorting by country should organise records by the country, a to z.	Pass	
6.4	Does Open button work	1, 4.7	Click on the Open button and then 1. Guest Management, 2. Log Equipment, 3. Guest Invoice, 4. Course Management, 5. Course Manager, 6. Make a booking, 7. Get a course report, 8. Exhibition Management, 9. Artwork Management and 10. Main Menu.	Button Click	Clicking Open should open a submenu. Clicking on each option should open the relevant menu or form, which will be titled the button that was clicked. If Make a Booking is clicked between dec 20th and Jan 20th, it will not open and an error will pop up explaining bookings cannot be made at this time.	FAIL	Pg 82
6.5	Does Search	3.2, 11	Click on the combo box and select, 1. Guest ID,	VALID: Forename search "Sam",	Selecting an option should show the relevant text box. Guest ID should	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	Type select combo box work AND Does typing in the search bar work		2. Forename and 3. Surname. After Surname and Forename enter Sam into the text box.	Surname Search "Sam" Testing ability to select and type so no invalid or extreme.	result in a text box with a G. Forename and typing Sam should bring up forenames including Sam. Typing Sam in Surnames should bring up surnames including Sam.		
6.6	Does Clear search button work	3.2	Type in the search bar and then click the Clear button. Repeat for each search type, Forename, Surname and Guest ID.	VALID: GuestID "2240", Forename "Sam" and Surname "Sam" Button Click therefore no invalid or extreme.	Each time clear is clicked, the search bar should empty and the records should default back to everything being shown.	Pass	
6.7	Does clicking on the Data Grid View have the desired effect in each mode	3.9	Select 1. Add, 2. Delete and 3. Update and for each click on a record in the data grid view.	Click on top record.	In add mode, you should be unable to select a record and nothing will happen. In delete mode, the record will highlight grey and the guest ID text box will autofill with the record's Guest ID. In update mode, the record will be selected, highlighted in grey, and all information will be filled into the text boxes.	Pass	
6.8	Can you update skill level	3.7	Enter Update mode and click on the records, G1780 and G1785.	Only selection so no invalid or extreme. Click records "G1780" and "G1785"	G1780 should disable skill changes as they have future bookings. G1785 should allow you to change the skill level as they have no future bookings.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
6.9	Can you enter add mode	3.4	Click on Add button	Button click	The form should enter add mode. The Add button and data entry text boxes and combo box should appear. The add button will disable.	FAIL	Pg 83
6.10	Can you enter delete mode	3.5	Click on the Delete button	Button click	The form should enter delete mode. Data entry text boxes and combo box will disappear and one Guest ID box will appear. The delete button will disable.	Pass	
6.11	Can you enter update mode	3.6	Click on the Update button	Button click	The form should enter update mode. Data entry text boxes and combo box should be visible and the update button will disable.	FAIL	Pg 83
6.12	Can you type non-numeric characters into the phone number text box	11	Enter Add mode and attempt to type symbols and letters.	VALID: "1234567890" INVALID: "!@#$%^&*()_querty"	The numbers should appear but symbols and letters will be deleted immediately.	Pass	
6.13	Can you delete the + in the phone number text box	11	Enter Add mode and try to delete the + in the phone number text box	Testing backspace, so no valid invalid or extreme, just keypress.	This should be impossible.	Pass	
6.14	Can you type in the delete mode guest	3.9	Enter Delete mode and try to type into the Guest ID Box.	Testing disabled text so no valid invalid or extreme. Attempt to type "G2240"	This should be impossible. The box can only be filled by selecting a record from the data grid view.	FAIL	Pg 84

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	ID text box						
6.15	Does the Add Guest button work as intended	3.4, 9	Enter add mode and click the Add guest button with 1. No information filled into the text boxes, 2. Some info but some left blank in the text boxes, 3. Full details	VALID: "Krist" "Novoselic" "2064712070" "2206 Wright Road" "Seattle" "USA" "Advanced" INVALID: See valid, but remove one at a time and try after each.	With no details a popup should open asking the user to enter all details carefully. With missing details, a popup should occur asking the user to enter all details carefully. Entering the details example will result in a popup saying "Krist Novoselic successfully added" and the record number at the bottom of the form should increase. The record will be visible at the bottom of the data grid view.	FAIL	Pg 85
6.16	Does the Delete Guest button work as intended	3.5, 9	Enter delete mode and click the Delete Guest button with 1. No record selected and 2. Krist Novoselic selected 3. With Guest G1780 selected.	VALID: ID of the record created in test above. INVALID: No selection EXTREME: Select guest "G1780"	With no selection, the button should be disabled and you should be unable to press it. With Krist Novoselic selected, a pop up will appear asking for user confirmation to delete. Selecting yes will result in a popup will appear saying he was successfully removed. The Data grid view will refresh and the record count will decrease. All past and future bookings will be cascade deleted. With G1780 selected, a popup will open asking for user confirmation. Clicking yes will result in a popup saying guests with existing artwork	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
					cannot be removed. Artworks must be removed first.		
6.17	Does the Update Guest button work as intended	3.6, 9	Enter update mode and select the record for Krist Novoselic. Remove a detail and click update. Then change the detail and click update.	Select record of Record created above. If deleted, recreate with same info. Change skill type to "Beginner" and house number in address line 1 to "2200" from 2206.	Removing a detail will prompt the user to check all fields are filled. Changing the detail and clicking the button will prompt a popup saying update successful, and the data grid view will refresh to show the new details.	Pass	
6.18	Does the Custom Exception for sort country work	3.10	Add an invalid address to the system manually, and then sort by country	Button click	The exception message box should pop up but the system should not crash.	Pass	
Log Equipment							
7.1	Does back button work	1	Click the back button in the menu strip	Button click	The current form should close and the Guest Management menu should open	Pass	
7.2	Does Options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
7.3	Does Open button work	1, 4.7	Click on the Open button and then 1. Guest Management,	Button click	Clicking Open should open a submenu. Clicking on each option should open	FAIL	Pg 82

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
			2. Guest Management, 3. Guest Invoice, 4. Course Management, 5. Course Manager, 6. Make a booking, 7. Get a course report, 8. Exhibition Management, 9. Artwork Management and 10. Main Menu.		the relevant menu or form, which will be titled the button that was clicked. If Make a Booking is clicked between dec 20th and Jan 20th, it will not open and an error will pop up explaining bookings cannot be made at this time.		
7.4	Does Guest ID text box function as intended	7, 11	Click on the guest ID text box and 1. Try to delete the G, 2. Type 9999 and 3. Type 1780.	VALID: "1780" INVALID: "9999"	You should not be able to delete the G. Typing 9999, an incorrect Guest ID should have no impact. Typing 1780, a correct guest ID, should populate the Course ID and bring up the Guest's name.	Pass	
7.5	Does Course ID combo box function as intended	7	Click on the Combo Box for Course ID while 1. No guest is entered, 2. A correct guest has been entered.	VALID: "2240" INVALID: no entry	If there is no guest, or the guest has not attended a course, there will be no options. If the guest is correct and has a course or courses, you should be able to select one. Once selected, it should make the item type combo box available. You should not be able to type in the combo box.	Pass	
7.6	Does Item Type combo box function as intended	7	Select each option from the combo box, 1. Clay, 2. Glaze and 3. Paint.	Button click	Each should make the amount and item name boxes visible. Next to the label for amounts, selecting clay should say pounds, glaze should say pints and paint should say ounces. You should	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
					not be able to type in the combo box.		
7.7	Does Item Name combo box function as intended	7	Click on the combo box item name for each item type 1. Clay, 2. Glaze, 3. Paint.	Button click	Each type should populate the item name box differently. Clay results in earthenware, stoneware, porcelain and raku. Glaze should result in Majolica, celadon, crystalline and raku. Paint should result in Overglaze, underglaze, majolica and metallic. You should not be able to type in the combo box.	Pass	
7.8	Does Amount Numeric Up Down function as intended	7	1. Click on the amount entry and try to type in value. 2. Click on the upwards arrow on the numeric up down 3. Click on the downwards arrow on the numeric up down	Testing something disabled so no valid invalid or extreme. Try to type 15.	You should be unable to type in the amount or delete a number using delete or backspace. Clicking the up arrow should increase the value by 0.25. Clicking the downward arrow should decrease the value by 0.25. you should not be able to select a negative number.	Pass	
7.9	Does Log Equipment button work	7, 9	Click Log Equipment when 1. No guest has been selected, 2. A guest has been selected but no course, 3. A guest and a course has been selected but no items, 4. Guest, course and items Clay, earthenware, have been selected but no amount,	VALID: 4. “1780”, “002”, “clay”, “earthenware” “0.25” INVALID: 1.no data, 2. ID “1780” 3. ID “1780”, Course “002” 4. ID “1780”, Course “002”, “Clay”,	1.Results in a popup stating that details are missing, 2. Results in a popup stating details are missing, 3. Popup stating details are missing, 4. Popup asking to select an amount used, 5. Popup asking you to select an item name, 6. A popup stating equipment has been successfully logged.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
			5. Guest, course, and amount but no items, and 6. All details entered.	"Earthenware", no amount.			
Guest Invoice ID Entry							
8.1	Does back button work	1	Click the back button in the menu strip	Button click	The current form should close and the guest management menu should open.	Pass	
8.2	Does Guest ID entry work	8, 11	Type 1. A valid ID with courses 2. A valid ID without courses and 3. An invalid ID	VALID: Guest "2240" Course "020" INVALID: Guest "9999" EXTREME: Guest "1785"	A valid ID with courses should enable the print report button and populate the course ID box. A valid ID with no courses will prompt a label that the guest has no courses. An invalid ID should have no impact.	Pass	
8.3	Does Course ID selection work	8	Click on the course ID combo box and select an option	Button click	There should be at least one course available if the no courses popup has not occurred. Selecting a course should enable the print button.	Pass	
8.4	Does print report button work	8, 9	Click the print report button	Button click	The current form should close and the invoice form should open.	Pass	
Guest Invoice							
9.1	Does the Done button work	8	Click the done button	Button click	The current form should close and the guest management menu should open.	Pass	
Course Management Menu							
10.1	Does back button work	1	Click on back button in menu strip	Button click	Current form will close and pottery course menu will open	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
10.2	Does Options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
10.3	Does Course Manager button work	2	Click Course Manager button	Button click	Current form should close and the course manager should open	Pass	
10.4	Does Make a Booking button work	2	Click make a booking button	Button click	Current form should close and the make a booking form should open unless it is between 20th Dec and 20th Jan. Between these dates a popup will appear stating bookings cannot be placed at this time.	Pass	
10.5	Does Get Course Report button work	2	Click get course report button	Button click	Current form should close and the course ID entry for report form should open.	Pass	
Course Management							
11.1	Does Back button work	1	Click the back button in the menu strip	Button click	The current form should close and the course menu should open	Pass	
11.2	Does Options	1	Click on the options button in the menu strip and 1. Click on Logout,	Button click	When clicked the dropdown menu should appear. Clicking Logout should return to the login	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	button work		2. Click on Exit, 3. Click on Help.		screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.		
11.3	Does Open button work	1, 4.7	Click on the Open button and then 1. Guest Management, 2. Guest Management, 3. Log Equipment, 4. Guest Invoice, 5. Course Management, 6. Make a booking, 7. Get a course report, 8. Exhibition Management, 9. Artwork Management and 10. Main Menu.	Button click	Clicking Open should open a submenu. Clicking on each option should open the relevant menu or form, which will be titled the button that was clicked. If Make a Booking is clicked between dec 20th and Jan 20th, it will not open and an error will pop up explaining bookings cannot be made at this time.	FAIL	Pg 82
11.4	Does search radio button work	4.3	Click the radio button without changing the search dates	Button click	The data grid view should go blank with a no results label	Pass	
11.5	Does selecting a start date for search work	4.3	Select a start date 1. Before the current end date and 2. After the current end date	VALID: Before the current end date. Validity based on today's date. INVALID: After the current end date. Validity based on today's date.	Before should have no effect but the results in the data grid view may change. After will cause the end date to change to match it.	Pass	
11.6	Does selecting	4.3	Select an end date	VALID:	Selecting before prompts an error message that end	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	an end date for searching work		1. Before the current start date, and 2. After the current start date	After the current start date. Validity based on today's date. INVALID: Before the current start date. Validity based on today's date.	cannot be earlier than start. Selecting after as no effect but search results may change.		
11.7	Does switching into add mode work	4.4	Click Add mode button	Button click	The form should switch into Add mode and then disable the add button. Detail input controls should become visible.	Pass	
11.8	Does switching into delete mode work	4.5	Click Delete mode button	Button click	The form should switch into Delete mode and only a course ID text box should be visible. The Delete mode will become disabled.	Pass	
11.9	Can you select a record in each mode	4.5	Click on a record in 1. Add mode and 2. Delete mode	Selecting a record therefore no valid and invalid. Select the top record in the table.	In add mode, nothing should happen. In delete mode the record should highlight in grey and the course ID delete field should populate accordingly.	Pass	
11.10	Do the course type tick boxes work	4.4	Click one tick box, and then the other.	Button click	The first should tick, and when you click the second, the first should untick and the second tick.	Pass	
11.11	Does the start date selection work	4.4	Click on the date time picker and select a 1. past date, 2. occupied date and 3. future, free date.	All input validity based off of actual date. VALID: Future, free. INVALID: Past or occupied	A past date prompts a popup that date must be future. An occupied date prompts a popup that the date is already booked and a future free date allows you to select it with no popup.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
11.12	Does skill type selection work	4.4	Click weekend and open the skill type. Then click five day and open the skill type.	Button click. Verifying correct conditions, no input.	Weekend should only populate with beginner and intermediate. Five-Day should populate intermediate and advanced.	Pass	
11.13	Does add course work when the year is fully booked	4.4	Try to add a course with 1. Weekend, 9/3/24, beginner 2. Weekend 9/3/24, intermediate, 3. Five-Day, 11/3/24, intermediate, 4. Five-Day, 11/3/24, Advanced	All are invalid. This is checking errors are correct.	All should prompt a pop up that there is no room in the year 2024 for another course of that skill type.	Pass	
11.14	Does add course work when details are missing	4.4	Try to add a course with 1. no selected skill type and 2. No selected course type.	All are invalid. This is checking errors are correct.	A popup will occur stating you need to select a skill type or you need to select a course type as it determines the cost.	Pass	
11.15	Does add a course validate start day	4.4	Try to add a weekend course that starts on a Thursday in 2025, and a five-day course that starts on a Saturday in 2025	All are invalid. This is checking errors are correct.	The weekend course will prompt it needs to start on a Saturday. The five-day course will prompt it needs to start on a Monday.	Pass	
11.16	Can you type in delete course ID	4.5	Open delete mode and try to type in the course ID field	Testing a disabled feature is disabled. Try to type 020.	You should not be able to type in the field.	Pass	
11.17	Does delete course button work	4.5, 9	Select and delete a course that 1. Has no participants 2. Has participants 3. Don’t select a course and then click delete	VALID: Click on “C006” INVALID: Click on “C002” EXTREME: no selection	No participants should delete successfully, participants should popup that booked courses cannot be deleted. No selection should prompt a popup to select a course.	Pass	
11.18	Does Add course	4.4, 9	Click the add button with correct details,	Only testing valid add. “Weekend”,	This should result in a popup saying course added successfully.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
	button work		weekend, 01/03/25 and beginner.	"01/03/25", "Beginner".			
Make A Booking							
12.1	Does back button work	1	Click on back button in menu strip	Button Click	The current form should close when clicked, and the course menu should open.	Pass	
12.2	Does options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
12.3	Does open button work	1	Click on the Open button and then 1. Guest Management, 2. Guest Management, 3. Log Equipment 4. Guest Invoice, 5. Course Management, 6. Course Manager, 7. Get a course report, 8. Exhibition Management, 9. Artwork Management and 10. Main Menu.	Button Click	Clicking Open should open a submenu. Clicking on each option should open the relevant menu or form, which will be titled the button that was clicked.	FAIL	Pg 82
12.4	Is record selection disabled	4.6	Click on records in the Data Grid View	No input, testing selection. Click top record.	This should do nothing.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
12.5	Does guest ID entry work	4.6, 4.8	Enter 1. An invalid Guest ID 2. A valid guest ID	VALID: "2240" INVALID: "9999"	A valid ID should make the course selection combo box visible, filter the data grid view, and display the guest's name. An invalid ID will have no effect.	Pass	
12.6	Does course ID combo box work	4.6	Select a course for guest 2240 that 1. Has not been booked and 2. Has been booked	VALID: "C019" INVALID: "C020"	The non booked one should enable the make booking button. The booked one should display a message that a booking already exists.	Pass	
12.7	Does Make booking button work	4.6, 9	Click the make booking button	Button click	The message that booking has been successfully made should pop up.	Pass	
12.8	Can Guest ID G be deleted	11	Backspace or delete the G in the Guest ID Text Box	Testing delete disabled, no input.	You should be unable to do this.	Pass	
Course Report ID Entry							
13.1	Does back button work	1	Click on the back button in the menu strip	Button click	The current form should close and the course management menu should open	Pass	
13.2	Does Course ID entry work	4.2	Type 1. A valid course ID and 2. An invalid course ID	VALID: "002" INVALID: "999"	Th valid ID should enable the print report button. The invalid should have no effect.	Pass	
13.3	Does print report button work	4.2, 9	Click the print report button	Button click	The current form should close and the report form should open	Pass	
Enrolment Report							
14.1	Does the Done Button work	4.2	Click the done button	Button click	The current form should close and the course management menu should open	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
Exhibition Management							
15.1	Does back button work	1	Click the back button in the menu strip	Button click	The current form should close when clicked, and the course menu should open.	Pass	
15.2	Does options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit, 3. Click on Help.	Button click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.	Pass	
15.3	Does view button work	1, 6.5	Hover over view after loading, and then click on each option. Then enter delete mode and do the same. Then enter update mode and do the same.	Button click	Hovering should display options for past, future and all exhibitions. Clicking should change to the relevant data grid view. In delete and update mode, you should be unable to leave future view.	Pass	
15.4	Does open button work	1, 4.7	Click on the Open button and then 1. Guest Management, 2. Guest Management, 3. Log Equipment, 4. Guest Invoice, 5. Course Management, 6. Course Manager, 7. Make a Booking, 8. Get a course report 9. Artwork Management and 10. Main Menu.	Button click	Clicking Open should open a submenu. Clicking on each option should open the relevant menu or form, which will be titled the button that was clicked. If Make a Booking is clicked between dec 20th and Jan 20th, it will not open and an error will pop up explaining bookings cannot be made at this time.	FAIL	Pg 82

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
15.5	Can you select records in All Exhibition view	6.1	Click on records in the data grid view in the All Exhibitions view. Repeat in add, update and delete mode.	Record selection, no input just click top record in data grid view.	In Add mode you should be unable to select a record. In update and delete, you should be unable to access All records.	Pass	
15.6	Can you select records in Past Exhibition View	6.1	Click on records in the data grid view in the Past Exhibitions view. Repeat in add, update and delete mode.	Record selection, no input just click top record in data grid view.	In Add mode you should be unable to select a record. In update and delete, you should be unable to access Past records.	Pass	
15.7	Can you select records in Future Exhibition view	6.1	Click on records in the data grid view in the Future Exhibitions view. Repeat in add, update and delete mode.	Record selection, no input just click top record in data grid view.	In Add mode you should be unable to select a record. In update and delete, you should be able to select records to autofill information into the fields.	Pass	
15.8	Does Add mode work	6.2	Click on the add mode button.	Button click	Clicking the button should flip the form into add mode. It should disable the add mode button and enable the other mode buttons.	Pass	
15.9	Does Delete mode work	6.3	Click on the delete mode button.	Button click	Clicking the button should flip the form into delete mode, and the data grid view should change to future view. The button should disable, and the other mode buttons will enable.	Pass	
15.10	Does update mode work	6.4	Click on the update mode button.	Button click	Clicking the button should flip the form into update mode and change the data grid view to future view. The button clicked will disable and	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
					the other two mode buttons will enable.		
15.11	Can you select a year in add mode	6.2	Enter add mode and select a year using the arrow buttons. Try 2000, 2024 and 2050.	Button click VALID: 2050 INVALID: 2024 EXTREME: 2000	You should be unable to type. You should be unable to select a year below the current, so 2000 is invalid. 2024 should be red, as it has already been used. 2050 should be available.	Pass	
15.12	Can you change the year using the month picker	6.2	Try to change the year or month by clicking the month.	Button click	It should not do anything.	Pass	
15.13	Can you select a date	6.2	Select the year 2050 and try to select the 12 December. Then try 2024 and 12 December.	Button click VALID: 12 Dec 2050 INVALID: 12 Dec 2024	You should be able to select a date and the add button should be available for 2050, but 2024 should not allow interaction with the month picker.	Pass	
15.14	Does the Add exhibition button work	6.2, 9	Enter valid details and click the add button. Then try to select a date and year that is already in the data grid view, as invalid data.	VALID: 2033, Dec 3 INVALID: 2024, any date.	The valid details should prompt a popup stating successfully added, and invalid should result in the add button being disabled.	Pass	
15.15	Does typing in Exhibition ID text box work	6.3, 11	Enter delete mode and attempt to type in the Exhibition ID box.	Testing inability to type. Try to type 1115.	You should be unable to type.	Pass	
15.16	Does Exhibition ID entry work	6.3, 9	Enter delete mode and attempt to click on a record in the data grid view	Button click, select top record.	The record ID should fill the Exhibition ID text box.	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
15.17	Does Delete exhibition button work	6.3, 9	Enter delete mode, select a record without artworks and press delete. Then select one with artworks and press delete.	VALID: "EX039" INVALID: "EX015"	Selecting a record should populate it and clicking delete should result in a delete successful popup. Selecting an exhibition that has artwork should result in a popup saying this is not possible.	Pass	
15.18	Does Typing in exhibition ID work	6.4	Enter update mode and attempt to type in the exhibition ID box.	Testing inability to type. Try to type 015.	You should be unable to type.	Pass	
15.19	Can you change the year using the month picker	6.4	Enter update mode and attempt to change the year by clicking on the month in the month picker	Button click	You should be unable to click it and there should be no event.	FAIL	Pg 86
15.20	Does the catering number numeric up down work	6.4	Enter update mode and attempt to type a new catering number. Then use the arrow buttons to adjust the catering number.	Testing inability to type. Try to type 20.	You should be unable to type, but the arrow buttons should function, raising the number by 1 per click.	Pass	
15.21	Does update exhibition button work	6.4, 15	Enter update mode, select a record, change each of the available values, and click the update button.	Only valid. Select the existing top record and change each value.	The changes should be successful and be visible in the data grid view.	Pass	
Artwork Management							
16.1	Does back button work	1	Click the back button in the menu strip	Button click	The current form should close when clicked, and the course menu should open.	Pass	
16.2	Does options button work	1	Click on the options button in the menu strip and 1. Click on Logout, 2. Click on Exit,	Button Click	When clicked the dropdown menu should appear. Clicking Logout should return to the login screen. Clicking exit	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
			3. Click on Help.		should prompt confirmation from the user and then close the application. Clicking Help should open the User Guide PDF.		
16.3	Does view button work	1, 5.8	Hover over view after loading, and then click on each option. Then enter delete mode and do the same. Then enter update mode and do the same.	Button Click	Hovering should display options for all and advanced artworks. Clicking should change to the relevant data grid view. In update mode, you should be unable to leave advanced view.	Pass	
16.4	Does open button work	1, 4.7	Click on the Open button and then 1. Guest Management, 2. Guest Management, 3. Log Equipment 4. Guest Invoice, 5. Course Management, 6. Course Manager, 7. Make a Booking, 8. Get a course report 9. Exhibition Management and 10. Main Menu.	Button Click	Clicking Open should open a submenu. Clicking on each option should open the relevant menu or form, which will be titled the button that was clicked. If Make a Booking is clicked between dec 20th and Jan 20th, it will not open and an error will pop up explaining bookings cannot be made at this time.	FAIL	Pg 82
16.5	Does searching work on all view	5.2	Enter all view and select each option from the search by combo box, and for each, type in the search bar.	Testing search, only Valid. Guest ID "2240" Name "John"	Selecting an option and typing should filter down results in the data grid view based on what is entered.	Pass	
16.6	Does searching work in advanced view	5.2	Enter advanced view and select each option from the search by combo box, and for each, type in the search bar. You	Testing search, only Valid. Guest ID "2240" Name "John" Sale Price "45.00"	Selecting an option and typing should filter down results in the data grid view	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
			should not be able to type in the Numeric up down, instead use the arrow buttons.		based on what is entered.		
16.7	Does clear button work in all view	5.2	Click the clear button after searching anything in all view	Button Click	The data should clear out of all the boxes	Pass	
16.8	Does clear button work in advanced view	5.2	Click the clear button after searching anything in advanced view	Button click	The data should clear out of the text boxes	FAIL	Pg 87
16.9	Does add mode button work	5.3	Click the add mode button when not in add mode	Button click	The form should swap into add mode and the add mode button will disable.	Pass	
16.10	Does delete mode button work	5.5	Click on the delete mode button when not in delete mode	Button click	The form should swap into delete mode and the delete mode button will disable	Pass	
16.11	Does update mode button work	5.6	Click on the update mode button when not in update mode	Button Click	The form should swap into update mode and the update mode button will disable	Pass	
16.12	Does Guest ID entry work in add mode	5.3, 11	Type a valid ID with a course, trying beginner, advanced and intermediate guests, a valid ID without a course and an invalid ID into the text box.	VALID: Beginner “1780” Intermediate “2245” Advanced “2240” INVALID: “9999” EXTREME: “1785”	The valid with course should populate the course box. If advanced, a numeric up down will appear. Valid without course will bote that there is no course. Invalid will have no effect	Pass	

Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
16.13	Does course selection work in add mode	5.3	Enter a valid guest ID and select courses from the combo box	VALID: "2240" and "020"	The course should select without issue	Pass	
16.14	Does sale price work in add mode	5.3	Enter an advanced guest ID, a course ID and then change the price of the numeric up down	VALID: "2240", "020", "45.00"	You should be unable to type. Using the arrows you can adjust the cost.	Pass	
16.15	Does Add artwork button work	5.3, 5.4, 9	Enter valid details and click add Artwork. Then try invalid.	VALID: "2240", "020", "45.00" INVALID: "9999"	Valid should successfully add. Invalid should not allow the button to be clicked.	Pass	
16.16	Can you type in delete artwork ID entry	5.5, 11	Enter delete mode and try to type in the ID entry text box. Then try selecting a record from both advanced and all data grid views.	Testing inability to type. Try to type 20. Then click top record in table.	You should be unable to type, but selecting a record should populate the text box.	Pass	
16.17	Does Delete Artwork button work	5.5, 9	Select a record from the data grid view in both modes. Click the delete button. Try one that has been involved in a past exhibition, and one that has not from advanced.	Button Click	If the artwork was in an exhibition it cannot be deleted. If not, it should successfully be deleted.	Pass	
16.18	Can you type in Update artwork ID entry	5.6	Enter update mode and try to type in the ID entry. Then try selecting a record from the data grid view.	Testing inability to type. Try to type 20. Then click top record in table.	You should be unable to type but selecting records from the data grid view should populate the text box.	Pass	
16.19	Does Change Price Numeric up down work	5.6	After selecting a record try typing to change the price, and then use the buttons	Testing inability to type. Try to type 20. Then click top record in table.	You should be unable to type and can change the values using the arrow buttons.	Pass	

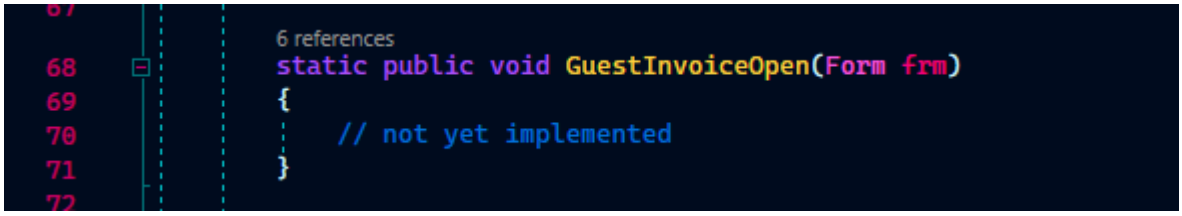
Test No.	Reason	Requirement No.	User Action	Test Data	Expected Outcome	Pass / Fail	Evidence page No.
16.20	Does Exhibition Change work	5.7	Select a new exhibition from the combo box after selecting a record.	Button Click	You should be able to select a new exhibition from the list of future exhibitions.	Pass	
16.21	Does Sold state update artwork	5.7	Select a record that has been sold and try to change it. Then select one that has not.	Testing selection. Select a record with sold checked, and one without.	You should only be able to change this for unsold artwork.	Pass	
16.22	Does Update Artwork button work	5.7, 9	Enter valid data and click the update button. Then select invalid data.	Click record and change exhibition, tick sold, and adjust price by 5.	You should only be able to update an unsold artwork.	FAIL	Pg 88

Test Actions

Test 6.4, 7.3, 11.3, 12.3, 15.4, 16.4

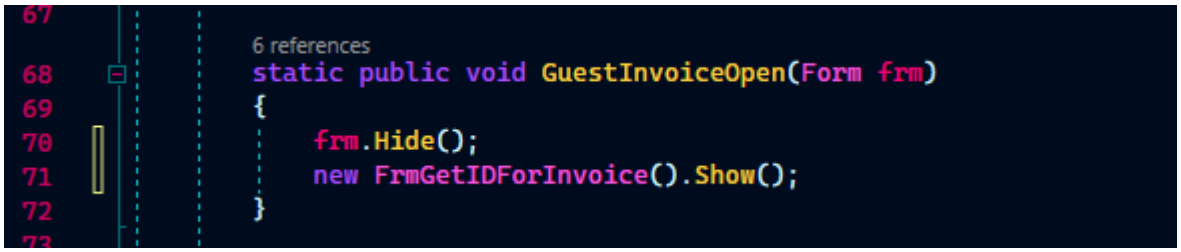
Clicking "Guest Invoice" does nothing. It should open the Guest Invoice ID Entry form. I suspect I had forgotten to replace my placeholder for this event in my common methods class. Evidence cannot be given for this test as there is nothing to see.

Original Code



As you can see above, since I developed my menu strip before my invoice, I forgot to change the click event method that is used in every form. This means by fixing this I should solve every instance of the issue.

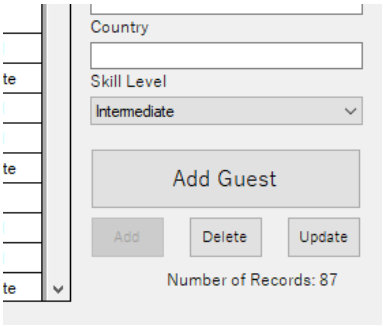
Solution



The solution was very simple. Use the form passed into the method, hide it, and then open the invoice form. This method is already being called in every click event in the solution, so this is the only change I needed to make.

Test 6.9 and 6.11

When swapping into add mode from update mode, or add mode to update mode, the Skill Level combo box does not clear. This does not impact functionality to a large extent, but makes the form look far less clean. Below is how the form looks after moving from Update mode with full details filled into the fields.



Original Code

```
265     // method to empty all controls
        4 references
266     public void EmptyBoxes()
267     {
268         txtForenameAddUpdate.Text = string.Empty;
269         txtSurnameAddUpdate.Text = string.Empty;
270         txtContactNoAddUpdate.Text = "+";
271         txtAddressLine1AddUpdate.Text = string.Empty;
272         txtAddressLine2AddUpdate.Text = string.Empty;
273         txtAddressLine3AddUpdate.Text = string.Empty;
274         cbxSkillLevelAddUpdate.Text = string.Empty;
275         txtGuestIDDelete.Text = "G";
276     }
277
```

Above is the method I use to clear the controls. Upon closer inspection, I discovered setting the text doesn't actually impact the combo box the way I intended. Instead, I need to change the selected index instead of text on line 274.

Solution

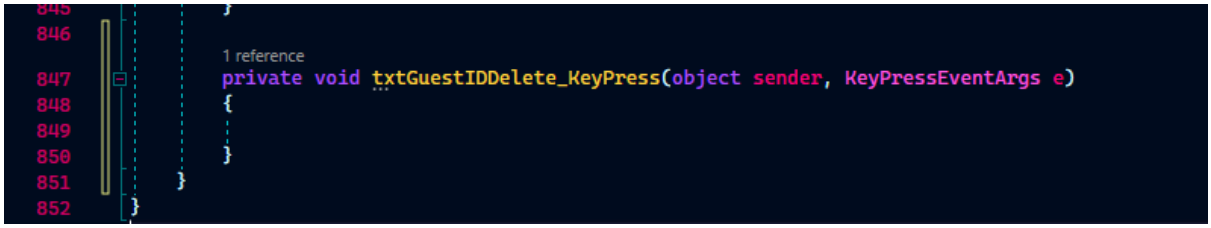
```
265     // method to empty all controls
        4 references
266     public void EmptyBoxes()
267     {
268         txtForenameAddUpdate.Text = string.Empty;
269         txtSurnameAddUpdate.Text = string.Empty;
270         txtContactNoAddUpdate.Text = "+";
271         txtAddressLine1AddUpdate.Text = string.Empty;
272         txtAddressLine2AddUpdate.Text = string.Empty;
273         txtAddressLine3AddUpdate.Text = string.Empty;
274         cbxSkillLevelAddUpdate.SelectedIndex = -1;
275         txtGuestIDDelete.Text = "G";
276     }
277
```

After some quick research on the Microsoft documentation, I discovered the index of -1 will empty my combo box visually, so I changed this on line 274. It now works. Since I use this method on each switch, this one line solves both test 6.9 and 6.11.

Test 6.14

In this test the user should be unable to type in the Guest ID field when in delete mode. This I suspect is due to a simple missing event. This is important to fix, as it helps avoid incorrect or invalid data being passed into a query.

Original Code



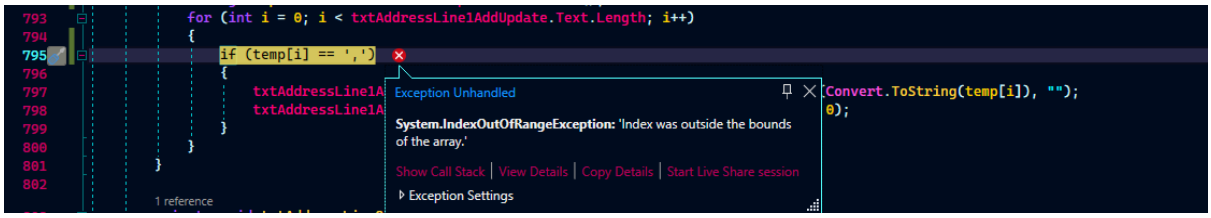
As you can see here, I have forgotten to add the key press event to handle this, as I developed the add form before I had the idea to disable typing to minimise errors.

Solution

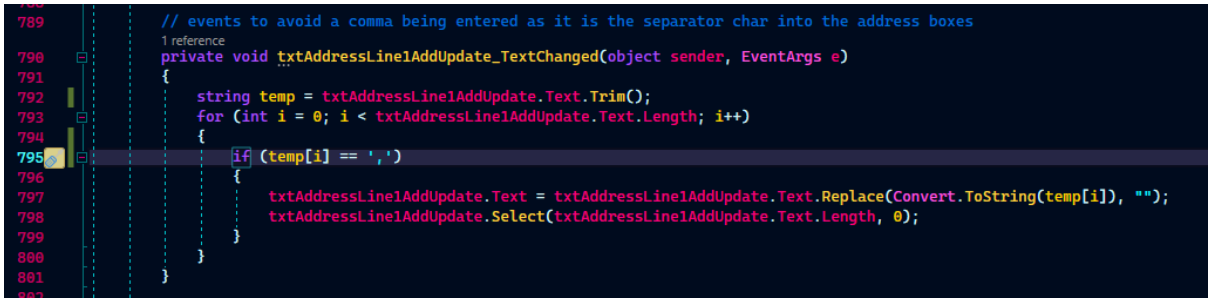
This solution has worked perfectly, using a simple line that tells the system to ignore the keypress.

Test 6.15

The user should be able to type an address into the text box, but without using any commas, as this is the character, I use to split address strings. The user is able to type, and the comma is correctly handled, but pressing space causes the system to throw an out-of-range exception:

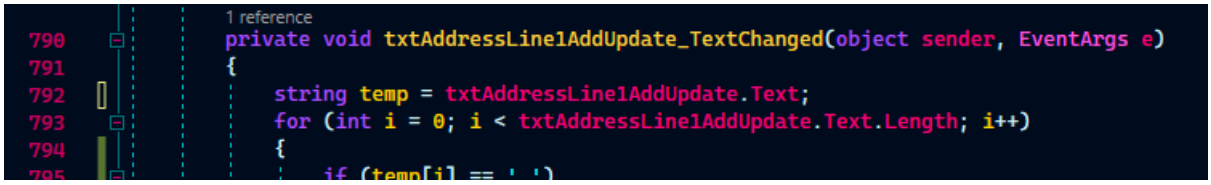


Original Code



After some consideration, I deduced that the issue was that when I check the text in the text box for commas, I use Trim() on line 792, which strips any whitespace from the beginning and end of the string. This means when searching the length of the string, the final index will be out of range, due to the space being stripped off of the string.

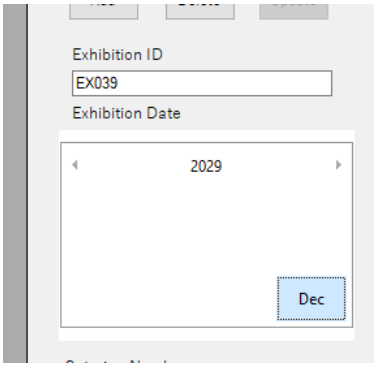
Solution



The solution to this problem was simple, just removing the Trim() method when creating a temporary variable on line 792.

Test 15.19

In a month picker, if you click the header with the month name, the control will back track to show the year, decade, and then century. Since I only want my guests to be able to select a date in December of the specific year, I want to disable the ability to click on this. I succeeded with this in Add mode, but in Update mode, the click is not caught and the user can soft-lock themselves in the year view.



Original Code

```
2 references
335     private void mcExhibitionDate_MouseDown(object sender, MouseEventArgs e)
336     {
337         MonthCalendar.HitTestInfo x = mcExhibitionDate.HitTest(e.Location);
338         if (x.HitArea.ToString() == "TitleBackground")
339         {
340             mcExhibitionDate.Enabled = false;
341             mcExhibitionDate.Enabled = true;
342         }
343     }
```

This is the method called by the event MouseDown on the Update Month Picker. To try and avoid repeating code, I thought I could use the same events for the same type and function of controls, and while this has worked well elsewhere, I now see that the method only references mcExhibitionDate, which is the picker in add mode, not update mode. This is on line 337. To fix this error I will need to create a separate event for the update mode picker.

Solution

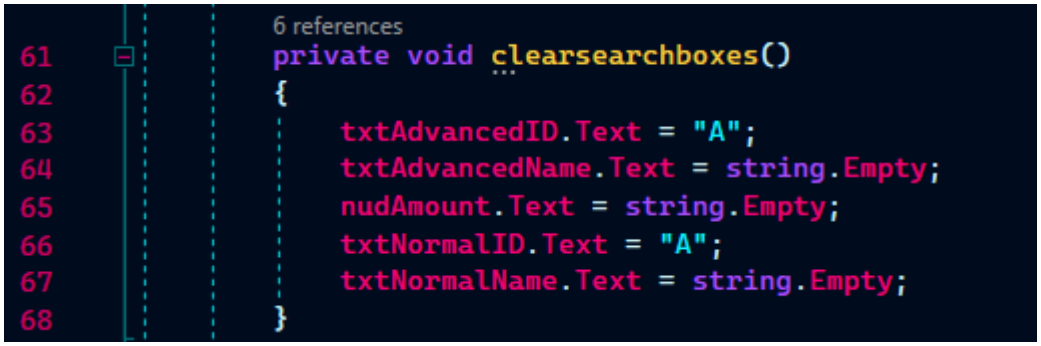
```
1 reference
581     private void mcExhibitionDateUpdate_MouseDown(object sender, MouseEventArgs e)
582     {
583         MonthCalendar.HitTestInfo x = mcExhibitionDateUpdate.HitTest(e.Location);
584         if (x.HitArea.ToString() == "TitleBackground")
585         {
586             mcExhibitionDateUpdate.Enabled = false;
587             mcExhibitionDateUpdate.Enabled = true;
588         }
589     }
590 }
591 }
```

While creating this event, I also noticed that there would have been a different reference required for each picker on lines 340 and 341! These issues are now solved successfully.

Test 16.8

When clicking the clear button on the artwork manager, all searches should be cleared, but in advanced mode, the price search does not clear.

Original Code

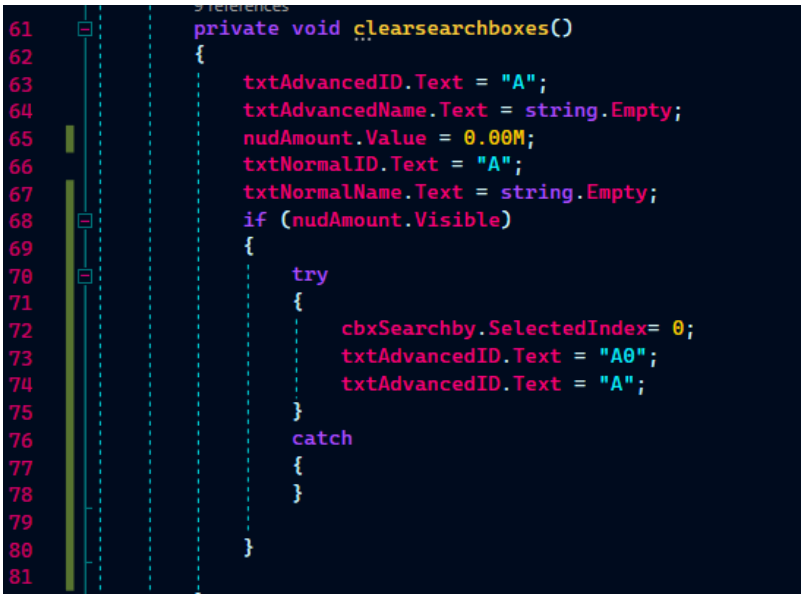


```
6 references
61  private void clearsearchboxes()
62  {
63      txtAdvancedID.Text = "A";
64      txtAdvancedName.Text = string.Empty;
65      nudAmount.Text = string.Empty;
66      txtNormalID.Text = "A";
67      txtNormalName.Text = string.Empty;
68  }
```

This is the method for clearing searches, and initially I was unsure what the issue was as I clearly emptied the Numeric up down on line 65, However I wondered if this works similarly to the combo boxes before. While this did reset the box back to zero, because the search is performed on text changed or value changed, it the performs a search of 0 with no results.

To solve this problem, I want to reset the search back to ID and then clear again.

Solution

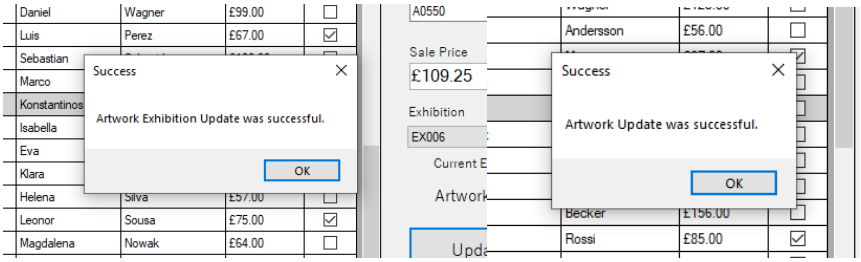


```
61  private void clearsearchboxes()
62  {
63      txtAdvancedID.Text = "A";
64      txtAdvancedName.Text = string.Empty;
65      nudAmount.Value = 0.00M;
66      txtNormalID.Text = "A";
67      txtNormalName.Text = string.Empty;
68      if (nudAmount.Visible)
69      {
70          try
71          {
72              cbxSearchby.SelectedIndex= 0;
73              txtAdvancedID.Text = "A0";
74              txtAdvancedID.Text = "A";
75          }
76          catch
77          {
78          }
79
80      }
81
```

This fix was a little more confusing than I wanted it to be, as I found just setting the search by index to 0, like on line 72, did not work in the middle of a mode change, and will throw an exception. Instead, I placed it inside a try catch, and forced a search with results of all records to occur inside the try. The catch does nothing if it can't be done without exceptions. This works because search is automatically cleared in mode changes anyway.

Test 16.22

This error is less functional and more aesthetic. When updating a record, there are two different message box pop ups to notify the user it was successful. There should only be one, so to solve this I need to find the source of one and just comment it out.



The issue appears to occur when the exhibition is changed, so I suspect it is a debug message I have forgotten to remove.

Original Code

```
int outco                    eArtwork(txtArtworkIDUpdate.Text.Trim(), nudSalePriceUpdate.Value, sold);
if (outcome < 2)
{
    MessageBox.Show("Something went wrong, artwork may not have fully updated. " +
        "\nPlease check the update was successful or try again.", "Failed");
}
else
{
    MessageBox.Show("Artwork Update was successful.", "Success");
}
if (cbxExhibtionEdit.Text.Trim() != "")
{
    int exoutcome = ArtworkDal.UpdateArtworkExhibition(txtArtworkIDUpdate.Text.Trim(), cbxExhibtionEdit.Text.Trim());
    if(exoutcome < 1)
    {
        MessageBox.Show("Something went wrong, artwork Exhibition may not have fully updated." +
            " \nPlease check the update was successful or try again.", "Failed");
    }
    else
    {
        MessageBox.Show("Artwork Exhibition Update was successful.", "Success");
    }
```

As you can see, in my code I perform a check for both successful exhibition update and normal update, as thy are two different procedures. Instead, I am happy to remove one as I am confident this part of my code works, so I can simply comment the message out.

Solution

```
    {
        MessageBox.Show("Something went wrong, artwork Exhibition may not have fully updat
            " \nPlease check the update was successful or try again.", "Failed");
    }
    else
    {
        //MessageBox.Show("Artwork Exhibition Update was successful.", "Success");
    }
}
```

To solve this issue, I simply commented out the exhibition success. If the exhibition fails to update the user will be notified, as it is important. If it is successful only the general success method will show, not both.

Evaluation

Evaluation of Approach

In hindsight, I believe Waterfall was the perfect choice of methodology to develop this project with, and the use of SQL and C# allowed me to produce a unique and customised solution for the client.

The use of waterfall allowed an inexperienced developer like me to produce my solution with a logical, step by step approach. It was also the perfect methodology, as my client's requirements did not change, meaning contact was not required. Overall, it was a successful decision.

The use of SQL meant the system, and I, were able to easily manipulate and retrieve data from a database with a language designed for this purpose, meaning it was straightforward and efficient, allowing me to perform complicated operations.

C# is versatile and robust, as well as having a huge library of tools that make developing both a unique interface and a robust back end easy and efficient.

These languages blend perfectly for my solution, as Visual Studio Code allows me to work with both SQL Server and C# Windows forms in one place, making the experience of development faster and more enjoyable. While the software has its bugs and issues, such as having to restart the solution after every stored procedure update, it is free and open source, making it extremely accessible! This also means there is a huge array of help and discussion about both languages online, allowing me to have the opportunity to solve my issues my reading up on previous problems other developers have had.

Overall, the combination of SQL and C#, as well as the Waterfall methodology was the right choice for my development style, and I am satisfied with the results.

Evaluation of Project Plan

During my Planning and designing of the system, I am pleased that I made the effort to get feedback on my first draft, as this allowed me to alter my design before development, saving me a lot of time in the long run. After receiving feedback, I was able to condense my system into fewer forms, making it more intuitive and user-friendly. This final design was carried through development.

Between the end of my planning phase and the end of my development phase, I am confident that my system stayed loyal to its design and no changes were made, as my client was satisfied with the final design. I also felt I was able to comfortably stick to the planned timeboxes, with the only issue arising at the end of development when I lost a small chunk of work due to a power outage. This, however, only ran a few days over, and while it could have been mitigated by a more thorough risk assessment, had very little impact on the rest of the project, as the float from other tasks meant I had space to breathe.

Evaluation of Testing

In the testing phase of this project, a test plan was designed and developed, tests were carried out, and then failed tests were evaluated and solved. This is done to ensure the system is functional, reliable, and performs well, and overall, I feel my testing process successfully ensured

that the Lakeside Escapes Pottery System is robust, reliable, and reasonably fast.

In the process of creating my test plan, I carefully worked through each form, listing how each control and functionality should function, as well as define invalid and valid test data, and the requirement it fulfils. In hindsight, I believe it would have saved time to list these during development, as this would also reduce the chance of forgetting a test as well as streamline the testing process. Once I had listed the test scenarios, I created a concise table of test number, test, requirement, data, outcome and pass or fail. I feel this was a successful testing table, and made the process of testing simple and easy, as well as making it far easier to see which tests have failed.

When executing the test plan, because of my test table I was easily able to put the test plan into action, one by one filling in each row of the table and noting errors. I am very happy with how quick and easy the table made carrying out the testing process because when I encountered a challenge of the power going out and I lost all my testing results, while this could have been avoided by saving more frequently, the table made it far less painful and tedious to redo all my tests.

Because of the concise pass-fail column and colour of my table, it was clear to see what tests had failed so I could evaluate the issues and solve them. When solving, I was careful to document each change so it would be simple to revert if other errors were caused by the changes.

In conclusion, the meticulous planning and layout of my test plan proved invaluable to my development process, and while noting the tests earlier might have made the design easier, the effort was worth it to minimise the impact of setbacks. I am happy with the result and am confident in the reliability of my system.

Evaluation of the Solution

Looking back at my completed solution, I am extremely happy with what I have achieved. One of the strengths of my system is its intuitive and compact user interface. I am very pleased with my mode switch feature, allowing multiple processes to be carried out on one form with just the toggle of a button.

Another strength is the robustness of my system, as I put a lot of work into avoiding duplicated keys and double booking, for example when logging equipment ensuring the equipment for the same person and same course is added rather than creating a new record, or ensuring the same guest cannot book a course twice, and ensuring courses cannot be booked for the same days.

A third strength is the navigation of my system. I am confident that all features are always easily accessible, whether that be through a menu strip and submenu or through the system menus themselves.

Every system has its weaknesses and limits, and this solution is no different. One limitation of my system is the lack of ability to delete bookings. This would have required more work than I had time for, and for now, bookings can only be placed. Another limitation is the inability to remove equipment that has been logged for a user. While this is unlikely to be as large of an issue as the bookings, it means it is an issue if a tutor logs too much or the wrong item.

An improvement that I would like to have added, perhaps in the future, would be a more detailed admin login, providing the ability to add and remove users, and making the system more secure.

On top of this, it would be interesting to try and develop a report that shows a colour-coded calendar, indicating when each course is booked and how booked up it is at that time. This could allow the client to get an overview of his most successful time of year.

Evaluation of User Requirements

1. *The system requires a menu strip to help the user navigate the system and menus. This will include a logout, exit, and help button, plus the ability to open other forms quickly.*

This requirement has been successfully met but did go through a few changes during development. The Logout and Exit buttons remained the same throughout, but initially, I planned for the help system to be made up of integrated forms. I had planned initially to use GIFs to show how each form is used, but the more complicated and detailed my forms became, the more difficult this was to understand. The image quality of the GIFs also made them hard to read. Halfway through development, I made the executive decision to remove this feature, and instead developed an easy-to-read and pleasant-to-look-at User Guide pdf, added it to resources and then opened that in the user's browser when the help button was clicked.

The menu strip also allows the user to navigate easily to all management forms plus reports and the main menu, which I believe makes the system easier to navigate. The functionality of this menu strip is tested in each form, and the only issue I ran into was forgetting to implement the Guest Invoice, but thanks to my common methods class, fixing tests 6.4, 7.3, 11.3, 12.3, 15.4 and 16.4 was as simple as adjusting a few lines in one method. Some forms also contain their sorting or data views on the menu strip, which is discussed in later requirements, but works very well. I am confident this requirement has been well met at a high standard.

2. *The system should have easy to understand menu that can implement the other systems being developed with ease.*

This requirement has been successfully fulfilled, and I did not encounter any issues. It is tested in tests 2, 3, 4 and 5. I felt the best way to do this would be with a concise main menu, where button events could be changed to implement another full system at any point! I am happy with the standard I have met this requirement.

3. *The user should be able to manage existing and new guests of the pottery course:*
    - 3.1. *The user needs to be able to view all existing guests and their details.*
    - 3.2. *The user needs to be able to enter a name or part of a guest name and view all guests whose name includes the entered characters.*
    - 3.3. *The user needs to be able to sort guests by skill level or country.*
    - 3.4. *The user needs to be able to add a new guest and their information as and when required.*
    - 3.5. *The user needs to be able to delete a guest, but only when that guest has no existing artwork.*
    - 3.6. *The user needs to be able to change a guest's details with ease, but keep the same ID.*
    - 3.7. *If editing a guest's details, you should not be able to change the skill of someone with a future booking, as courses have specific skill types.*
    - 3.8. *If a user chooses to delete a guest who has attachment to courses, their bookings, past and future, should also be deleted.*
    - 3.9. *The user should be able to click on a record when deleting or updating to fill in information in order to mitigate error.*
    - 3.10. *When sorting by country, the format of addresses should be verified using a custom exception to avoid crashing the system.*

This requirement has also been successfully fulfilled and tested in test 6. While tests 6.9, 6.11, 6.14, and 6.15 failed, they were all reasonably easy fixes, as described in their subsequent test actions. When

designing and developing these requirements I also came across a variety of other challenges.

When I initially created my database, I wanted to autoincrement my primary key but add a letter as an identifier for the user, so my keys are not meaningless numbers. To do this I had to use calculated columns that were persisted, and I found it extremely difficult to set up foreign keys with these calculated columns. In the end I discovered that length of data types is very important, and they had to match precisely between each table for the foreign key to work. The code for these columns, specifically guest records, looks like this:

```
[GuestID] AS
(CONVERT([varchar](15), 'G'+right('0000'+CONVERT([varchar](15),[RecordNo]),
(4)))) PERSISTED NOT NULL
```

The next challenge came with trying to add a record, as it appeared that the system wanted me to pass in the primary key instead of it being created for me. This was a very simple fix that I learnt from stack overflow – I needed to make sure Auto-Increment was set to true. I was able to set the default to true so I wouldn't need to worry about this again.

Another problem I had to overcome was when performing a sort by skill on the guest data grid view, I would pass my sorted dataset into the table, and it would be completely blank, but would have the correct number of rows! I struggled with this for a while but learned that the table must have the same data property names in its columns as my dataset did. At this point I also had to consider a way to make search and sort work together – this meant learning how to use IF statements inside SQL stored procedures. This can be seen in my stored procedures BeginnerSort, IntermediateSort and AdvancedSort.

While designing my second sort, by country, I found it very tedious and error-prone to have to rely on splitting the string correctly. Instead, I decided to create a class called Address with attributes for each address line and a constructor with a custom exception to ensure the address is in the right format to sort. If it comes across a 'bad' address, it will cancel the sort and notify the user country sort is not available at this time. As well as the custom exception, this address class also inherits from IComparable, and contains its own custom compareTo methods that allow the address objects to be sorted by country.

While developing the ability to update guest details, I found an issue where when attempting to update a guest I just created, the system would flag the record as one with existing future bookings. This was clearly incorrect, and it turned out that in my condition for determining this was using the stored procedure's name rather than the Boolean variable's name returned by the stored procedure.

I noticed later in the development process when I loaded the form initially, the search function does not work. This was a very quick fix – I force the system to perform a search for the Guest ID "G" which will activate the search function and return all records, giving the illusion nothing has happened.

Another final issue that I noticed, excluding failed tests, was there may be errors arising if a guest skill is changed when they have a future booking. To restrict this, I simply disable the ability to update skill level on guests with future bookings. All other attributes of a guest can still be changed.

After resolving these issues, plus resolving the failed tests, I feel confident that this user requirement is complete and completed to a good standard.

4. *The user needs to be able to manage existing and new pottery courses and the information surrounding them:*
    - 4.1. *The user needs to be able to view all existing pottery courses.*
    - 4.2. *The user needs to be able to enter a course ID, and get a report showing the course, and the names and phone number of the guests enrolled in it, as well as the date the booking was made.*
    - 4.3. *The user needs to be able to enter two dates and view all courses between those dates.*
    - 4.4. *The user needs to be able to create new pottery courses, but only if there is space for it in the current year.*
    - 4.5. *The user needs to be able to delete existing pottery courses, but only if there are no guests assigned to this course.*
    - 4.6. *The user needs to be able to assign a guest to a pottery course, but only if the course has space, to avoid overbooking, and the guest's skill level matches the course. Double booking will be handled by this also.*
    - 4.7. *The booking cannot be made between December 20th and January 20th.*
    - 4.8. *The system should automatically filter out courses that are not suitable to the guest being booked.*

This requirement has been successfully satisfied, and is tested in tests 11, 12, 13 and 14, all of which passed with no issues excluding the menu strip issue mentioned previously. While my final version of this requirement passed tests without issue, I did come across a variety of challenges when developing it.

My first problems involved the course manager forms. The first issue involved the grouping of radio buttons, as on my form I wanted separate radio buttons for the search toggle and the course type selectors. This proved both ugly and difficult to use so I simply overcame it by using the radio button for the search toggle, and some checkboxes for the course type. This removes the requirement for ugly group boxes and fiddly code to separate them.

When first designing my date time picker for selecting a course date, I integrated immediate feedback, meaning a date would be validated when clicked. In the design process this seemed efficient, but in practice it proved irritating and repetitive, especially since dates may throw multiple errors or get stuck in an error loop. At this point, I decided to change this so validity, such as not a past date or a date already assigned, is checked when the date is clicked, but suitability, such as does a five-day start on a Monday, is checked when the add course button is clicked. This proved far less disruptive and annoying to use. The method I created to ensure the date picker never automatically sets a date that will put it in an error loop is something I am very happy with; it looks like this:

```
private void safeDate(DateTimePicker x)
{
    if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Monday")
    {
        x.Value = DateTime.Today.AddDays(3);
    }
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Tuesday")
    {
        x.Value = DateTime.Today.AddDays(2);
    }
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Wednesday")
    {
        x.Value = DateTime.Today.AddDays(1);
    }
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Thursday")
    {
        x.Value = DateTime.Today.AddDays(7);
    }
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Friday")
    {
        x.Value = DateTime.Today.AddDays(6);
    }
```

```
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Saturday")
    {
        x.Value = DateTime.Today.AddDays(5);
    }
    else if (DateTime.Today.AddDays(1).DayOfWeek.ToString() == "Sunday")
    {
        x.Value = DateTime.Today.AddDays(4);
    }
}
```

Another challenge I was stuck on for a while was how to force one check box to untick when the other is checked. In the end, I found it best to have a state check for each on the "onChecked" and "Click" events, which manage the current state before changing the states. This avoids the issue where the state would change before it was actually checked, resulting in nothing happening.

When I began working on my date validation, I also forgot to include a check for days that are already in use! To fix this, I had to run a stored procedure query to check for existing date, and provide feedback based on the result. This ensures double booked dates are avoided.

When creating my course enrolment report, I ran into a host of annoying issues, as this was the first more complicated report I had made. My first problem arose when passing variables into the report view. I first had the same blank table issue from before, meaning I had to be careful that my headers had the same data property name when they were passed in. secondly, the date values pulled from the database were automatically converted into datetime, meaning they had a timestamp I found impossible to remove. Initially I tried to change it to string and then substring to remove the timestamp, but this caused the blank table error again, as the query works on the basis the result is date. The final solution is not perfect, but by restricting the row height the time is unable to be visible. This does make the table look slightly odd, but it works.

When developing the ability to place a booking, the only database related issue I ran into was that my time booked had a strange decimal after it. This was due to accidentally setting the datatype to 8 decimal places instead of total length 8.

Since I was able to solve these during development and my testing process went off without a hitch, I am happy with the quality this requirement has been fulfilled to.

5. The user needs to be able to manage existing and new pieces of artwork:

- *5.1. The user needs to be able to view a record of all existing artwork, and the information about it.*
- *5.2. The user needs to be able to search for an artwork ID, and show the Artwork ID, Guest name, and price, as well as whether it has sold or not. The price should be deducted from the Guest's first booking after the exhibition.*
- *5.3. The user needs to be able to create records of a piece of artwork but only I there is a guest to be associated with it at the time of creation, and that guest has been booked into at least one course.*
- *5.4. When an artwork is created for an advanced guest, it should be automatically added to the nearest future exhibition.*
- *5.5. The user needs to be able to delete a piece of artwork. If it is deleted, it should be cascade deleted from all other places. If it is linked to a past exhibition, it cannot be deleted.*
- *5.6. The user needs to be able to edit the price of an advanced artwork.*
- *5.7. The user needs to be able to change an artwork's exhibition, but only if the artwork was created by an advanced guest, has not sold, and the new exhibition has not happened.*

5.8. The system should force the user into the advanced view when in update mode, as advanced artworks are the only ones with details that can be edited.

This requirement has been successfully fulfilled, and fully tested in test 16. The tests went well with the only errors being the menu strip issue mentioned earlier, the issue with the price search, plus a small aesthetic problem caused by a debug message being left in by accident. In terms of the functionality of the system, all issues except the price search were resolved during the development process.
One thing from this requirement I am especially happy with is my custom numeric up down control. I modified a version of this control to add a £ symbol before the number, making it much more visually clear what the purpose of the control was.

The first challenge I faced involved my searches, where artwork searches were not providing output, but this turned out to be that I had my conditions the wrong way around. When the row count returned is greater than zero, the dataset should be passed in but instead I had accidentally used a less than sign, forcing no results for every search.

One thing that required a lot of thought to develop was how to tell the difference between an advanced guest artwork and a normal artwork being created, in order to correctly assign table links. In the end I used methods with an overload to assign only advanced guests an exhibition and a guest link to their artwork, while normal guests will only get a guest link.

I also struggled to work out a way to restrict the update form to only advanced artworks. It turned out the easiest way to manage this was to force a change to advanced view when update is clicked, and then disable the other view from the menu strip.

Initially my update button would not work, and the new record could not be created. This was due to me passing the variables into my stored procedure's method in the wrong order, meaning the datatypes did not match and the record could not be created. I then found that trying to perform an update on a new record would raise a past exhibition error which is impossible, however I had a similar issue before so quickly worked out that my greater than and less than signs were in the wrong order.

Since I was able to resolve all of this before the testing phase I am confident that this requirement has been satisfied to a high standard.

6. The user needs to be able to manage details about existing and new exhibitions:

6.1. The user needs to be able to view details about each exhibition, including the artwork assigned to it.

6.2. The user needs to be able to create an exhibition, but it should be limited to one exhibition annually.

6.3. The user needs to be able to delete an exhibition but only if the exhibition has no artwork assigned to it, and the date has not yet passed.

6.4. The user needs to be able to update an exhibition's date and catering number.

6.5. The system should disable the past and all artwork views when in update or delete mode, as past exhibitions, cannot be deleted or edited.

This requirement has been satisfied and tested in test 15. The only issues I had with this form during testing was the menu strip error mentioned earlier and failed test 15.19, where my month picker was not correctly catching the location of the user's click, however this was resolved quickly by creating a duplicate method and then adjusting the variables and controls used by it. Outside of testing, I also resolved many issues during development.

The first of these problems was actually working out how to restrict the month picker in the first place. I was hoping there would be a

way to restrict it already in the control, but this did not exist so I read through the month picker documentation, to no avail. My next solution was to look through the month picker class itself, and backtracking through variables until I discovered the existence of the HitTest method. Through this I am able to pin point the location of a click, compare it to the location of the control variable TitleBackground, and then quickly enable and disable to interrupt the process. This is not a perfect solution, as the title changes to blue when clicked which may be confusing, but the user cannot accidentally soft lock themselves into the decade view, so it functions as required. The code to evaluate the click location looks like this:

```
private void mcExhibitionDate_MouseDown(object sender, MouseEventArgs e)
{
    MonthCalendar.HitTestInfo x = mcExhibitionDate.HitTest(e.Location);
    if (x.HitArea.ToString() == "TitleBackground")
    {
        mcExhibitionDate.Enabled = false;
        mcExhibitionDate.Enabled = true;
    }
}
```

The next challenge I faced was involved with the common methods class I built for frequently used methods. For my code validation I had developed the method based around the code format of a letter and some numbers, but the exhibition code has two letters. In order to solve this and future proof the method from this point forward, I created an overload method that takes the letters, the number of letters and the code, meaning any number of letters and a code can be validated in the future. The original method is still used elsewhere, and this one overloads it.

The next problem involved my selections in my data grid view. When entering delete mode, the system would throw an out-of-range exception for the selection, because the view would change to future. To solve this, when delete mode is toggled on, the selection will reset before the form changes, meaning the selected row will be the top one in the view, and the exception is no longer thrown.

After solving these issues, I can confidently confirm that this requirement has been fulfilled in a high quality and robust manner.

7. *The user needs to be able to log what equipment a guest uses, and it should be specific to the current course.*

This requirement has been fulfilled well and is tested in test 7. There were no errors during testing except the menu strip problem that was discussed previously, however I came across a couple of small hiccups during development.

First, I had a lot of difficulty working out how to hide the caret on the numeric up down while trying to disable typing in it. I wanted to restrict it to just the up down buttons. Through stack overflow I learnt I could use an imported library that included a method specifically for this purpose. I am then able to call it after every value changed event in the numeric up down.

Secondly, I came across a duplicated primary key issue when logging the same equipment for the same guest on the same course. To fix this, I must perform a check that this type does not exist beforehand, if it does append the new amount, and if not then create a new record. It works perfectly.

Finally, I noticed typing was not correctly disabled in the item type and name drop-down combo boxes, however, this was just a silly mistake, as I had forgotten to set the drop-down type for the other layered combo boxes, only the top one had the correct property.

I am happy that this requirement has been satisfied in a robust and high-quality manner.

8. *The user needs to be able to print off an invoice including the guest's course cost and the added cost of equipment used. This should be an automatic calculation. If the guest sold artwork in the most recent exhibition, and this is their first course their selling price should be deducted from the bill.*

This requirement has been successfully fulfilled, and thoroughly tested in tests 8 and 9. Both tests were passed with no issues raised, but during development, I ran into a variety of problems.

Firstly, I had issues populating report tables. Once again, the tables were blank, but I struggled greatly to create datasets, with exceptions for rows belonging to an existing table. The blank rows were a different issue to last time, and instead, the solution was to clear the data source and reassign it every time the report loads. The second problem was resolved by using methods I had never used before, clone and ImportRow.

Another problem that occurred was working out how to make empty tables and parameters invisible. I thought I had completed the condition correctly, but the issue turned out to be the condition defines hidden rather than shown, which I was unaware of so the solution was to invert my condition.

The final error was I was finding the running total on my report was far too high, but this was simply because I was accidentally using the same variable name twice, meaning there was already a value assigned to the running total before I started counting.

All issues have been resolved and I can confidently state that the requirement has been fulfilled to a high standard.

9. *The system needs to ensure all data entered into the database is entered correctly and attempt to correct it or ask for re-entry if it is not.*

This requirement has been successfully fulfilled and tested in tests 6, 7, 8, 11, 12, 13, 15 and 16. I did not run into any specific issues with this requirement, and I am confident it has been completed to a high standard. I was careful to avoid allowing the user to enter their own input and where possible used multiple choice or record selection instead.

10. *The user should be able to log into the system with a username and password for security. There should also be a password visibility button.*

This requirement has been fulfilled successfully and tested in test 1. This was a simple section, and I ran into no issues, so I am confident that this requirement is robust and high quality.

11. *The user should not be able to enter letters or symbols into phone number or ID fields. The field should also automatically contain the letter that precedes the code or the + before a phone number, and this should be unable to be deleted by the user.*

This requirement is successfully satisfied and was tested in tests 6, 7, 8, 12, 15 and 16. These tests were passed but during development I came across a couple of issues.

All issues came from the phone number entry. Any issues with code methods were discussed in the relevant forms. The first was where the + could be deleted, but not retyped due to the restriction on special characters and letters. The solution was to perform a length check and add the + after every change in text if it is not there.

The second issue was the cursor being set to the start of the text box after each letter is typed. This occurred because the text is removed when checked and replaced afterwards. This is solved by also resetting the cursor position to the index one less than the length of the string.

After solving these phone number issues, I am confident that this requirement is reliably fulfilled on all forms.

12. *In the management of artworks, the user needs to be able to search for artworks by a specific guest ID.*

This requirement has been fulfilled successfully and tested in test 16. I ran into no issues with this requirement and am happy that is fulfilled to a high standard despite being non-essential.

13. *The client needs the pottery system to be intuitive and easy to use as he is moving from a paper-based system. It should include a help document to demonstrate how to use the specific aspect of the application.*

I feel that this requirement has been successfully fulfilled. As it is more aesthetic than functional it has not been tested but the help document is the user guide that has been integrated into the options of the menu strip. I believe my decision to alter the design and compress add update and delete onto one form has improved this requirement and made the system far more user friendly.

14. *The system should be secure and comply with GDPR Standards.*

In summary, to comply with GDPR standards, a database application must adhere to several key principles. Firstly, it should incorporate robust data protection measures, such as encryption and access controls, to safeguard personal data. Secondly, the database should allow for easy data management, including the ability to update or delete personal information upon request. Regular audits and assessments should be conducted to ensure ongoing compliance, with mechanisms in place to promptly address any breaches or incidents. Overall, a GDPR-compliant database prioritizes privacy, security, and accountability in handling personal data.

While audits are in the client, Lakeside Escapes, responsibilities, I believe this requirement has been fulfilled, as the robust login system complies with security and privacy, plus the ability to remove and add records is clean, simple, and easy to use. I have also used Stored Procedures for all queries, meaning user data is protected from the injection of malicious code.

15. *The colour scheme needs to be clean, consistent, and simple.*

This requirement has been fulfilled successfully, as my system adheres to the simple, clean, and user-friendly house style that I defined in the design phase of the project. I am very happy with how the resulting system looks.

16. *The application should have the company logo as the icon on every form.*

This requirement has been fulfilled, as the lakeside escapes icon appears on all forms, and the user guide! I am happy with the quality to which this requirement has been fulfilled.

17. The system should have a consistent house style.

As mentioned in requirement 15, I have adhered to the house style I defined in the design of my program, and this has aided the system in being clean and easy to understand. I am confident this requirement has been satisfied, and my design is consistent.

18. The system should be suitably fast and reliable.

Finally, I believe this requirement has been fulfilled, as all forms and reports will open consistently in under 30 seconds, with all information presents. I believe the system has enough try catch statements to avoid exceptions, and all queries are run using stored procedures to avoid injection of malicious code. I am confident my system is fast, robust, and reliable.

Evaluation of Personal Performance

Development of Personal Skills

Throughout the duration of this project, I have had many opportunities to develop upon my basic Database and SQL understanding from the course's introduction, as well as expand further on my AS C# abilities. I also found it constructive to read extensively on the solution to my issues, in order to learn from those obstacles and ensure they don't arise again. With problems I try to find a solution first by researching the area on the Microsoft documentation or W3Schools, and if it is not resolved, then search the specific problem on Stack Overflow or Reddit. I endeavour to never copy code and instead learn from other solutions before improving upon them. I feel I succeeded in this during my development process.

When developing the Database for my system, I was able to build upon my basic concept of tables and relationships. I wanted to be able to make my primary keys for each table more useful and identifiable than a meaningless number, so planned to add letters such as C for course and G for guest to each auto incremented key. To do this, I researched calculated columns. From here, I learned how important the datatype and length of datatype is, as well as how to use the CAST() function in SQL. By keeping the column not null and persisted, meaning the computed value is stored, not calculated every time, it can then be used as the primary key. The code on how I did this is mentioned in my user requirement evaluation.

Because I had learned from my research on calculated columns, I was then able to use these to make calculations and conditions work for dataset columns as well, as combined with SQL functions, you can declare variables and perform calculations to be stored in a column. One such function in my project is titled CheckFullyBooked, and is used to determine the state of my Fully Booked column in course, by counting the bookings for this course in a separate table!

Cascade Delete was something else I researched, as I felt there must be a simpler way to remove records across multiple tables at once than having to perform a delete on each record. Once I had found it, I was able to implement it in multiple places across my database, making my code far less bulky and repetitive.

When creating queries for my application, including adding, updating and deleting records, I recalled the starter course I initially learned from had mentioned stored procedures as a method to defend a system from injection of unwanted code. While this is not a concern in my system, I decided to learn more about it and in hindsight this was a good idea, as it means all my queries are neatly stored in a folder where I can reuse them again and again in my code, instead of having to repeat strings over and over. My system then has the added bonus of being secure and robust.

Something I had only ever tried once before was reports. I found it easy enough to add a report to a form, but where I hit an issue was passing datasets and parameters into the report! To find this information I had to explore the Microsoft documentations, and browse through a forum on Stack overflow until I found out about parameter arrays. This allows you to pass all parameters and conditions, and even data sets into a report in a single, and in my case very long, array. Any issues I had were simply due to my parameter names not matching the parameter names on the report. On top of this, I learned how to create a dataset from a query instead of just

a table! This allows my reports to use information from multiple tables in my database.

SQL is a language far newer to me than C#, but I found it quite easy to pick up and enjoyed learning about new methods I could use. One feature that made my stored procedures far more efficient was the ability to use IF statements. This meant code that would have had to be performed in C#, and then run one of 3 stored procedures could be condensed into a single stored procedure. One instance was when performing a search:

```
@Type INT,
@Value VARCHAR(250)
AS
IF (@Type = 1)
	SELECT Guest_Artwork.ArtworkID, Guest_Artwork.GuestID, Guest.Forename, Guest.Surname,
	Guest.SkillLevel
	FROM Guest_Artwork INNER JOIN Guest ON Guest.GuestID = Guest_Artwork.GuestID
	WHERE Guest_Artwork.ArtworkID LIKE '%' + @Value + '%';
ELSE IF (@Type = 2)
	SELECT Guest_Artwork.ArtworkID, Guest_Artwork.GuestID, Guest.Forename, Guest.Surname,
	Guest.SkillLevel
	FROM Guest_Artwork INNER JOIN Guest ON Guest.GuestID = Guest_Artwork.GuestID
	WHERE Guest.Forename LIKE '%' + @Value + '%' OR Guest.Surname LIKE '%' + @Value + '%';
```

Another feature of SQL that allowed me to perform conditions inside a stored procedure instead of performing it in C# with the resultant data set, was the ability to compare to the current date. Initially I was using just the CURRENT_TIMESTAMP function, but this compared times too so if it was not precisely the same millisecond it would not work. Instead, I discovered I could combine the function with CAST(), to cast the timestamp to just a DATE. The SQL code looked like this:

```
SELECT Exhibition.ExhibitionID FROM Exhibition
WHERE Exhibition.ExhibitionDate > CAST(CURRENT_TIMESTAMP AS DATE)
```

Finally, as a couple of my report table columns required currency, I wanted to work out how to format as a currency inside the query instead of having to append a '£' to every row of a dataset. To do this I was able to use the SQL FORMAT() function in conjunction with region codes, and it looks like this:

```
SELECT Equipment.ItemName, Guest_Equipment.AmountUsed,
	FORMAT((Guest_Equipment.AmountUsed * Equipment.ItemCost), 'c', 'en-GB')
FROM Guest_Equipment
```

During this project I even learnt new features in C#! The first of these was how to use Icomparable properly. I had heard of this before but never used it, and it became a necessity when trying to sort records by country, as my addresses are one long string. To combat this, I created an address class with its own comparer and sort method, as well as custom exceptions to deal with mis-formatted addresses. This class is something I am very proud of in my code, as I have finally understood Icomparable. My CompareTo interface looks like this:

```
public int CompareTo(object incoming) {
	int returnval = 0; try {
		Address incomingAddress = incoming as Address;
		returnval = Country.CompareTo(incomingAddress.Country); }
	    catch (Exception){}
	return returnval; }
```

In a solution, especially one with the potential to store and manage a very large quantity of records, it is important to make your code as efficient as possible, with as little repetition as you can. In my previous projects this has not been my strong point, so at the start of this development I decided to create a 'Common methods' class, where I

could place repeating and common code and methods used in all forms in my system. This worked very well, and allowed me to manage my menu strips from the same method in every form, as well as code validation methods. When it came to my code validations, I was able to reuse the original code method, and add overloads for extra needs in a form, such as the exhibition code having two characters in comparison to most others having just one. I feel this class has made both testing and developing a little easier and far more efficient.

On the topic of efficiency, one change that I made to my code during development was to create on and off toggles for modes. In my guest form I initially only used the button press events, but this meant I was frequently repeating things and the events were long and complicated. By the time I was onto the second form, I decided to refactor my code to include on/off methods for each mode at the top of the form, so it could be done with one line of code further down than between 5 and 50 lines repeatedly. I then continued to create these on off methods for the rest of my forms and I am very happy with them. One such method looks like this:

```
private void DeleteModeSwap(string type) {
    switch (type) {
        case ("On"):
            UpdateMode = false; DeleteMode = true; AddMode = false;
            lblGuestIDDelete.Visible = true; txtGuestIDDelete.Visible = true;
            txtGuestIDDelete.Text = "G"; btnDeleteGuest.Visible = true;
            dgvGuestsOut.ClearSelection();
            break;
        case ("Off"):
            DeleteMode = false; lblGuestIDDelete.Visible = false;
            txtGuestIDDelete.Visible = false; btnDeleteGuest.Visible = false;
            dgvGuestsOut.ClearSelection();
            break;}
}
```

Something else I had never tried before now was a custom control for my forms. In my Artwork management form, the user can create an advanced artwork and decide on its sale price using a numeric up down. When creating this I disliked how it looked and wanted to add a '£' character before the cost. In order to do this, I created a custom numeric up down with room for system expansion, as it allows you to place any character before the number! This means the system can be used with any currency. This control can be seen inside the Methods folder of my solution.

The final piece of code that I learnt about for the purpose of this project was The HitTest attribute of a month picker. I wanted to be able to lock down the ability to change the month picker view, fixing it in December. To do this, I had to test for the location of the click, and if the area clicked was forbidden, interrupt the click by disabling and reenabling the control. This was hard to work out and involved me reading through the code defining the control to find out how a click is processed. The code I came up with is shown in the user guide evaluation.

Overall, I believe throughout this project I have broadened my understanding and knowledge of C#, on top of learning how to use databases and SQL, which I think I have been able to implement very well! All code was stored on multiple computers and kept carefully backed up, as well as being tested on multiple systems. I am extremely happy with my solution, and believe both my coding abilities and my organisation, file and version management has become far stronger and more efficient while building this solution.

Evaluation of Time Management

Throughout this project, I believe I have learned a significant amount about personal time management. In order to plan the timing of my project, I had decided to use a Gantt chart. Not only did this allow me to have a checklist and guide to visualise how I could reach the completed solution on time, but it also allowed me to pace myself, and avoid cramming in a lot of unrefined work and burning myself out.

I found managing my time with regards to the actual coding very easy. All coding related tasks were completed, with time to spare as these were tasks, I enjoyed, and found it very easy to motivate myself. I started coding (task D) a day or so late, at the start of November, however I finished task U with time to spare, at the end of December! This was, in hindsight, fantastic as it allowed me more time to deal with a chunk of lost progress.

Time management became more difficult when working on design and documentation, as I find these tasks more tedious and it is difficult to motivate myself to do large chunks at once. To combat this, I chose to break up this process with other coursework I had to do at the time, and while I lost a large chunk of testing documentation due to a power outage, I finished my documentation and solution entirely two days early, on the 5th February.

From my activity log I can see that more of my time was spent on backend development rather than frontend. I am pleased with this, as it means my careful planning and storyboard design paid off. Building forms became quick and easy, and I was able to focus on the functionality and robustness of my system.

I am very happy that I succeeded in pacing myself but still finishing the project on time, however if I were to do this again, and in future projects I will leave far more float for challenges such as losing progress, as it was dangerous to rely on finishing other tasks early to allow for this.